Technical Interview Questions on C

Basic Questions:

1. What are the key features of the C programming language?
2. Explain the difference between C and C++.
3. What are the different data types in C?
4. What is the difference between signed and unsigned data types?
5. What is the use of the sizeof operator in C?
6. Explain the concept of pointers in C.
7. What is the difference between NULL, void, and wild pointers?
8. How does memory allocation work in C?
9. What are malloc(), calloc(), realloc(), and free()?
10. What is the difference between stack and heap memory in C?

Control Structures & Loops:

11. What are the different types of loops in C?
12. How does the for loop work in C?
13. What is the difference between while and do-while loops?
14. What is the purpose of the break and continue statements?
15. How does the switch statement work in C?

Functions & Scope:

16. What is a function in C?
17. What is the difference between call by value and call by reference?
18. What are recursive functions? Give an example.
19. What is the difference between global and local variables?
20. What are static variables and functions in C?

Arrays & Strings:

21. What is an array in C?
22. How do you declare and initialize an array?
23. What is the difference between a one-dimensional and a two-dimensional array?
24. What is a string in C?
25. How does the strcpy() function work?
26. What is the difference between strcmp() and strncmp()?

Pointers & Memory Management:

27. What is pointer arithmetic?
28. How can we pass a pointer to a function?
29. What is a function pointer in C?
30. What is a dangling pointer?
31. What are memory leaks in C, and how can they be prevented?

Structures & Unions:

32. What is a struct in C?
33. How is a struct different from a union?
34. How do you define and access a structure?

35. Can a structure contain a pointer to itself?
36. What is a bit field in a structure?

File Handling:

37. What are the different file handling functions in C?
38. What is the difference between fopen() and freopen()?
39. What are fseek() and ftell() used for?
40. How do you read and write data to a file in C?

Preprocessor & Macros:

41. What are preprocessor directives in C?
42. What is the use of #define in C?
43. How does #include work?
44. What is the difference between #include "file.h" and #include <file.h>?
45. What are conditional compilation directives in C?

Advanced Concepts:

46. What is the difference between a process and a thread?
47. What is the difference between exit(), _exit(), and return in C?
48. What is the volatile keyword in C?
49. What is the use of the restrict keyword in C?
50. What is meant by undefined behavior in C?

Java Technical Interview Questions

Core Java Questions:

1. What are the main features of Java?
2. Explain the difference between JDK, JRE, and JVM.
3. What is the difference between == and .equals() in Java?
4. What is the difference between final, finally, and finalize()?
5. What are wrapper classes in Java?
6. What is the difference between static and instance methods?
7. How does garbage collection work in Java?
8. What are the different types of memory areas allocated by JVM?
9. What is the difference between String, StringBuilder, and StringBuffer?
10. What is the main() method in Java? Can we overload it?

Object-Oriented Programming (OOP) Questions:

11. What are the four pillars of OOP in Java?
12. What is method overloading and method overriding?
13. What is the difference between an abstract class and an interface?
14. Can we create an object of an abstract class? Why or why not?
15. What is multiple inheritance? Does Java support it?
16. What is polymorphism in Java? Give real-world examples.
17. Explain encapsulation with an example.
18. What is the purpose of the super keyword in Java?
19. What is the difference between this() and super()?
20. What is the difference between an interface and a functional interface?

Exception Handling Questions:

21. What are exceptions in Java? How are they handled?
22. What is the difference between checked and unchecked exceptions?
23. What is the difference between throw and throws?
24. Can we have multiple catch blocks for a single try block?
25. What is the purpose of the finally block?
26. What happens if an exception occurs in the finally block?
27. Can we write try without catch?
28. What are custom exceptions in Java?

Multithreading & Concurrency Questions:

29. What is multithreading in Java?
30. What is the difference between Runnable and Thread?
31. What is the difference between synchronized and volatile?
32. What is a deadlock in Java? How do you prevent it?
33. What is the purpose of the wait(), notify(), and notifyAll() methods?
34. What is the difference between ExecutorService and Thread class?
35. How does Callable differ from Runnable?

Collections Framework Questions:

36. What is the Java Collections Framework?
37. What is the difference between ArrayList and LinkedList?
38. What is the difference between HashMap and TreeMap?
39. What is the difference between HashSet and TreeSet?
40. What is the difference between fail-fast and fail-safe iterators?
41. How does ConcurrentHashMap work?
42. What is the difference between Comparable and Comparator?
43. Why is hashCode() used in Java?

Java 8 Features & Functional Programming Questions:

44. What are the key features introduced in Java 8?
45. What is a lambda expression in Java?
46. What are functional interfaces? Give an example.
47. What is a Stream in Java? How is it different from Collections?
48. What is the difference between map() and flatMap() in Streams?
49. What are default and static methods in interfaces?
50. What is the purpose of the Optional class in Java?

