PRE PLACEMENT PRESENTATION

NIT
JAMSHEDPUR

Training and Placement Cell, MCA

TABLE OF CONTENTS.

01 HOW’s THE MARKET

 Let’s see how current market is both on/off campus.

GROW THE RIGHT MINDSET...

 Everyone is learning to get a job not to upskill.

- Tech hiring landscape is becoming more competitive, focused, and dynamic.
- The demand is high for skilled problem solvers, creative thinkers and for those who can write clean quality code.
- It's no longer about knowing "everything"—it's about having deep expertise in one area, being able to work collaboratively in teams, and clearly communicating ideas with peers and stakeholders.

GROW THE RIGHT MINDSET...

- Candidates who stand out are the ones who have depth knowledge of one domain and can communicate peer to peer.
- With AI reshaping automation, reducing redundant roles, standing out means showing you're adaptable, curious and consistent.
- The competition is real, but so is the opportunity—for those who prepare smartly and stay grounded in the basics.

02 BARE MINIMUM

 Let’s see the non-negotiable things you should have.

COMMUNICATION IS A MUST

- To crack interviews, your communication doesn't need to be perfect, it just needs to be clear, structured, and confident.
- Focus on explaining your thoughts in simple language, using logical flow: start with the context, explain your approach, and finish with the result.
- Practice speaking about your projects, technical concepts, and problem solving strategies out loud.
- Avoid filler words like "umm" or "like" and instead pause briefly to think.
- With just basic grammar, a calm tone, and consistent practice, you can make a strong impression, even if you're not a native speaker.

BARE MINIMUM IN (DS)

Section 1 - Time and Space Complexity

- Different Notations and Complexity functions
- Difference between different complexities

Section 2 - Data Structures (Theory + OOPs Based implementation)

- Arrays (Jagged Arrays, How Dynamic array expands its size)
- Hash Tables (Collisions, Time complexity tradeoffs)
- Linked List (Implementations of SLL, DLL, CSLL, CDLL)
- Stack and Queue (Implementations of Stack and Queue, Prefix, Infix, Postfix all variants)
- Stack using Queue and Queue using Stack
- Trees (Preorder, Postorder, Inorder Traversals, BFS, DFS)
- Graphs (Representation of Graph, MSTs, Shortest Path finding Algos)
- Greedy and Dynamic Programming (Differences)

BARE MINIMUM IN (ALGO)

Section 1 - Arrays

- Rotation and Reversal
- Largest and Second Largest
- Kadane’s Algorithm
- Next Permutation
- Union and Intersection of two Arrays
- 2 Sum, 3 Sum, 4 Sum all variants
- Pascal Triangle
- Dutch National Flag Algorithm
- Moore’s Voting Algorithm

Section 2 - Matrix

- Rotation of Matrix by 90,180,270,360
- Spiral Traversal
- Matrix indexing tricks

BARE MINIMUM IN (ALGO)

Section 3 - Sorting (Very Important) → Implementations with theory

- Bubble Sort
- Selection Sort
- Insertion Sort, Improved Insertion Sort
- Merge Sort
- Quick Sort

Section 4 - Searching (Very Important) -> Implementation with theory

- Linear and Binary Search
- Binary Search on Answer questions

Section 5 - Hashing

- Hash maps and Sets
- Spiral Traversal
- Follow Neetcode Sheet to solve some Hashing problems (Easy) -> (Medium)

BARE MINIMUM IN (ALGO)

Section 6 - String

- Reverse, Rotate, Palindrome
- Reverse Words
- Anagram, Isomorphism

Section 7 - Linked List (Josephus Problem)

- Sort, Palindrome, Reverse, Rotate
- Tortoise and Hare pattern or Slow or Fast pattern
- Find Middle and Delete middle
- Intersection of 2 LL
- Cycle finding algo all variants

BARE MINIMUM IN (ALGO)

Section 8 - Stack and Queue (Josephus Problem)

- NGE and NSE Patterns
- Stock Span
- Trapping Rainwater
- Largest rectangle in histogram
- Maximal Rectangle

Section 9 - Maths and Bits

- Primes, Factors, GCD, LCM, Power, Nth Root
- Prime Factorisation
- Sieve of Eratosthenes
- Basic Bit tricks learn from TLE Video - https://youtu.be/LGrE0siZ-ZA?si=DWN-nkAI1cdt_cFI

BARE MINIMUM IN (ALGO)

Section 10 - Recursion

- Clear up Theory and Dry Running
- Recursion Tree
- Fibonacci, Factorial, Reverse String
- Sort array, Sort Stack, Reverse Stack
- Nth Stair, Power(x,n)

