Survery Sparrow Placement Drive

# ****SurveySparrow Placement – Fresher (FrontEnd / BackEnd / QA / Testing)****

* **Role**: Software/QA Trainee (varies by opening)
* **Company Size**: 51–200 employees
* **Experience**: Freshers off-campus or campus placements

# ****Key Skills & Topics (Based on Role Track)****

1. **Core Programming**: JavaScript, HTML, CSS (frontend) or Java/Node.js (backend)
2. **Data Structures & Algorithms**: Arrays, strings, recursion, LeetCode-level problem solving
3. **React/JS**: hooks (useEffect), state management, class vs functional components
4. **DOM Manipulation**: querySelector, getElementById, CSS block vs inline
5. **SQL/DB Design**: schema design (e.g., IRCTC booking), query optimization
6. **OOP / System Design**: for backend and full‑stack roles
7. **Soft Skills**: situational responses, conflict resolution, communication

# ****Candidate-Revealed Process (3–5 Rounds)****

* **Round 0** – Resume Screening
* **Round 1 – Written/Coding Test**
  + Frontend: HTML/CSS/JS Qs
  + Backend & general: DSA + SQL + networking
* **Round 2 – Technical Interview 1**
  + Frontend: DOM, CSS styling, JS basics, React hooks
  + Backend: LeetCode array/string problems, database schema design
* **Round 3 – Technical Interview 2 (and sometimes 3)**
  + Backend: optimization, caching, API design, SQL tweaks
  + QA/Automation: writing/presenting scripts to automate dynamic tables, pagination flows
* **Round 4 – HR / Culture Fit**
  + Situational queries (handling clients, conflicts, priorities)
  + General behavioral: “Tell me about yourself,” “Why SurveySparrow?”
  + Salary/location discussion (for experienced roles)

**Timeline**: Overall rated “moderately difficult” (avg score ~2.9/5); friendly interviewers; process typically wraps in 1–2 weeks

**Candidate Sentiment**: ~57% positive for overall experience

# Overall Analysis – Backend Developer Requirement at SurveySparrow

## ****Role Expectations (From JD & Past Questions)****

|  |  |
| --- | --- |
| Category | Details |
| Problem Solving | Strong focus on **DSA-based questions**: arrays, matrices, binary search, hashmap, etc. |
| Optimization | Solutions must be **time/space efficient**. Test cases are evaluated for performance. |
| Coding Language | Candidates used **Python**, but **Java/C++** is also fine. |
| Database | Ability to write **basic SQL queries**: joins, filters, grouping |
| Other Concepts | Core programming logic — no REST API, no frameworks required |
| Evaluation | 4 coding problems (DSA-heavy) + SQL task + mock interview-style discussion |

## Core Areas to Focus (Backend)

|  |  |  |
| --- | --- | --- |
| Area | Subtopics / Problem Themes | Source of Task |
| Arrays & Hashmaps | Job-employee mapping, frequency, prefix sum | Given in past paper |
| Matrix Traversals | Forest fire spread, time-skipping matrix problems | Given in past paper |
| Binary Search Variants | Piles of chocolate (min-max problem) | Leetcode-style |
| Dictionaries | Missing person problem – key-value + nested dictionaries | Past interview question |
| SQL Queries | Joins, Group By, Filters (not DB design) | Company expectations |
| Optimization Practice | Same problem, better code (space/time) | Mock round review |
| Test Case Thinking | Build and test 6 edge cases per problem | Leetcode pattern |

## 2-Day Backend Training Plan (14 Hours Total)

🔁 All tasks include **problem understanding**, **dry run**, **optimized code**, and **test case validation**