JDBC & Database Connectivity Questions:

51. What is JDBC in Java?
52. What are the steps to connect to a database in Java using JDBC?
53. What is the difference between Statement, PreparedStatement, and CallableStatement?
54. What is connection pooling in JDBC?
55. How can we handle SQL injection in JDBC?

Spring & Spring Boot Questions:

56. What is Spring Framework?
57. What are the key features of Spring Boot?
58. What is dependency injection in Spring?
59. What are Spring Boot annotations? Name a few important ones.
60. What is the difference between @Component, @Service, and @Repository?
61. What is the purpose of @RestController in Spring Boot?
62. How do you handle exceptions in Spring Boot?
63. What is the difference between @Autowired and @Qualifier?

Miscellaneous Java Questions:

64. What is serialization in Java? How is it implemented?
65. What is the difference between deep copy and shallow copy?
66. What is the use of the transient keyword in Java?
67. What is the difference between Enumeration, Iterator, and ListIterator?
68. What are design patterns in Java? Can you name a few?
69. What is reflection in Java?
70. What is the purpose of the volatile keyword?

technical interview questions on HTML:

1. What is the difference between and tags in HTML?
2. What is the purpose ofthe <head> tag in an HTML document?
3. Explain the difference between inline, block, and inline-block elements in HTML.
4. What is the use of the alt attribute in the <img> tag?
5. What is the purpose of the <!DOCTYPE> declaration in HTML?
6. Can you explain the difference between the id and class attributes in HTML?
7. What are semantic elements in HTML? Give some examples.
8. What is the <form> tag in HTML? What are the main attributes it can have?
9. What is the difference between the <link> and <script> tags?
10. How can you create a table in HTML? What are the key tags involved?
11. What is the role of the <meta> tag in HTML?
12. How would you include a CSS file in an HTML document?
13. How do you embed a video in HTML5?
14. What is the purpose of the target="_blank" attribute in an anchor (<a>) tag?
15. What is the difference between the <b> and tags?
16. How would you create a dropdown menu in HTML?
17. What is the difference between the <iframe> and <embed> tags?
18. How do you define a hyperlink in HTML? What is the purpose of the href attribute?
19. What is the difference between <ol> and <ul> tags in HTML?
20. How do you add comments in HTML?
21. What is the <canvas> tag in HTML5? How is it used?
22. How can you define the character encoding in an HTML document?
23. What is the role of the <header>, <footer>, and <nav> elements in HTML5?
24. What are the various types of input fields that can be used in an HTML form?
25. How would you create a responsive design using HTML?
26. What is the difference between a <textarea> and <input type="text">?
27. How do you specify an image size in HTML?
28. What is the <legend> tag used for in an HTML form?
29. How do you specify a background image in HTML?
30. What is the difference between and <i> tags in HTML?

Technical CSS interview questions:

1. What is the difference between display: none and visibility: hidden?
2. What is the CSS box model, and what are its components?
3. What is the difference between inline, block, and inline-block elements?
4. Explain the difference between absolute, relative, fixed, and sticky positioning.
5. What is the purpose of the z-index property in CSS? How does it work?
6. How can you center a div horizontally and vertically in CSS?
7. What are pseudo-classes in CSS? Can you give some examples?
8. What is the use of the opacity property in CSS? How does it affect other elements?
9. How do you implement responsive design in CSS?
10. What are media queries, and how are they used?
11. What are the differences between em, rem, px, %, and vh units in CSS?
12. Explain the concept of a "flexbox" layout in CSS. How do you create a flexible container?
13. What is the CSS grid system, and how is it different from flexbox?
14. How do you apply multiple CSS classes to a single HTML element?
15. What are CSS transitions and animations? Can you describe the differences?
16. What is the box-sizing property in CSS, and what is its default value?
17. How would you create a simple CSS dropdown menu?
18. What is the purpose of the float property in CSS, and how can it be cleared?
19. What is a CSS selector, and what are the different types of selectors you can use?
20. Can you explain the concept of CSS specificity and how it works?
21. What are the benefits of using CSS preprocessors like Sass or LESS?
22. How do you use the @import rule in CSS, and what are its alternatives?
23. What is the difference between position: absolute and position: fixed?
24. How can you make a website accessible and visually appealing for both desktop and mobile devices using CSS?
25. What are the performance considerations to keep in mind when writing CSS?
26. How can you style a form using only CSS, making it look appealing?
27. What is the use of the calc() function in CSS?
28. What is the clip-path property in CSS? How can it be used to create shapes?
29. How do you implement a grid-based layout in CSS?
30. What are @keyframes in CSS, and how are they used in animations?