Section 11 - Interval Problems - Merge, Overlap, N-meeting, My Calender - 1,2

Section 12 - Two Pointers and Sliding Window

- Classic Problems on Subarrays

Section 13 - Greedy - Coin change, Knapsack, Jump Game

A SERIOUS NOTE IN (DSA)

- Try to finish Blind 75, LC - 150 repeatedly many times.
- Everyone must specify the problem number and all platform-IDs in their resume.
- Any good achievement in contests is also worth adding.
- Give meaningful variable names not i,j,a,b.. etc.
- Practice everything without autocompletion in notepad like editor.
- Must also know how to write OOPs based codes.
- You should brainstorm about the problem, then dry run, coding should be the last part and must take the least time.
- Start with extreme brute force approach and improve gradually communicating with the interviewer.
- If stuck ask for hints from interviewer.
- Think and emphasize on corner cases, it has the greatest impact.

BARE MINIMUM IN (PROJECTs)

- Everyone must have at least 2 projects in your resume.
- HTML, CSS, JS (MERN/Spring/Django or Flask/ASP.NET Core is a huge plus) is a must to have to show web programming skills.
- If you are web development guy, make sure you deploy your projects.
- If you have added metrics to improve your ATS, be ready to prove them.
- In Machine learning have proofs in hand about cost, accuracy and error measurements
- Be ready to explain why you have done that specific thing in that specific way.
- Be ready to point out in your code whatever the interviewer is asking.
- Be ready with the problem statement to justify why you have created the project.
- Be ready to answer what are the challenges faced and how you overcame them.

OFF CAMPUS IN (PROJECTs)

- Your project should solve a real-world problem—no matter how small. Recruiters love candidates who think like engineers, not just coders.
- Everyone just builds one of these applications -
  - Chat Apps/Video Streaming
  - CRUD Websites(Todo, Food, Furniture, Education, Library etc..)
  - Ecommerce
  - Booking Systems
  - Crypto Tracker/converter
  - Games (Tic Tac Toe, Snake)
  - Blog Websites
  - URL Shortener
  - News Application
- Either go deep (showcase a well-architected system using good practices: clean code, layered architecture, rate-limiting, auth, CI/CD), or build something that scales (e2ee chat, concurrent booking module).

OFF CAMPUS IN (PROJECTs)

- Mention why you chose a certain stack (why MongoDB over PostgreSQL, why Redis, why WebSockets over polling). This shows maturity.
- Live URL + GitHub repo. Bonus if deployed on Docker/Kubernetes or cloud (Render, Railway, Vercel, or AWS).
- Tailor your project showcase based on where you are applying. For example:
  - Applying to a fintech? Show a billing dashboard or secure API.
  - SaaS startup? A dashboard with multi-user login and analytics.
- Bonus if you have an open-source contribution or your project is used by others (even 5 people). It shows initiative.

BARE MINIMUM IN (OOPs)

Section 1 - OOP Basics

- Understand what OOP is
- Difference between Procedural and OOP
- Real life analogies of OOP

Section 2 - Class & Object

- Know how to define and use classes/objects in code.
- Class syntax and definition
- Creating objects (instances)
- Instance vs static variables and methods
- Constructors (default, parameterized), Destructors
- this keyword
- __init__ or constructor overloading (language-specific)

BARE MINIMUM IN (OOPs)

Section 3 - Encapsulation

- Learn data hiding
- Access modifiers: private, public, protected
- Getters and Setters
- Real life examples

Section 4 - Abstraction

- What is abstraction?
- Abstract classes
- Interfaces
- Partial vs Full Abstraction
- Real-world examples
- Abstract class vs Interface comparison

BARE MINIMUM IN (OOPs)

Section 5 - Inheritance

- Single Inheritance
- Multi-level Inheritance
- Multiple Inheritance (and how it's handled in your language)
- Overriding methods
- super keyword usage
- Constructors in inheritance
- Constructor chaining

Section 6 - Polymorphism

- Compile-time (Method Overloading)
- Run-time (Method Overriding)
- Function Overloading
- Operator Overloading (C++/Python)
- Real-world examples

BARE MINIMUM IN (OOPs)

Section 7 - Access Modifiers & Scope

- private, protected, public
- Access within class, package, subclass
- Language-specific access rules

Section 8 - Composition vs Inheritance

- What is composition?
- When to prefer composition over inheritance
- Real-life examples

Section 9 - Key Keywords and Concepts

- this vs self vs super
- final, const, readonly (language-specific)
- static methods and variables
- Object Slicing (C++)
- Diamond Problem & Virtual Inheritance (C++)

BARE MINIMUM IN (OS)