|  |  |  |  |
| --- | --- | --- | --- |
| Hour | Topic | LeetCode Problem (Title + Link) | Concepts Practiced |
| 1 | **Array - Prefix Sum** | 🔗 [560. Subarray Sum Equals K](https://leetcode.com/problems/subarray-sum-equals-k) | Prefix sum, HashMap |
| 2 | **Array - Frequency** | 🔗 [287. Find the Duplicate Number](https://leetcode.com/problems/find-the-duplicate-number) | Cycle detection, Binary search |
| 3 | **Matrix - BFS** | 🔗 [994. Rotting Oranges](https://leetcode.com/problems/rotting-oranges) | BFS in matrix, direction vectors |
| 4 | **Matrix - Distance** | 🔗 [542. 01 Matrix](https://leetcode.com/problems/01-matrix) | BFS for shortest distance |
| 5 | **Matrix - Pattern Logic** | 🔗 [73. Set Matrix Zeroes](https://leetcode.com/problems/set-matrix-zeroes) | In-place grid updates |
| 6 | **Binary Search - Min Time** | 🔗 [875. Koko Eating Bananas](https://leetcode.com/problems/koko-eating-bananas) | BS on answer space |
| 7 | **Binary Search - Split Array** | 🔗 [410. Split Array Largest Sum](https://leetcode.com/problems/split-array-largest-sum) | BS + prefix sum + greedy |
| 8 | **HashMap - Basics** | 🔗 [1. Two Sum](https://leetcode.com/problems/two-sum) | HashMap for pair sum |
| 9 | **HashMap - Longest Sequence** | 🔗 [128. Longest Consecutive Sequence](https://leetcode.com/problems/longest-consecutive-sequence) | Set-based range detection |
| 10 | **String - Unique Chars** | 🔗 [3. Longest Substring Without Repeating Characters](https://leetcode.com/problems/longest-substring-without-repeating-characters) | Sliding window + HashSet |
| 11 | **String - Pattern Build** | 🔗 [13. Roman to Integer](https://leetcode.com/problems/roman-to-integer) | Parsing + HashMap |
| 12 | **String - Reverse & Word Manipulation** | 🔗 [151. Reverse Words in a String](https://leetcode.com/problems/reverse-words-in-a-string) | Word boundaries, trimming, splitting |
| 13 | **Heap/Greedy** | 🔗 [**57. Insert Interval**](https://leetcode.com/problems/insert-interval/) | Interval merging, min-heap |
| 14 | **Bonus Practice / SQL / Speed Round** | ✅ Speed run: |  |

## 10 MySQL Tasks

All tasks include creating schema + writing one or more queries

### ****TASK 1: Employee Work Hours****

**Schema**

Employee(id, name)

WorkLog(emp\_id, date, hours\_worked)

**Query Task**  
Write a query to find total hours worked by each employee for the current month.

**Concepts:** JOIN, SUM, GROUP BY, DATE filter

### ****TASK 2: Highest Salary Per Department****

**Schema**

Employee(id, name, salary, dept\_id)

Department(id, dept\_name)

**Query Task**  
List the employee(s) with the highest salary in each department.

**Concepts:** JOIN, GROUP BY, MAX + subquery or CTE

### ****TASK 3: Attendance Report****

**Schema**

Attendance(emp\_id, date, status) -- status: 'Present' or 'Absent'

**Query Task**  
Find employees with more than 3 absences in the last 30 days.

**Concepts:** COUNT, GROUP BY, HAVING, DATE functions

### ****TASK 4: Average Salary Filter****

**Schema**

Employee(id, name, salary)

**Query Task**  
List employees whose salary is above the average salary.

**Concepts:** Subquery, AVG

### ****TASK 5: Most Ordered Products****

**Schema**

Orders(id, product\_id, quantity)

Product(id, name)

**Query Task**  
Find top 3 most ordered products by quantity.

**Concepts:** GROUP BY, ORDER BY, LIMIT

### ****TASK 6: Department Without Employees****

**Schema**

Department(id, dept\_name)

Employee(id, name, dept\_id)

**Query Task**  
Find all departments that currently have no employees.

**Concepts:** LEFT JOIN, NULL filter

### ****TASK 7: Duplicate Records Detection****

**Schema**

Feedback(id, user\_id, message)

**Query Task**  
Find users who submitted the **same message more than once**.

**Concepts:** GROUP BY, HAVING COUNT > 1

### ****TASK 8: Product Price Ranges****

**Schema**

Product(id, name, price)

**Query Task**  
Classify products into price bands:

* Low (< 100)
* Medium (100–500)
* High (> 500)

**Concepts:** CASE WHEN, aliasing

### ****TASK 9: Active Users with Latest Login****

**Schema**

Login(user\_id, login\_time)

**Query Task**  
Find each user’s **most recent login time**.

**Concepts:** MAX, GROUP BY

### ****TASK 10: Monthly Revenue****

**Schema**

Sales(id, date, amount)