Technical Interview Questions on JavaScript

1. What are the differences between var, let, and const in JavaScript?
2. What is a closure in JavaScript, and how does it work?
3. Explain how JavaScript's event loop works.
4. What is the difference between null and undefined in JavaScript?
5. How do JavaScript promises work, and what is the difference between Promise.all() and Promise.race()?
6. What are JavaScript's data types?
7. Explain the concept of hoisting in JavaScript.
8. What is the difference between synchronous and asynchronous code execution in JavaScript?
9. What is the purpose of the bind(), call(), and apply() methods in JavaScript?
10. How do you create an object in JavaScript?
11. What is the difference between == and === in JavaScript?
12. What is a prototype in JavaScript, and how does inheritance work?
13. Explain the difference between function declarations and function expressions.
14. What are arrow functions, and how are they different from regular functions in JavaScript?
15. What is the purpose of the setTimeout() and setInterval() functions in JavaScript?
16. What are the different ways to handle errors in JavaScript?
17. What is the spread operator and how is it used in JavaScript?
18. Explain the concept of "this" in JavaScript.
19. What are closures, and why are they useful in JavaScript?
20. What is event delegation in JavaScript?
21. How does JavaScript handle type coercion?
22. What is the difference between forEach(), map(), filter(), and reduce() in JavaScript?
23. What are the benefits of using async/await over traditional promise chains?
24. How do you prevent a default action in JavaScript (e.g., in an event handler)?
25. What is the purpose of Object.freeze() in JavaScript?
26. How would you clone an object in JavaScript?
27. What is a "debounce" function in JavaScript, and when would you use it?
28. What is a "throttle" function in JavaScript, and how does it work?
29. What are "IIFE" (Immediately Invoked Function Expressions) and how do they work in JavaScript?
30. What is the difference between localStorage, sessionStorage, and cookies in JavaScript?
31. What is destructuring assignment in JavaScript?
32. How do you handle asynchronous operations using async/await in JavaScript?
33. What is the difference between window and document objects in JavaScript?
34. What is the purpose of Symbol() in JavaScript?
35. How do you make an HTTP request in JavaScript?

36. What is a "callback hell," and how can you avoid it in JavaScript?
37. What are JavaScript modules, and how do you import/export them?
38. What is the difference between Array.prototype.slice() and Array.prototype.splice() in JavaScript?
39. What is event.preventDefault() and event.stopPropagation()?
40. How would you deep clone an object or array in JavaScript?

Technical Interview Questions on jQuery

1. What is jQuery, and why is it used in web development?
2. Explain the concept of "chaining" in jQuery with an example.
3. What are the differences between jQuery's .bind(), .live(), and .on() methods?
4. How can you select an element using jQuery by its ID, class, and attribute?
5. What is the purpose of the $(document).ready() function in jQuery?
6. What is the difference between .html() and .text() methods in jQuery?
7. How do you handle events using jQuery? Can you provide an example of event delegation?
8. What is the difference between .val() and .text() in jQuery?
9. How can you make an AJAX request using jQuery?
10. How do you animate elements with jQuery? Provide an example of animating an element's width.
11. What are jQuery's .fadeIn() and .fadeOut() methods used for?
12. How do you create a custom animation in jQuery?
13. What is the difference between .attr() and .prop() in jQuery?
14. How can you add, remove, or toggle classes on an element in jQuery?
15. What is the purpose of $.each() in jQuery? How is it different from forEach() in JavaScript?
16. What is the significance of jQuery's this keyword? How does it behave in different contexts?
17. How can you prevent the default behavior of an event in jQuery?
18. How can you get and set the CSS styles of an element using jQuery?
19. What are jQuery plugins, and how do you create a custom jQuery plugin?
20. How do you handle errors in jQuery AJAX requests?
21. How can you check if an element is visible or hidden using jQuery?
22. How do you traverse the DOM using jQuery methods like .parent(), .children(), and .siblings()?
23. What is the purpose of jQuery's $.Deferred() object?
24. How do you manipulate DOM elements by their position using jQuery (e.g., first, last, nth-child)?
25. What is event delegation, and why is it important in jQuery?
26. How do you use jQuery to serialize form data into a query string?
27. What is the difference between .show() and .css('display', 'block') in jQuery?
28. How can you delay or queue multiple animations in jQuery?
29. What is the $(this) keyword, and how does it behave in event handling?
30. How can you remove an element from the DOM using jQuery?

