CSPA Core Exam™ Topics

The CSPA Core Exam™ covers seven high level topics. To maximize your composite score, you are expected be proficient in CORE, FE, BE, and at least ONE other topic. For details on how we score the exam, please read How Is the CSPA Scored.

Do you have feedback or suggestions about our topics? Let us know.

Algorithms, Data Structures, and Programming (CORE)

This topic covers foundational computer science knowledge. All great software engineers have a solid understanding in the following core principles.

- Algorithms
  - Performance analysis (big-O) for time and space (example)
  - Sorting (example)
  - Tree traversal and manipulation (example)
    - Breadth first search
    - Depth first search
  - Graph algorithms (example)
    - Shortest path, longest path, cycle detection, and more
    - Breadth first search
    - Depth first search
  - Dynamic and linear programming (example)
- Data structures
  - Scalars (numbers, booleans, strings)
    - Endianness
    - Floating point
  - Lists, queues, stacks, arrays, vectors, and more (example)
  - Hash tables, dictionaries, sets, and indexes

    - Trees and graphs (example)
- Programming languages and compilers (example)
    - Interpreted scripts vs. compiled
    - Intermediate bytecode (JVM, .NET CLR)
    - Assembly and machine code
    - Scopes (lexical vs. dynamic, global vs. function vs. block)
    - Typing (strong, weak, static, dynamic)
    - Transpilers
- Programming paradigms and patterns
    - Functional programming (example)
    - Declarative programming
    - Object-oriented programming
    - Recursion
    - Abstractions, interfaces, and separation of responsibilities (example)
    - Map and reduce
    - Factories
    - Listeners and observers
    - Asynchronous and threaded programming

Front end web (FE)

This topic examines your ability to create the client side of functional web applications. High scorers may be suitable for front end web or full stack web roles.

- JavaScript and UI framework concepts (e.g. React, Vue, Angular)
- HTML5 (Common tags and their attributes)
- CSS2/3 (Common styles and basic understanding of rendering algorithms) (example)
- DOM event model (listeners and event handlers)

Back end web (BE)

This topic examines your ability to create the server side of functional web applications. High scorers may be suitable for back end web or full stack web roles.

- Web framework concepts (e.g. MVC, separation of responsibilities, dependency management)
- Data retrieval and storage (APIs, AJAX, REST, ORMs, json, local storage)
- HTTP 1.1 and 2 protocols
    - Request and response format (example)
    - Methods (POST, GET, etc.) (example)
    - Common status codes (200, 404, etc.) (example)
    - Common headers, including cookies, mime types, languages
    - HTTP session management
    - Cross domain concerns
    - SSL/TLS (example)

Operating systems, networking, and tools (OPS)

This topic covers your ability to navigate and build single machines, as well as a network of machines. High scorers may be suitable for devops, technical operations, or systems administrator/IT roles.

- Mathematics and computing foundations
    - Logical operators (bit shifting, AND, OR, XOR, NOT)
    - Regular expressions (example)
- Operating systems
    - File systems (descriptors, inodes, journaling, pipes, sockets) (example)
    - Syscalls and interrupts
    - Processes, concurrency, threading, and scheduling
    - Locks and transactions

    - Memory systems (allocations, paging, virtual addressing, garbage collection)
    - Storage systems and their tradeoffs (SSD, magnetic, cloud) (example)
    - User vs. kernel space
    - Linux shell usage (example)
        - Manipulating files (ls, chmod, cp, mv, tar, zip)
        - Working with text-based files and piping outputs (grep, cut, sed, awk)
        - Monitoring system resources (top, ps, netstat, iostat, etc.)
    - Virtualization and containerization
- Internet infrastructure
    - TCP/IP, IPv4, IPv6, subnets, gateways (example)
    - Domain name system (DNS, whois, ICANN, A/AAAA/CNAME/MX records)
    - Email (SMTP, SPF, DKIM, IMAP, POP)
    - CDNs and load balancers
- Version Control Systems
    - git (command line) (example)
    - Other VCS (Subversion/SVN, Mercurial, Perforce)
    - Branching and tagging strategies
    - Conflicts and merging

Systems design, architecture, and infrastructure (ARCH)

This section tests your abilities to view complex machines holistically. Strong candidates are able to understand concepts at high levels, as well as meticulous details at low levels. High scorers may be suitable for systems architect and infrastructure roles.

- Client-server architecture (example)
- Peer-to-peer architecture

- Design principles
  - The ability to analyze and make tradeoffs on certain design choices
  - Data abstraction, interface design (e.g. APIs, REST), and parametrization choices
  - Techniques for reusability, testability, and maintainability
  - Techniques for reliability of systems
  - Techniques for performance and scalability of systems
- Distributed systems
  - Compute (hadoop and map reduce) (example)
  - Storage (techniques to partition and retrieve data), CAP theorem (example)
  - Availability and reliability concerns (example)
  - Service and resource discovery techniques (example)
  - Techniques to achieve consensus and eventual consistency

Databases and storage systems (DB)

This section tests your knowledge and experience with various data storage systems. High scorers may be suitable for database administrator roles.

- ACID (transactions, strongly and eventually consistent, isolation levels, reliability)
- Relational databases (example)
  - SQL queries and commands (example)
  - Schema design (tables, columns, data types, indexes)
  - Indexes (B-trees, memory hashes, etc.)
  - Compute (stored procedures and triggers)
- NoSQL Databases
  - Document stores
  - Key-value stores
  - Graph stores
  - Search indexes
- Distributed storage, including partitioning and sharding (example)

Security (SEC)

This section tests your ability and experience in defending applications and systems from malicious users. High scorers may be suitable for network and data security roles.

- Encryption, salting, hashing, symmetric/asymmetric (example)
- Access control and authentication (MFA, OAuth)
- Network security (firewalls, ports, subnets, VPNs, SSH)
- Defensive programming
- Data cleaning, including SQL injection
- Secrets management (example)
- Detecting vulnerabilities and common methods of attack
- Security authorities (OWASP, CIS, W3C, PCI SSC, etc.)