# Technical Interview – Homework

## Interactive Interview Simulation with ChatGPT

Practice with a **simulated practical interview session** using **ChatGPT as an interactive technical interviewer**. Open ChatGPT (<https://chat.openai.com>) and enter the following **prompt**:

|  |
| --- |
| **Interview Simulation Prompt for ChatGPT** |
| I am applying for “junior Java developer”. I want to train my interview skills. This is a technical interview with a recruiter. Act as a recruiter, but wait for my response, before you ask me the next question. After each answer I give you, give me feedback by rating my answer on the scale from 1 to 10, also tell me what is missing or incorrect in my answer before asking me the next question. |

My answer: <https://chatgpt.com/share/67e96a4d-18b8-800d-9dda-ddf385a5d599>

## Answer Technical Interview Questions

### Junior .NET Developer – Interview Questions

Answer the following **sample interview questions** to prepare for a technical interview. When you need to write code, or draw something, feel free to use any tool for the purpose (for example Paint), afterwards insert your screenshots in the table below. Finally, upload the filled table with your answers in the course Web site.

|  |  |
| --- | --- |
| 1. Draw a **LinkedList**. How it works? | A LinkedList is a data structure where elements (nodes) are linked together using pointers. Each node contains:   * Data (the actual value) * Pointer (a reference to the next node) * The last node points to NULL, indicating the end of the list. * In a doubly linked list, each node has a pointer to both the next and previous nodes. * Operations: Insertion and deletion are O(1) if you have a reference to the node but O(n) if you have to search for it. |
| 2. Draw a **hash-table**. How it works? | A hash table stores key-value pairs and uses a hash function to compute an index where the data is stored.   * If two keys hash to the same index, collision handling methods like chaining (linked lists) or open addressing (probing) are used. * Time complexity for search, insert, and delete is O(1) on average but degrades to O(n) in worst-case collisions. |
| 3. What is **quicksort**? How does it work? | QuickSort is a divide and conquer sorting algorithm:   * Pick a pivot element. * Partition the array so that elements smaller than pivot go left, greater go right. * Recursively apply QuickSort to both partitions. * Example: Sorting [8, 4, 7, 3, 9] with pivot 7: * Partitioning: [4, 3] (left) 7 [8, 9] (right) * Recursive calls on [4, 3] → [3, 4] * Recursive calls on [8, 9] → [8, 9] * Final sorted array: [3, 4, 7, 8, 9] Best case: O(n log n) | Worst case: O(n²) (if bad pivot choice) |
| 4. What is **bubble** **sort**? How does it work? | Bubble Sort repeatedly swaps adjacent elements if they are in the wrong order. Example:   * Sorting [5, 2, 9, 1]: * (5,2) → swap → [2,5,9,1] * (5,9) → no swap → [2,5,9,1] * (9,1) → swap → [2,5,1,9] * Repeat until sorted [1,2,5,9]. Worst case: O(n²) | Best case (already sorted): O(n) |
| 5. Can you write a **function** that takes a string as input and returns the string with the first letter of each word capitalized? | string CapitalizeWords(string input)  => string.Join(" ", input.Split(' ').Select(word =>  char.ToUpper(word[0]) + word.Substring(1).ToLower())); |
| 6. Can you write a **function** that takes two arrays as input and returns a new array containing the elements that are common to both arrays? | int[] CommonElements(int[] arr1, int[] arr2) =>  arr1.Intersect(arr2).ToArray(); |
| 7. Give me an example of **abstraction** in OOP. | abstract class Animal  {  public abstract void MakeSound();  }  class Dog : Animal  {  public override void MakeSound() => Console.WriteLine("Bark!");  }  Here, ***Animal*** is an abstract class that forces subclasses to implement ***MakeSound().*** |
| 8. Can you explain how does **Liskov** **Substitution** work and give an example? | class Bird { public virtual void Fly() {} }  class Sparrow : Bird { }  class Penguin : Bird { public override void Fly() { throw new Exception("Cannot fly!"); } }  **Wrong**: A ***Penguin*** violates LSP since it cannot fly, but it inherits ***Fly().*** A better solution would be to separate flying behavior into an interface. |