Technical interview questions on PHP:

1. What are the differences between echo and print in PHP?
2. How do you handle errors in PHP? What are the different types of errors in PHP?
3. What are PHP sessions and how do they differ from cookies?
4. What is the purpose of isset() and empty() functions in PHP?
5. Explain the concept of PHP traits. How are they different from classes and interfaces?
6. How can you prevent SQL injection in PHP?
7. What is the use of the __construct() method in PHP?
8. What is the difference between == and === in PHP?
9. Explain the difference between include() and require() in PHP.
10. What are the different types of loops available in PHP?
11. How does PHP handle type conversions (casting)?
12. What is the use of the header() function in PHP?
13. What are the differences between GET and POST methods in PHP?
14. What is the difference between $_GET, $_POST, and $_REQUEST in PHP?
15. What is the purpose of the explode() and implode() functions in PHP?
16. What is the PDO extension in PHP, and how does it help in database handling?
17. What are PHP superglobals and give examples.
18. How can you define constants in PHP?
19. What are the visibility types for class properties and methods in PHP?
20. Explain the use of abstract classes and methods in PHP.
21. How would you handle file uploads in PHP?
22. What is NameSpace in PHP, and why is it used?
23. What is the difference between array_map() and array_walk() in PHP?
24. How do you debug a PHP application?
25. What is the spl_autoload_register() function in PHP? How does it work?
26. Explain PHP Data Objects (PDO) and how they differ from mysqli.
27. What are magic methods in PHP? Provide examples.
28. What are anonymous functions (closures) in PHP? How are they used?
29. What is the serialize() function in PHP? How does it work?
30. How do you manage cookies in PHP?
31. What are some ways to optimize the performance of a PHP application?
32. Explain the concept of MVC in PHP.
33. How can you send emails using PHP?
34. What is the difference between php.ini and .htaccess in PHP configuration?
35. How do you connect to a MySQL database in PHP using PDO?
36. What is the purpose of the date() function in PHP?
37. What are the differences between session_start() and session_destroy() in PHP?
38. How do you prevent file inclusion vulnerabilities (LFI/RFI) in PHP?
39. What are the advantages of using prepared statements in PHP?
40. What is composer in PHP, and how is it used to manage dependencies?

Basic DBMS Questions:

1. What is a Database Management System (DBMS)?
2. What are the differences between DBMS and RDBMS?
3. What is a primary key? Can a primary key contain null values?
4. What is a foreign key? How is it different from a primary key?
5. What is normalization? Why is it important?
6. What are the different types of normal forms in DBMS?
7. What is denormalization? Why is it used?
8. What is a schema in a database? What is the difference between schema and instance?
9. What is a DBMS index? What are its types?
10. What is a view in DBMS, and how is it different from a table?

Intermediate DBMS Questions:

11. What are the advantages of using indexing in a database?
12. What is the ACID property in DBMS? Explain each property.
13. What is a transaction in DBMS? What are the properties of a transaction?
14. What is the difference between a clustered and a non-clustered index?
15. What is referential integrity, and how is it maintained in DBMS?
16. What is a join? Can you explain different types of joins in SQL?
17. What is a subquery? How is it different from a join?
18. What is an ER diagram, and what are its components?
19. What is SQL injection, and how can it be prevented?
20. What are triggers and stored procedures? How are they different?

Advanced DBMS Questions:

21. What is the difference between a temporary table and a permanent table in SQL?
22. What is the difference between an inner join and an outer join?
23. What is normalization, and what are the different types of normal forms?
24. Explain the concept of a deadlock and how to avoid it in DBMS.
25. What is a transaction log, and why is it important in DBMS?
26. What is a lock in DBMS? What are the types of locks?
27. What is the difference between a shallow copy and a deep copy in the context of database operations?
28. What are the various types of anomalies in DBMS, and how can they be prevented?
29. What is sharding, and how does it improve the performance of a database?
30. Explain the concept of horizontal and vertical scaling in a database.

SQL and Querying Questions:

31. Write an SQL query to find the second-highest salary from an Employee table.
32. What is the difference between the WHERE clause and the HAVING clause in SQL?
33. Write an SQL query to retrieve all the employees who have been hired in the last 6 months.
34. How can you find the duplicate records in a database using SQL?

35. Explain the difference between UNION and UNION ALL in SQL.
36. What is the difference between TRUNCATE and DELETE commands in SQL?
37. What are aggregate functions in SQL? Provide examples.
38. How would you design an efficient database for a large-scale e-commerce platform?

Database Design and Optimization Questions:

39. How would you optimize a database query that is taking too long to execute?
40. What is database denormalization, and when would you use it?
41. How do you ensure data integrity and consistency in a multi-user database system?
42. What are the advantages and disadvantages of storing large binary data (like images, videos) in the database versus in the file system?
43. How do you handle data redundancy in a large-scale database?
44. What is the role of a database administrator (DBA)?
45. How do you perform database backup and recovery?
46. How would you handle the scalability of a database for high traffic applications?
47. What is partitioning in a database, and why would you use it?
48. What are the different types of database relationships (one-to-one, one-to-many, many-to-many)?
49. How would you design a database for a social media platform?
50. How do you handle large-scale data migration from one database system to another?