**Query Task**  
Calculate total revenue for each month in 2024.

**Concepts:** DATE\_FORMAT, GROUP BY

# ****Overall Analysis – Frontend Developer Requirement at SurveySparrow****

## ****Role Expectations**** (From JD + Past Drive)

|  |  |
| --- | --- |
| Category | Details |
| **Core Skills** | HTML, CSS, JavaScript (Vanilla), and basics of React (optional but preferred) |
| **Design Ability** | Able to **replicate static UI from screenshots** and build **portfolio websites** |
| **Problem Solving** | Solve **logic-based JavaScript problems** with **test cases** (similar to coding rounds) |
| **Frontend Thinking** | Good understanding of **UI/UX**, responsive layouts, clean structure |
| **Tools** | Familiarity with GitHub, ability to push projects |
| **Evaluation** | Website replication + 1–2 logic-based problems + portfolio presentation |

## ****Core Areas to Focus (Frontend)****

|  |  |  |
| --- | --- | --- |
| Area | Subtopics / Problem Themes | Source of Task |
| **HTML + CSS** | Semantic tags, layouts, forms, Flexbox, media queries | UI replication, Portfolio, FAQs |
| **JavaScript Logic** | Arrow direction logic, DOM manipulation (toggle/show/hide) | Past drive (Arrows problem) |
| **DOM Manipulation** | Toggle text, toggle class, update style/attributes dynamically | Student-interactive projects |
| **GitHub Deployment** | GitHub repo creation, commits, and Netlify deployment | Mentioned as deliverables |
| **JS Test Case Thinking** | Handle 6 test cases in code (problem-solving mindset) | Direct from previous round |
| **React Basics (Day 2)** | useState, useEffect, list/conditional rendering, props | Preferred skill (React familiarity) |
| **Portfolio Building** | Personal site with About, Skills, Projects, Contact sections | JD + drive expectations |

## Summary of What to Train

|  |  |  |
| --- | --- | --- |
| Training Day | Focus | What to Cover |
| **Day 1** | HTML + CSS + JavaScript | UI replication, DOM tasks, interactivity, 1 logic problem |
| **Day 2** | React Fundamentals | Components, state, effect, API calls, portfolio in React |

## Day FrontEnd Training Program

### Day 1

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Task Title | Description | Skills Covered |
| 1 | **Build Your Resume (HTML only)** | Create a simple resume webpage with sections like About, Education, Skills, and Contact. | HTML tags, semantic structure |
| 2 | **Calculator (2-operand)** | Create a calculator that performs +, −, ×, ÷ between two numbers. | Input, basic math, conditionals |
| 3 | **Click Counter** | Create a button that increases a counter value on each click and displays the count. | JS event listeners, variables |
| 4 | **Click Counter Button** | A button that increases a number every time you click it. | Button event listener, innerText manipulation |
| 5 | **Color Picker** | A dropdown with color names. When a user selects a color, the page background changes to that color. | select tag, onchange event, DOM style update |
| 6 | **Colorful Button Page** | Create 4–5 different styled buttons with hover effects and transitions. | CSS styling, hover, transitions |
| 7 | **Dark Mode Toggle** | Add a toggle button to switch the page between light and dark mode. | JS DOM classList, styling changes |
| 8 | **Digital Clock** | Show current time on the screen and update it every second. | setInterval, Date object, JS formatting |
| 9 | **Image Gallery Grid** | Build a 2x3 image gallery that adjusts on smaller screens. | CSS Grid/Flexbox, media queries |
| 10 | **Interactive List** | User types a task, presses "Add", and it appears below in a list. | JS input, createElement, appendChild |
| 11 | **Navigation Bar with Links** | Build a horizontal navigation bar with working (dummy) links. | HTML <nav>, inline-block/flexbox layout |
| 12 | **Personal Bio Page** | Create a webpage with your name, picture, and a paragraph about yourself. Use basic layout and tags like <header>, <img>, <section>. | HTML tags, semantic layout |
| 13 | **Personal Portfolio Skeleton** | Create a multi-section page (Home, About, Projects) using navigation links. | HTML anchors, layout, basic styling |
| 14 | **Show/Hide Paragraph** | A paragraph and a button — clicking the button hides/shows the paragraph. | Toggle logic, JS events, DOM manipulation |
| 15 | **Show/Hide Password Field** | Add a checkbox to toggle password visibility in a login form. | JS DOM events, input type toggle |
| 16 | **Simple Contact Form with validation** | Create a contact form with Name, Email, and Message fields, and a Submit button. | Form elements, labels, input types |
| 17 | **Toggle Text on Click** | A button labeled "Show More" → when clicked, toggles to "Show Less" and vice versa. | innerText toggle |
| 18 | **Toggle Paragraph HTML** | A div contains a quote. A button toggles it between two different quote versions using innerHTML. | innerHTML manipulation |
| 19 | **Toggle Theme (Light/Dark)** | A button switches the theme by toggling a dark-mode class on the <body>. | classList.toggle() |
| 20 | **To-Do List** | Input + Add button. Add items to a list with a delete button for each. | DOM element creation, deletion, events |
| 21 | **Change Text Color** | 3 buttons (Red, Green, Blue). Clicking one changes the color of a heading text. | Inline style.color change |
| 22 | **Toggle Image Source** | One button switches between two images (e.g., day/night). | Change src attribute dynamically |
| 23 | **Enable/Disable Input** | A checkbox enables or disables a text input field. | setAttribute, removeAttribute |
| 24 | **Show/Hide Password** | A password field + checkbox to show/hide password using type toggle. | Toggle type="password" ↔ text |
| 25 | **Live Character Counter** | Textarea shows live character count below it. | input event + character counting |
| 26 | **Highlight on Hover (Bonus)** | Hovering over a paragraph highlights it (add/remove class). | mouseover, mouseout events, classes |