| 9. Give me an example of **polymorphism** in OOP. | class Shape  {  public virtual void Draw() => Console.WriteLine("Drawing a shape.");  }  class Circle : Shape  {  public override void Draw() => Console.WriteLine("Drawing a circle.");  }  Polymorphism allows treating all shapes as a ***Shape*** but calling the overridden method dynamically. |
| 10. Draw a **database schema** for a library management system. You have the following **entities**: Book with fields title, author, ISBN and checked out, Author with fields name, nationality and date of birth, Genre with fields name and description, Librarian with fields name, salary, and books given. Consider the relationships between these entities. | **Tables:**   * Book (id, title, author\_id, ISBN, checked\_out) * Author (id, name, nationality, birth\_date) * Genre (id, name, description) * Librarian (id, name, salary, books\_given)   **Relations:**   * Book → Author (Many-to-One) * Book → Genre (Many-to-Many) |
| 11. Draw a **database schema** for an e-commerce website. You have the following **entities**: Customer with fields name, email and address, Order with fields order number, date and total price, Product with fields name, description price, and category, and Category with fields name and description. Consider the relationships between these entities and ensure that your schema can efficiently handle a large number of orders and products. | **Tables:**   * Customer (id, name, email, address) * Order (id, customer\_id, date, total\_price) * Product (id, name, description, price, category\_id) * Category (id, name, description)   **Relations:**   * Order → Customer (Many-to-One) * Order → Product (Many-to-Many) |
| 12. How do you **join** two tables in SQL? How does “**outer join**” works? | A **JOIN** in SQL is used to combine rows from two or more tables based on a related column, typically a foreign key. An **INNER JOIN** returns only the rows that have matching values in both tables. An **OUTER JOIN**, on the other hand, returns all rows from one table and the matching rows from the other, filling in NULL for missing matches. The main types of outer joins are **LEFT OUTER JOIN**, **RIGHT OUTER JOIN**, and **FULL OUTER JOIN**, each returning rows from the left, right, or both tables, respectively. |
| 13. Given a table of employees with columns for employee ID, name, and salary, write a SQL query to **find the average salary** of employees at the company. | SELECT AVG(salary) AS average\_salary FROM employees; |
| 14. How would you implement **authentication** in a server-side Web app in your favorite MVC framework? | * Use ASP.NET Identity * Hash passwords with bcrypt * Use JWT tokens for API authentication |
| 15. How do you send **email** from a Web app? Which **SMTP** server would you use? | SmtpClient client = new SmtpClient("smtp.gmail.com")  {  Port = 587,  Credentials = new NetworkCredential("yourEmail@gmail.com", "password"),  EnableSsl = true  }; |
| 16. You are developing a web application for an e-commerce website. You have database with tables for products, customers, orders and order details. How would you implement a feature to allow customers to **place an order**? | * Validate customer info * Insert an order record * Insert order details * Deduct stock * Send confirmation email |
| 17. You are building a healthcare app. How would you implement a feature for doctors to **view** a patient’s medical records during an appointment, and allow doctors to **add** new information to the record? | * Store medical records in a PatientRecords table * Implement role-based access (RBAC) * Use transactions for updates |
| 18. You are developing a hotel booking system. How would you implement a feature to **retrieve** all bookings for a particular **date range**? | SELECT \* FROM bookings WHERE check\_in\_date BETWEEN '2025-04-01' AND '2025-04-10'; |
| 19. You are implementing a Web form in ASP.NET MVC. How would you add **validation**? | public class UserModel  {  [Required]  [EmailAddress]  public string Email { get; set; }  }  // Using Data annotation and then using ***ValidationContext*** |
| 20. You are building a social media app. How would you implement a feature to allow users to **post** messages, **view** other user’s post and **like** or **comment** on posts? | * Post table (id, user\_id, content, timestamp) * Like table (id, user\_id, post\_id) * Comment table (id, user\_id, post\_id, text, timestamp) * Use SignalR for real-time updates |