Data Structure

1. What is a data structure?
2. What are the types of data structures?
3. What is the difference between an array and a linked list?
4. What is a stack? Can you explain its operations?
5. What is a queue? What are its different types (e.g., circular queue)?
6. Explain the difference between a stack and a queue with examples.
7. What is the time complexity of accessing an element in an array and in a linked list?
8. What are the advantages and disadvantages of arrays?
9. What is a doubly linked list? How is it different from a singly linked list?
10. What is a binary tree? What are its types?
11. What is a binary search tree (BST)? How is it different from a regular binary tree?
12. What are the different types of tree traversal methods?
13. What is a heap data structure? Can you explain max heap and min heap?
14. What is a graph? What are the different types of graphs (e.g., directed, undirected, weighted)?
15. Explain the difference between depth-first search (DFS) and breadth-first search (BFS).
16. What is hashing, and how does it work? What is a hash table?
17. What are collisions in a hash table? How can they be resolved?
18. What is a priority queue? How does it work?
19. Can you explain the difference between an array and a linked list in terms of memory allocation and access speed?
20. What is a circular linked list?
21. What is a trie data structure? Where is it used?
22. What is the difference between a directed graph and an undirected graph?
23. What are the operations supported by a linked list?
24. How do you reverse a linked list?
25. What is the time complexity of inserting and deleting elements in a stack and queue?
26. What is the purpose of a sentinel node in a linked list?
27. How do you detect a cycle in a linked list?
28. What is the difference between a depth-first search (DFS) and a breadth-first search (BFS) algorithm?
29. How do you implement a graph using an adjacency matrix and an adjacency list?
30. Can you explain the difference between an AVL tree and a red-black tree?
31. How would you implement a stack using queues?
32. What is the time complexity of searching in a hash table?
33. How do you implement a queue using two stacks?
34. What is a "balanced tree"?
35. Can you explain the concept of dynamic programming and how it relates to data structures?

General Internet Knowledge:

1. What is the Internet?
2. Explain how the Internet works.
3. What is the difference between the Internet and the World Wide Web (WWW)?
4. What is DNS (Domain Name System), and why is it important?
5. What is an IP address? What are the differences between IPv4 and IPv6?
6. What is the purpose of a web browser? Name some commonly used browsers.
7. What is a URL (Uniform Resource Locator)? Explain its components.
8. What is an HTTP request, and how does it work?
9. What is the difference between HTTP and HTTPS? Why is HTTPS considered more secure?
10. What is a firewall, and how does it help in network security?
11. What is a VPN (Virtual Private Network), and why is it used?
12. What is the difference between a LAN (Local Area Network) and a WAN (Wide Area Network)?
13. What is the purpose of a router in a network?
14. What is a proxy server, and how does it work?
15. What is bandwidth, and how is it different from latency?
16. What is cloud computing, and how does it relate to the Internet?
17. What are the different types of Internet connections (e.g., Fiber, DSL, Cable, Satellite)?
18. What are cookies in web development, and what are they used for?
19. Explain the concept of latency in networking. How does it affect performance?
20. What is the role of TCP/IP in Internet communication?
21. What is a web server? Can you name a few popular web servers?
22. What is a DNS lookup, and how is it performed?
23. What are the primary functions of an operating system related to the Internet?
24. What is a CDN (Content Delivery Network), and how does it improve the user experience?
25. What are the different types of network topologies?
26. What is a MAC address, and how is it different from an IP address?
27. What is a TCP handshake, and why is it important in establishing a network connection?
28. What are SSL/TLS certificates, and how do they ensure security in online transactions?
29. What is the difference between a public and a private IP address?
30. What is an API (Application Programming Interface) in the context of the Internet?

Web Development and Technologies:

1. What is HTML? How is it used in web development?
2. What is the role of CSS in web development?
3. What is JavaScript, and how does it contribute to interactive web pages?
4. What are web frameworks? Can you name a few popular ones?
5. Explain the difference between front-end and back-end development.
6. What is the role of AJAX in web development?
7. What is a responsive web design, and why is it important?
8. What is RESTful API, and how does it work in web development?
9. What is a CMS (Content Management System), and why is it used?
10. Explain the concept of client-side and server-side scripting.

Security and Privacy:

1. What are the basic security risks associated with the Internet?
2. What is phishing, and how can you protect against it?
3. What are the most common types of cyberattacks?
4. Explain what encryption is and why it's important for Internet security.
5. What is two-factor authentication (2FA)?
6. What is malware, and how can it affect users on the Internet?
7. What is the role of HTTPS in web security?
8. What is a DDoS (Distributed Denial of Service) attack?
9. What are security certificates, and why are they important for websites?
10. What is the difference between a virus and a worm in terms of Internet security?

General Introduction & Background

1. Can you tell me a little about yourself?
2. Why did you choose your specific field of study?
3. What motivated you to apply for this role/company?
4. How would you describe your strengths and weaknesses?
5. What achievements are you most proud of during your academic career?
6. Why do you think you are a good fit for this position?
7. What are your short-term and long-term career goals?