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Task Title | Description | Skills Covered |
| 1 | **Accordion FAQ Panel** | Click to expand/collapse answers, only one open at a time | DOM manipulation, forEach, classList.toggle |
| 2 | **Image Carousel** | Show 3–4 images with next/prev buttons and auto-slide | JS timers, array indexing, setInterval, transitions |
| 3 | **Responsive Pricing Cards** | Build 3-column pricing layout with hover effects, responsive on mobile | CSS Grid/Flexbox, transitions, media queries |
| 4 | **Tabs Interface** | Show different content for each tab; active tab highlights | DOM querySelector, event delegation |
| 5 | **Expandable Cards** | Cards expand smoothly on click to show more content | max-height, transition, classList.toggle |
| 6 | **Custom Dropdown Menu** | Build a stylized dropdown without <select> tag | Advanced CSS, JS toggle, keyboard accessibility |
| 7 | **Scroll to Top Button** | A floating button appears after scrolling 300px | scrollY, scrollTo(), window event listeners |
| 8 | **Theme Toggle (Multi-theme)** | Toggle between light, dark, and sepia themes | classList, stateful JS, CSS variables |
| 9 | **Search Filter (Live)** | Filter a list of items as the user types | input events, filter(), includes() |
| 10 | **Animated Skill Bars** | Animate skill progress bars on scroll | IntersectionObserver, CSS keyframes |
| 11 | **Sticky Navigation** | Nav bar sticks to top after scrolling past header | position: sticky, JS scroll event |
| 12 | **Countdown Timer** | Countdown to a future date (e.g. event, deadline) | Date, setInterval, time calculations |
| 13 | **Floating Label Form** | Form fields where labels float on focus/input | Advanced CSS selectors, transitions |
| 14 | **Draggable List** | Drag-and-drop items to reorder a list | dragstart, dragover, drop events |
| 15 | **Animated Hamburger Menu** | Mobile nav toggle with animated icon | CSS transitions, JS toggle, responsive layout |
| 16 | **Keyboard Event App** | Show which key was pressed and its code in real-time | keydown, event.key, UI display |
| 17 | **Pomodoro Timer** | Build a 25-min work/5-min break timer | setInterval, UI state management |
| 18 | **LocalStorage Todo List** | Save todos to localStorage and load on refresh | localStorage, JSON, event persistence |
| 19 | **Form Step Wizard** | Multi-step form with “Next”/“Back” buttons and progress | DOM control, conditional rendering |
| 20 | **Modal with Escape Close** | Modal closes on button click or Esc key | keydown, overlay handling, accessibility |
| 21 | **Responsive Card Flip** | Flip card on hover to show back (CSS + JS version) | 3D transforms, perspective, event triggers |
| 22 | **Weather Widget (Static API)** | Use dummy JSON to show weather card info | JSON, template literals, card layout |
| 23 | **Email Suggestion Box** | Show dropdown with email domain suggestions as you type | input, keyup, array filter |
| 24 | **Infinite Scroll List** | Load more items as user scrolls down | scrollTop, offsetHeight, fetch() mock |
| 25 | **Real-Time Clock with Greeting** | Show greeting based on time of day + live clock | Date, conditional logic |
| 26 | **Responsive Grid Layout** | Build Pinterest-style layout with grid/flex and media queries | CSS Grid, responsive design |
| 27 | **Password Strength Checker** | Show password strength as user types | Regex, DOM feedback, conditional styles |
| 28 | **Notification Toast** | Show auto-dismissing toast alert on action | DOM injection, timers, styling |
| 29 | **User Profile Card Generator** | Use JS objects to render multiple user cards | Object iteration, template literals |
| 30 | **Scroll Reveal Animation** | Elements fade in as you scroll | IntersectionObserver, transitions, animation delay |

