100 Promises & Async/Await Technical Questions

Basic Promise Creation & Handling

1. Write a function to create a Promise that resolves after 2 seconds with message "Hello World"
2. Create a function that returns a Promise which rejects with error message "Something went wrong"
3. Write a function using .then() to fetch user data and display their name in console
4. Create a function using .catch() to handle API errors and show user-friendly message
5. Write a function using .finally() to hide loading spinner regardless of success or failure
6. Create a function that chains multiple .then() calls to process user data step by step
7. Write a function to convert callback-based setTimeout into Promise-based delay function
8. Create a function that returns Promise resolving with random number between 1-100
9. Write a function using Promise constructor to validate user input asynchronously
10. Create a function that wraps XMLHttpRequest in Promise for making HTTP calls

Async/Await Fundamentals

11. Write an async function to fetch user profile and await the response before processing
12. Create an async function that waits for file upload completion and returns success status
13. Write a function using await to pause execution until database connection is established
14. Create an async function that fetches weather data and awaits JSON parsing
15. Write a function using async/await to login user and wait for authentication token
16. Create an async function that downloads image and awaits blob conversion
17. Write a function using await to wait for form validation before submitting data
18. Create an async function that loads configuration file and waits for parsing completion
19. Write a function using async/await to calculate total from multiple async price lookups
20. Create an async function that waits for user permission before accessing camera

Error Handling with Try-Catch

21. Write an async function using try-catch to handle API call failures gracefully
22. Create a function that catches JSON parsing errors when fetching data from server
23. Write an async function with nested try-catch blocks for multiple error scenarios
24. Create a function using try-catch to handle network timeout errors with retry logic
25. Write an async function that catches database connection errors and logs details

26. Create a function using try-catch to handle file reading errors and return default data
27. Write an async function that catches authentication errors and redirects to login
28. Create a function using try-catch to handle payment processing errors with rollback
29. Write an async function that catches image upload errors and shows progress status
30. Create a function using try-catch to handle multiple async operations with specific error messages

Promise.all() and Concurrent Operations

31. Write a function using Promise.all() to fetch multiple user profiles simultaneously
32. Create a function that uses Promise.all() to load all required assets before page render
33. Write a function using Promise.all() to validate multiple form fields in parallel
34. Create a function that fetches product details and reviews concurrently using Promise.all()
35. Write a function using Promise.all() to upload multiple files and track overall progress
36. Create a function that loads user data, settings, and notifications simultaneously
37. Write a function using Promise.all() to fetch data from multiple APIs and combine results
38. Create a function that validates username, email, and phone number in parallel
39. Write a function using Promise.all() to process multiple images and generate thumbnails
40. Create a function that checks multiple service endpoints health status concurrently

Promise.race() and Timeout Handling

41. Write a function using Promise.race() to implement request timeout after 5 seconds
42. Create a function that races between actual API call and cached data retrieval
43. Write a function using Promise.race() to get fastest response from multiple servers
44. Create a function that races between user input and automatic form submission timer
45. Write a function using Promise.race() to implement circuit breaker pattern for API calls
46. Create a function that races between file upload and cancel button click
47. Write a function using Promise.race() to show loading indicator if request takes too long
48. Create a function that races between geolocation and manual address input
49. Write a function using Promise.race() to implement progressive image loading
50. Create a function that races between voice recognition and text input

API Integration and Data Fetching

51. Write a function to fetch user list from API and filter users by last name
52. Create a function that fetches posts from API and sorts them by creation date
53. Write a function to get product data from API and calculate total inventory value

54. Create a function that fetches weather data and formats it for display
55. Write a function to retrieve user orders from API and group them by status
56. Create a function that fetches news articles and filters by category
57. Write a function to get customer data from API and search by email address
58. Create a function that fetches employee data and calculates average salary
59. Write a function to retrieve task list from API and filter by priority level
60. Create a function that fetches book data and sorts by author name

Sequential vs Parallel Processing

61. Write a function to process user registrations sequentially to avoid overwhelming database
62. Create a function that uploads files one by one to prevent bandwidth issues
63. Write a function to fetch dependent data where each call depends on previous result
64. Create a function that processes payments sequentially to maintain transaction order
65. Write a function to backup data files one at a time to avoid storage conflicts
66. Create a function that sends email notifications in parallel for better performance
67. Write a function to resize images simultaneously to improve processing speed
68. Create a function that validates multiple inputs in parallel for faster form submission
69. Write a function to fetch user preferences in parallel from different services
70. Create a function that downloads multiple reports simultaneously

Real-time Data and WebSocket Integration

71. Write a function to establish WebSocket connection and handle incoming messages
72. Create a function that listens for real-time price updates and updates UI
73. Write a function to send chat messages through WebSocket and await confirmation
74. Create a function that subscribes to live data feeds and processes updates
75. Write a function to handle WebSocket reconnection when connection drops
76. Create a function that streams live video data and handles buffering
77. Write a function to receive push notifications and update application state
78. Create a function that monitors server status through real-time connection
79. Write a function to handle live collaboration features like document editing
80. Create a function that processes real-time analytics data and generates reports

Advanced Promise Patterns

81. Write a function that implements retry logic with exponential backoff for failed requests

82. Create a function that implements Promise queue to limit concurrent API calls
83. Write a function that cancels pending requests when user navigates away from page
84. Create a function that implements optimistic updates with rollback capability
85. Write a function that batches multiple API calls into single request for efficiency
86. Create a function that implements request deduplication to prevent duplicate calls
87. Write a function that caches API responses with expiration time management
88. Create a function that implements progressive loading for large datasets
89. Write a function that handles offline/online scenarios with request queuing
90. Create a function that implements request prioritization based on user actions

Database Operations and CRUD

91. Write an async function to create new user record and return generated ID
92. Create a function that updates user profile and handles validation errors
93. Write a function to delete multiple records and confirm each deletion
94. Create a function that reads user data with pagination and sorting options
95. Write a function to perform transaction with multiple database operations
96. Create a function that migrates data between different database schemas
97. Write a function to backup database records to external storage
98. Create a function that synchronizes local data with remote database
99. Write a function to perform bulk insert operations with progress tracking
100. Create a function that handles database connection pooling and cleanup

Implementation Guidelines

For Each Question:

1. **Use Proper Async Syntax**: Include async/await or .then()/.catch() appropriately
2. **Handle Errors**: Always include error handling mechanisms
3. **Add Loading States**: Show progress indicators where appropriate
4. **Include Comments**: Explain the asynchronous flow
5. **Test Edge Cases**: Handle network failures, timeouts, invalid data

Example Implementation Format:

```javascript

```

```
// Question 51: Write a function to fetch user list from API and filter users by last name
async function fetchAndFilterUsersByLastName(lastName) {
  try {
    // Show loading indicator
    console.log('Loading users...');

    // Fetch users from API
    const response = await fetch('https://api.example.com/users');

    // Check if request was successful
    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`);
    }

    // Parse JSON data
    const users = await response.json();

    // Filter users by last name (case-insensitive)
    const filteredUsers = users.filter(user =>
      user.lastName.toLowerCase().includes(lastName.toLowerCase())
    );

    // Hide loading indicator
    console.log('Loading complete!');

    return filteredUsers;

  } catch (error) {
    console.error('Error fetching users:', error.message);
    throw new Error('Failed to f
```