Skills & Knowledge

8. How do you keep yourself updated with the latest trends and technologies?
9. Can you explain a project you worked on during your studies?
10. What programming languages are you most comfortable with, and why?
11. Which software tools are you proficient in?
12. How do you approach problem-solving when faced with a complex issue?
13. Have you ever worked on a team project? How did you contribute?

Work Style & Personality

14. How do you handle pressure and tight deadlines?
15. Can you describe a time when you faced a challenge and how you overcame it?
16. How do you manage your time and prioritize tasks?
17. How do you approach learning new skills or technologies?
18. Do you prefer working independently or as part of a team? Why?
19. What kind of work environment do you thrive in?

Work Ethic & Motivation

20. How do you stay motivated during difficult or repetitive tasks?
21. Can you describe a time when you had to go above and beyond to achieve something?
22. How do you handle constructive criticism or feedback?
23. Do you think work-life balance is important? Why?
24. How do you ensure you maintain a high level of productivity?

Problem-Solving & Adaptability

25. Tell me about a time when you had to adapt to a change at school or in a project.
26. How do you approach learning a new concept or skill that seems challenging?
27. If you were given a task you had no prior knowledge about, how would you go about completing it?
28. Describe a situation where you worked with someone difficult or challenging. How did you handle it?

Company-Specific Questions

29. What do you know about our company and its products/services?
30. Why do you want to work with us instead of other companies in the same industry?
31. How do you think you can contribute to our team or organization?
32. What attracts you to this particular role or department?

Future Aspirations

33. Where do you see yourself in the next 3-5 years?
34. How do you plan to develop your career moving forward?
35. What skills or knowledge do you want to gain from this role?

Behavioral Questions

36. Can you give an example of a time when you worked in a group and had a conflict? How did you resolve it?
37. What motivates you to perform your best at work?
38. Can you describe a situation when you had to show leadership?
39. How would you handle a situation where you have multiple tasks to complete and limited time?

Closing Questions

40. What is your availability to start if hired?
41. Do you have any questions for us about the company or the role?

Task 1: Create a Simple Blog Application

Objective:

Develop a basic blog application with a homepage that displays a list of blog posts. Each post should have a title, content, and a date. Additionally, there should be a contact form that sends the form data to a PHP script.

Requirements:

1. Homepage (index.php)
    - The homepage should display a list of 3 blog posts.
    - Each blog post should show:
        - Title (use placeholder text).
        - Content (use placeholder text).
        - Date (static date).
    - Style the homepage using CSS to make it look clean and simple.
    - The homepage should have a navigation bar with links to:
        - Homepage
        - Contact Form
        - Admin Panel (For this task, it will just be a mock-up)
2. Contact Form Page (contact.php)
    - Create a contact form with the following fields:
        - Name (required)
        - Email (required)
        - Message (required, text area)
    - Add a submit button to send the form data.
    - The form data should be sent to send_contact.php (using PHP) where the data will be validated and displayed.
3. Backend (send_contact.php)
    - Create a PHP script to receive the data from the contact form and validate it:
        - Ensure Name and Email fields are not empty.
        - Email should be in a valid email format.
        - If validation is successful, display the form values back to the user.
        - If validation fails, display error messages and prompt the user to fill the form again.
4. Admin Panel (admin.php) (optional, just a mock-up)
    - Create a simple admin panel with the following features:
        - A text input for adding a new blog post title and content.
        - A button to save the post (no actual database storage required for this task, just a form submission with a success message displayed).
5. Frontend (HTML, CSS, JavaScript):
    - Use HTML and CSS to make the pages visually appealing.
    - Add basic JavaScript to the contact form:

        - Perform a simple client-side validation to ensure that none of the fields are empty before submitting the form.
    - Optionally, use JavaScript to add interactivity, like form field validation alerts or dynamic content changes.

Evaluation Criteria:

1. Frontend Skills:
    - Proper usage of HTML for structure.
    - Effective application of CSS for styling.
    - Basic JavaScript for form validation.
    - Simple, clean UI/UX.
2. Backend Skills:
    - PHP for form data handling and validation.
    - Use of basic PHP techniques for processing form submissions.
    - Proper use of PHP functions for validation and error handling.
3. Code Organization:
    - Separation of concerns (frontend and backend code should be separate).
    - Clean, well-commented code for readability.
    - Proper naming conventions for files, functions, and variables.
4. Error Handling:
    - Basic error handling for form submissions.
    - Graceful handling of invalid data with appropriate feedback to the user.

Bonus Points (Optional):

- Implement a simple blog post page where a user can click on a post's title and view the full content of the post (without linking to a separate page, you can use JavaScript to toggle the content visibility).
- Use AJAX to submit the contact form without reloading the page.