### Day 2:

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Task Title | Description | Concepts Practiced |
| 1 | Counter App | Create a simple counter with + and – buttons. | useState, event handling |
| 2 | Show/Hide Text | A toggle button to show/hide a quote paragraph. | useState, conditional rendering |
| 3 | Color Changer | A list of buttons (Red, Blue, Green) to change background color. | useState, inline styles |
| 4 | To-Do List | Add/delete tasks from a to-do list. Use array in state. | useState, list rendering, array update |
| 5 | Form Input Tracker | Show live preview of text as user types in input field. | Controlled components, useState |
| 6 | Digital Clock | Show live clock using Date. Update every second. | useEffect, setInterval, cleanup |
| 7 | Fetch Users from API | Fetch and display users from https://jsonplaceholder.typicode.com/users. | useEffect, fetch, API handling |
| 8 | Conditional Message App | Display “Welcome” if logged in; “Please login” if not. Add a login toggle button. | Conditional rendering, useState |
| 9 | User Search (Filter) | From fetched users, implement a search bar to filter by name. | API, filtering, input state, list rendering |
| 10 | React Portfolio (Mini Project) | Build a 2-page portfolio using React components (Home + About + Projects). | Routing (optional), props, structure |
| 11 | Temperature Converter | Convert Celsius to Fahrenheit and vice versa using inputs. | useState, input handling |
| 12 | Tab Switcher | Create 3 tabs ("About", "Skills", "Contact") — clicking a tab shows related content. | Conditional rendering, useState |
| 13 | Like Button Component | A post card with a like button. Count increases on each click, turns red if liked. | Props, useState, conditional class |
| 14 | Accordion Component | Create collapsible panels for FAQs. Only one panel open at a time. | List mapping, conditional rendering |
| 15 | Simple Quiz App | Show one question at a time with options. On select, go to next. Show score at end. | Props, state flow, conditional UI |
| 16 | Theme Switcher App | Toggle between Light/Dark mode across the entire app. | useState, global style, class toggling |
| 17 | Stopwatch | Start, pause, and reset a timer using buttons. | useEffect, setInterval, cleanup |
| 18 | Fake Blog Posts Feed | Fetch blog post titles from jsonplaceholder and render them as cards. | API call, list rendering, card layout |
| 19 | Weather Search | User types a city, fetch weather from OpenWeatherMap API (use fake key). | Controlled input, API call, conditional UI |
| 20 | Props Practice Portfolio | Parent component passes data (name, skills, links) to 2–3 child components. | Props, component reuse, structure |

# **Suggested Resources**

* **HTML/CSS/JS**: MDN + GFG interviews + GFG GFG GFG
* **React**: Official docs + small Mini-apps
* **DSA**: LeetCode (Easy), GeeksForGeeks
* **SQL/DB**: LeetCode, AmbitionBox questions for DB design
* **Frontend roles**: Emphasize clean CSS (inline vs block, centering), DOM selection, React fundamentals
* **Backend roles**: Sharpen logic & SQL; also practice schema design questions
* **Technical rounds**: Interviewers are deep-dive capable—be ready to explain code logic or design choices
* **Behavioral**: Prepare scenario-based responses (“client handling”, “time pressure”)—common in situational round

# Final Prep Checklist

* Master HTML/CSS fundamentals + centering/layout
* JS & React: hooks, JSX, DOM operations
* Solve 20 LeetCode easy problems (arrays/strings)
* Design SQL schemas; practice joins/subqueries
* Mockify interview environment + practice behavioral answers