Section 1 - Process Management

- What is a Process?
- Process States (New, Ready, Running, Waiting, Terminated)
- PCB (Process Control Block)
- Context Switching
- Thread vs Process
- Types of threads (user-level, kernel-level)
- Context switching explanation
- Multithreading vs Multiprocessing

Section 2 - CPU Scheduling Algorithms

- First-Come-First-Serve (FCFS)
- Shortest Job First (SJF) + Preemptive (SRTF)
- Round Robin (RR)
- Priority Scheduling
- Calculate Waiting Time, Turnaround Time

BARE MINIMUM IN (OS)

Section 3 - Threads & Concurrency

- Race Conditions
- Critical Section Problem
- Semaphores & Mutex
- Producer-Consumer Problem
- Deadlock: conditions & avoidance
- Deadlock: 4 necessary conditions
- Explain critical section in multithreaded context

Section 4 - Deadlock Handling

- Deadlock Prevention
- Deadlock Avoidance (Banker's Algorithm — theory only)
- Deadlock Detection
- Recovery methods

BARE MINIMUM IN (OS)

Section 5 - Memory Management

- Logical vs Physical Address
- Paging
- Segmentation
- Virtual Memory
- Page Faults & Thrashing
- Paging & Page Tables (single-level & basic idea of multi-level)
- Virtual Memory concepts
- Difference: Paging vs Segmentation

Section 6 - Page Replacement Algorithms

- FIFO
- LRU (Least Recently Used)
- Optimal (Theoretical)

BARE MINIMUM IN (OS)

Section 7 - File Systems & I/O

- File descriptors
- File Access Methods (Sequential, Direct)
- Inodes (in Unix/Linux)
- Directory structure (Single-level, Multi-level)

Section 8 - Miscellaneous Must-Know Concepts

- Kernel vs User Mode
- System Calls
- IPC (Inter Process Communication) — Pipes, Shared Memory, Message Queues
- Swapping
- Shell vs Kernel

BARE MINIMUM IN (DBMS)

Section 1 - Basics of DBMS

- What is DBMS? Why use it?
- DBMS vs RDBMS
- DBMS vs File System
- Data Models (Hierarchical, Relational, etc.)
- Database Schema (physical, logical, external)
- Instance vs Schema

Section 2 - ER Model & Relational Model

- Entity, Attributes, Keys
- Entity Sets and Relationship Sets
- Types of Attributes (composite, derived, multivalued)
- ER to Relational Mapping
- Keys: Primary Key, Candidate Key, Super Key, Foreign Key
- Identify all keys from a table

BARE MINIMUM IN (DBMS)

Section 3 - Normalization

- Functional Dependency (FD)
- 1NF, 2NF, 3NF, BCNF
- Anomalies (Update, Insert, Delete)
- Lossless Join
- Lossy and Lossless Decomposition

Section 4 - Indexing and Hashing

- Need for Indexing
- Types: Primary, Secondary, Clustered, Non-Clustered
- B+ Trees (basic structure)
- Hashing vs Indexing

BARE MINIMUM IN (DBMS)

Section 5 - Transaction Management

- ACID Properties
- Transactions
- Serializability (conflict, view)
- Schedules (serial, non-serial, recoverable)
- Concurrency Control
- 2PL (Two-Phase Locking)
- Deadlock and Prevention
- Difference between serializable and recoverable schedule
- Example of dirty read/lost update

BARE MINIMUM IN (SQL)

Section 1 - Basics

- Why SQL
- SQL vs NoSQL
- Basic Queries (SELECT, WHERE, GROUP BY, ORDER BY)
- JOINs (INNER, LEFT, RIGHT, FULL)
- Subqueries and Nested Queries
- IN, EXISTS, ANY, ALL
- Aggregate Functions (SUM, COUNT, AVG)
- UNION, INTERSECT, MINUS
- DDL & DML Commands (CREATE, INSERT, UPDATE, DELETE)
- Views

BARE MINIMUM IN (SQL)

Section 2 - Joins and Subquery

- Inner, Outer, Cross Joins
- Self Join
- Natural Join
- Set Operations (Union, Intersect, Except)

Section 3 - Miscellaneous

- What is the difference between WHERE and HAVING?
- Types of joins and when to use them.

BARE MINIMUM IN (CN)

Section 1 - Network Basics

- What is a network?
- Types: LAN, WAN, MAN
- Network Topologies (star, bus, mesh, ring)
- OSI Model (7 Layers — overview + functions)
- TCP/IP Model (comparison with OSI)
- Explain OSI vs TCP/IP
- What happens when you type google.com in browser?