Task 2: Create a Simple Online To-Do List Application

Objective:

Develop an interactive To-Do List application where users can add, delete, and view tasks. The tasks should be stored in a PHP session (no database needed for this task). The app should have a simple, clean UI, with functionality achieved using HTML, CSS, PHP, and JavaScript.

Requirements:

1. Homepage (index.php)
    - Display the list of tasks dynamically from the session.
    - Each task should have:
        - Task Name
        - Date Added
        - A Delete Button to remove the task.
    - The page should have an input field and a submit button to allow users to add a new task.
    - When a new task is added, it should appear at the top of the list.
2. Task Add Functionality:
    - Use PHP sessions to temporarily store tasks.
    - Each task should have:
        - Title (input field)
        - Description (textarea)
        - Date Added (auto-generated using PHP date functions)
    - When the user submits a new task, it should be stored in the session and displayed on the homepage.
3. Task Deletion:
    - Add a Delete button for each task.
    - When clicked, the task should be removed from the session and the page should refresh, showing the updated list of tasks.
4. Styling (CSS):
    - The page should be styled to make the to-do list clean and user-friendly.
    - Use flexbox or grid for layout.
    - Make sure the delete button is clearly visible next to each task.
5. Frontend (JavaScript):
    - Implement a JavaScript confirmation when the user clicks the Delete button (show a confirmation dialog: "Are you sure you want to delete this task?").
    - Optionally, add animations or effects when a task is added or removed.

Steps to Complete the Task:

1. Create the structure of the webpage (HTML).
    - Build the form to add a task.
    - Display the tasks dynamically.
2. Write PHP code to manage tasks using PHP sessions:
    - Store and retrieve tasks from the session.

    - Handle the addition and deletion of tasks.
3. Style the page using CSS to give it a neat and professional look:
    - Design the layout.
    - Style the task list, task items, and buttons.
4. Add interactivity using JavaScript:
    - Confirmation dialog for task deletion.
    - Dynamic content update on task addition and deletion.

Evaluation Criteria:

1. Frontend Skills:
    - Proper use of HTML to structure the page.
    - Effective application of CSS to style the tasks and page layout.
    - Basic JavaScript for interactivity (delete confirmation).
2. Backend Skills:
    - Correct usage of PHP sessions to store and manage tasks.
    - Proper form handling in PHP (task submission and task deletion).
3. Code Organization:
    - Code should be well-organized with proper indentation and comments.
    - PHP code should be in a separate block, not mixed with HTML.
    - Tasks should be dynamically displayed based on the session data.
4. UI/UX:
    - Clean and easy-to-navigate design.
    - Clear labeling of buttons and input fields.
    - The delete button should be easy to locate and accessible.

Bonus Points (Optional):

- Implement input validation to ensure the task name is not empty before submission.
- Use AJAX to dynamically add and remove tasks without reloading the page.
- Allow users to mark tasks as completed (using checkboxes) and style completed tasks differently.

Task 3: Blog Post Management System

Objective:

Create a Blog Post Management System where users can perform the following operations:

- View all blog posts.
- Add a new blog post.
- Edit an existing blog post.
- Delete a blog post.

Requirements:

1. MySQL Database:
    - Create a MySQL database named blog_system.
    - Create a table posts with the following fields:
        - id (Primary Key, AUTO_INCREMENT)
        - title (VARCHAR, not null)
        - content (TEXT, not null)
        - created_at (DATETIME, default current timestamp)
2. Frontend (HTML, CSS, JavaScript):
    - Create a Homepage that displays all blog posts from the database.
    - Each post should show:
        - Title (clickable to view more).
        - Content (short preview).
        - Date (timestamp of creation).
        - Buttons to Edit and Delete (these will trigger PHP scripts).
    - Use CSS for styling the page to be clean and simple.
    - Implement JavaScript for basic validation when adding or editing blog posts.
3. Blog Post Operations:
    - Add New Blog Post:
        - Create a form (with fields for title and content) to add a new blog post.
        - Validate that the fields are not empty using JavaScript.
        - Upon form submission, use PHP to insert the new post into the database.
    - Edit Blog Post:
        - When clicking Edit, navigate to a new page with a form pre-filled with the post's data.
        - After updating the fields and submitting the form, use PHP to update the post in the database.
    - Delete Blog Post:
        - When clicking Delete, prompt the user with a confirmation message using JavaScript.
        - Use PHP to delete the selected post from the database.
4. PHP Backend:

- Connection to MySQL Database: Use PDO or MySQLi to connect to the MySQL database.
- CRUD Operations:
  - Create a PHP file to handle the Create (Add), Read (View), Update (Edit), and Delete (Remove) operations for blog posts.
  - Use prepared statements to ensure safe database operations (avoid SQL injection).
  - Fetch blog posts from the database and display them on the homepage.

5. Security:
   - Ensure basic validation and sanitization of input data (e.g., title and content) to prevent malicious code injections.
   - Use prepared statements in PHP to avoid SQL injection vulnerabilities.
6. Pagination (Optional):
   - Implement basic pagination on the homepage to limit the number of posts shown per page (e.g., 5 posts per page).

Evaluation Criteria:

1. Frontend Skills:
   - Proper usage of HTML to structure the page.
   - Effective styling with CSS to make the page visually appealing.
   - Basic JavaScript validation for form fields (title and content).
2. Backend Skills:
   - Connection to MySQL database using PHP (using PDO or MySQLi).
   - Correct implementation of CRUD operations (Create, Read, Update, Delete).
   - Secure use of prepared statements to interact with the database.
3. Code Organization and Structure:
   - Code should be organized with separation of concerns (frontend, backend, database operations).
   - Proper naming conventions for files, functions, and variables.
   - Use of functions or classes for reusability and maintainability.
4. Security Considerations:
   - Ensure basic input validation and sanitization for security purposes.
   - Use of prepared statements for all database queries to avoid SQL injection.
5. Usability:
   - The application should be user-friendly, with clear instructions for adding, editing, and deleting posts.
   - The homepage should display blog posts in a clean, organized manner.

Instructions:

- Time Limit: Complete the task within 3-4 hours.
- Submission: Submit the code as a ZIP file containing:
  - PHP files for the backend (CRUD operations, connection).
  - HTML, CSS, and JavaScript files for the frontend.

    - SQL script to create the database and table.
- Bonus Points:
    - Add pagination to the homepage if you feel comfortable.
    - Implement a search functionality to search posts by title or content.
    - Use AJAX for form submissions to avoid page reloads (optional).

Task 4: User Registration, Login System with CRUD Operations

Objective:

Create a web application with a user registration and login system. Users should be able to sign up, log in, and manage their profiles. Additionally, the application should allow the admin to manage (CRUD) user data.

Requirements:

1. Database (MySQL):
    - Create a MySQL database named user_system.
    - Create a table named users with the following columns:
        - id (Primary Key, Auto Increment)
        - username (Unique, VARCHAR)
        - password (VARCHAR, hashed)
        - email (VARCHAR)
        - created_at (DATETIME)
        - updated_at (DATETIME)
2. Registration Page (register.php):
    - Create a registration form with the following fields:
        - Username (required)
        - Password (required, must be hashed using PHP's password_hash() function)
        - Email (required)
    - On form submission, store the data in the users table in the database (ensure the password is hashed).
    - After successful registration, redirect to the login page.
3. Login Page (login.php):
    - Create a login form with Username and Password fields.
    - On form submission, validate the username and password by checking the data in the users table (use password_verify() to compare hashed passwords).
    - If successful, store the user's session information (use PHP sessions) and redirect to the profile page.
    - If login fails, display an error message.
4. Profile Page (profile.php):
    - Create a profile page that shows the logged-in user's username and email.
    - The page should allow the user to:
        - Update their email (only email can be updated, not the username or password).
        - Change password (validate the old password, then hash and update the new password).
    - Store changes in the users table and display success/error messages.
5. Admin Panel (admin.php):
    - Admin should be able to:

        - View a list of all users (Display their username, email, and registration date).
        - Delete users (Delete a user from the users table).
        - Update user details (Update username and email; password cannot be changed by the admin).
6. Frontend (HTML, CSS, JavaScript):
    - Use HTML to create the registration, login, profile, and admin pages.
    - Style the pages using CSS to make the layout clean and user-friendly.
    - Use JavaScript for basic form validation (ensure fields are not empty, password format, etc.).
    - Optionally, use JavaScript to show/hide password or confirm password for the registration and profile pages.
7. Security Considerations:
    - Hash the passwords using password_hash() in PHP.
    - Use prepared statements (with MySQLi or PDO) to avoid SQL injection attacks.
    - Ensure that sessions are used properly for login management (e.g., using session_start()).

Bonus Points (Optional):

- Implement a forgot password functionality where the user can reset their password via email.
- Use AJAX to update user profile information without reloading the page.
- Implement pagination in the admin panel to view users.

Evaluation Criteria:

1. Database Design: Properly designed MySQL database and table with relevant fields.
2. Frontend Skills:
    - Proper use of HTML and CSS for creating clean and responsive layouts.
    - JavaScript for client-side validation and interactivity.
3. Backend Skills:
    - Correct use of PHP for handling user registration, login, and profile management.
    - Proper password hashing and validation.
4. Security:
    - Correct use of prepared statements to prevent SQL injection.
    - Proper use of sessions for maintaining login state.
5. Functionality:
    - Complete CRUD functionality for users (both frontend and backend).
    - Correct error handling and success messages displayed to the user.
6. Code Organization:
    - Well-structured and clean code with proper comments and naming conventions.