Section 2 - OSI & TCP/IP Layers (High-Level Overview)

- OSI Model, TCP/IP Mode
- Responsibilities of each layer
- Difference: TCP vs UDP

BARE MINIMUM IN (CN)

Section 3 - IP Addressing & Subnetting

- IPv4 vs IPv6
- Classful vs Classless Addressing
- Subnet Masks and CIDR Notation
- Private vs Public IP

Section 4 - Transport Layer (Most Asked Layer)

- TCP vs UDP
- 3-Way Handshake (Connection Establishment)
- Port numbers and multiplexing
- TCP handshake explained clearly
- TCP vs UDP

BARE MINIMUM IN (CN)

Section 5 - HTTP & Application Layer Protocols

- HTTP vs HTTPS
- HTTP Methods (GET, POST, PUT, DELETE, etc.)
- DNS working
- Difference between HTTP and HTTPS
- RESTful API over HTTP
- How DNS resolves a domain

REFER THESE

- For OOPs GFG OOPs section and Youtube, just pick one topic given above and search and learn it.
- For OS refer this Love Babbar Playlist - https://youtube.com/playlist?list=PLDzeHZWIZsTr3nwuTegHLa2qlI81QweYG&si=R69LrSHa0nmRvcbK
- For DBMS you can refer any of these two playlists
  - Love Babbar - https://youtube.com/playlist?list=PLDzeHZWIZsTpukecmA2p5rhHM14bl2dHU&si=0vDWYqGJOldLscum
  - Riti Kumari - https://youtube.com/playlist?list=PLrL_PSQ6q062cD0vPMGYW_AIpNg6T0_Fq&si=o5Fb1DN12eRsfWxU
- For SQL refer this playlist by Riti Kumari - https://youtube.com/playlist?list=PLrL_PSQ6q062H5CetdplYW7xQKeq8XaR0&si=rJeUInU66ogTuO7y
- For practice in SQL complete Hackerrank SQL + LC-50 in SQL.
- For CN refer this one shot by Kunal Kushwaha - https://youtu.be/IPvYjXCsTg8?si=EhQxhzdX265uNU5x

IMPORTANT NOTE...

- Mastering core CS subjects is non-negotiable for cracking top tech roles.
- Companies test these not just for knowledge, but to see how well you understand problem solving, systems thinking.
- Focus on Concept clarity over rote learning.
- Real world mapping (how a DBMS works behind a website)
- Interconnecting concepts (like how OS scheduling affects DB transactions or how networking relates to APIs)
- You don’t need to know everything—just know the right things deeply and how to explain them simply.

03 GOOD OPTIONALS

 You will definitely stand out if you have these.

- If you have done internships or have previous experiences and can demonstrate impactful results, you will stand out.

- If you have contributed to open-source projects it is a huge plus.

- If you have done extreme DSA/CP, can solve any new problem with the topics you are comfortable, you will definitely get good returns.

- If you have good depth of concepts in core Computer Science then you can stand out in every interviews.

- If you have worked on an impactful project that solves a real-world business problem, you will make a huge impact.

– Interviewers of XYZ Company

- If you are not limited to Web Development only and have taken steps to learn various topics like Scalability, System Design, Deployment Principles, and Clean Code writing.

- If you are exposed to other domains like Machine Learning, Blockchain, DevOps, Cloud Computing, Cybersecurity, and Big Data analytics, you can also stand out from the crowd.

- If you have done test driven development.

- If you are exposed to GenAI tools, have integrated them, and created projects, you can also make an impact.
-
- If you have learned everything deeply– for example, in the MERN stack, you understand how MongoDB, Express, React, and Node work behind the scenes.

– Interviewers of XYZ Company

04 FIND YOUR NICHE

 Choose one path to make your career AI-proof.

SYSTEMS PROGRAMMING

- Deep Dive into Microservices, REST APIs, Design patterns, UML, High availability, Load balancing, Caching.
- Learn about design patterns, SOLID principles, clean coding.
- Learn about system designs and how to design scalable system.
- Study real-world architectures used by large-scale systems (e.g., Facebook, Uber).
- Practice designing scalable systems on paper or through online platforms like Exercism.
- Read books on system design (Designing Data-Intensive Applications).
- Write Low Latency code and learn socket level programming.

ML & AI RESEARCH

- Deep Dive into Python, TensorFlow, PyTorch, scikit-learn, data preprocessing, model evaluation, statistics.
- Take online courses on platforms like Coursera or edX to grasp basic ML concepts.
- Learn the mathematics behind algorithms, and apply them in practical use cases like recommendation systems or image recognition
- Work on real-world datasets (Kaggle) and try building models to predict or classify data.
- Explore this video to know more - https://youtu.be/bkhUjwJbP1k?si=7tkyedxzKgci6bLU

AI APPLICATIONS (GenAI)

- Deep Dive into Python, TensorFlow, PyTorch, Text Generation, Image generation, RAGs, Langchain,Audio & Video generations, Deep Reinforcement Learning, Deep Learning, NLP etc.
- Extract APIs of Gemini or chatGPT and play with it.
- Build simple text generators or image creation apps.
- Take online courses (Coursera, fast.ai).
- Participate in AI competitions (e.g., Kaggle).
- Explore this video to know more - https://youtu.be/YVBeVM_fA4k?si=nL_iV1hKEvQizRn7

DEVOPS & CLOUD COMPUTING

- Deep Dive into AWS, Azure, Google Cloud, Docker, Kubernetes, CI/CD pipelines, Ansible, SonarQube, Terraform, Prometheus etc.
- Experiment with deploying your apps on the cloud and automate the deployment process using Docker and Kubernetes.
- Learn from Documentation and experiment with the tools.
- Build projects around infrastructure automation and understand the full software lifecycle from development to production.
- Enroll in Train with Shubham's Course or Harkirat's DevOps Cohort.
- Explore this video to know more - https://youtu.be/1J2YOV6LcwY?si=g6WWSydQ98gnaOvm

CYBER SECURITY

- Deep Dive into Network security, Cryptography, Ethical hacking, Penetration testing, Risk assessment, Firewalls.
- Take part in CTFs (Capture The Flag) to practice ethical hacking.
- Experiment with security tools like Wireshark or Burp Suite to understand vulnerabilities.
- Study real-world cyber attack cases to understand how security systems are breached and how to secure applications.
- Explore this video to know more - https://youtu.be/BoI2wQwa20M?si=DxG778xFz1-48IdT

BLOCKCHAIN & WEB3

- Deep Dive into Smart contracts, Solidity, Ethereum, Decentralized applications (DApps), Cryptography, Decentralized Finance.
- Learn by building your own simple DApp or creating smart contracts on Ethereum using Solidity.
- Follow blockchain projects open source. and start participating in DeFi or NFT development.
- Enroll in Harkirat's Open Source WEB3 Cohort.
- Explore this video to know more - https://youtu.be/D5CGlFQbgnk?si=9iJ9NSSGexIAegLF

MOBILE DEVELOPMENT

- Deep Dive into Swift (iOS), Kotlin (Android), React Native, Flutter, UI/UX principles, SQLite.
- Start with building a simple app that solves a personal problem (like a to-do list).
- Try cross-platform development frameworks like React Native and Flutter.
- Focus on user experience and learn how to make apps that are both functional and engaging.
- Explore this video to know more - https://youtu.be/z8j0nZDo8WQ?si=hrI96hEy-L5b33oD

INTERNET OF THINGS (IoT)

- Deep Dive into Arduino, Raspberry Pi, sensors, embedded systems programming, cloud integration.
- Start with simple projects like creating a smart light or a weather station using Raspberry Pi or Arduino.
- Learn how devices connect to the cloud for data storage and remote control.
- Dive into sensor technologies and communication protocols like MQTT.
- Explore this video to know more - https://youtu.be/MR3sW6vTm5Y?si=feObcuUmiRo9P2GZ

GAME DEVELOPMENT

- Deep Dive into Unity, Unreal Engine, C#, 3D modeling, Physics engines.
- Explore game mechanics, design principles, and physics engines.
- Join game development communities and participate in game jams to challenge yourself.
- Explore this video to know more - https://youtu.be/MjHalxr_tDw?si=K8Prvf1MKuOTDYTq

05 PUBLIC LEARNING & NETWORKING

 Do this if you want to grow very high, very fast.

LEARN & BUILD IN PUBLIC

- Share your learning journey, challenges, and projects with the community
- You receive feedback from others, helping you avoid common mistakes and improving your understanding.
- Publicly documenting your progress pushes you to keep learning and improve continuously.
- It opens opportunities to collaborate with others who share similar interests, boosting learning and networking.

NETWORK MORE

- Deep dive into open source projects and contribute heavily, it will open doors to opportunities.
- Interacting with professionals and mentors offers insights into industry trends, best practices, and new technologies.
- By engaging in the community, you put yourself on the radar of potential employers, collaborators, and mentors.
- Join online forums like Stack Overflow, Reddit, or specialized Discord servers related to your niche.
- Attend conferences, webinars, hackathons, seminars and community discussions where you can make an impact and learn many things.

06 CONCLUDING THOUGHTS

 This is all from our side. Best of Luck Everyone.