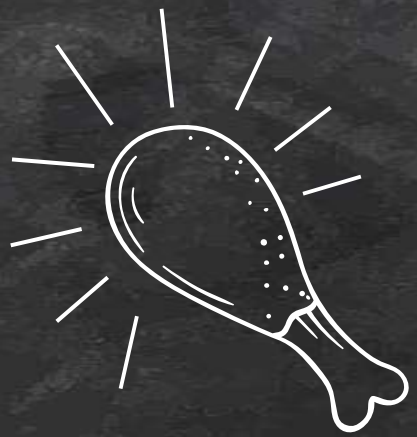
RESTAURANT MANAGEMENT SYSTEM



Debolina Saha – RA2311026010909

INTRODUCTION

What is a Restaurant Management System?

- A system that helps users find restaurants based on budget, cuisine, and rating.
- Enhances user experience by providing quick and accurate results.

Purpose

- Enables users to search and filter restaurants effectively.
- Helps in organizing and managing restaurant data efficiently.

Why Sorting & Searching?

- Organizes restaurants efficiently.
- Enables quick searches based on user preferences.
- Reduces the time required to find the best dining options.

FEATURES IMPLEMENTED

BUBBLE SORT

01

Sorts restaurants by price.

BINARY SEARCH

02

Quickly finds restaurants within a specified price.

FILTERING OPTIONS

CATEGORY

01

Veg / Non-Veg.

CUISINE TYPE

02

Indian, Italian, Chinese, BBQ, etc.

RATING SYSTEM

03

Matches ±0.2 of user input for better results.

PRICE RANGE

04

E.g., 1000-1500, 1000-2000, etc.

ALGORITHM EXPLANATION

BUBBLE SORT

How Bubble Sort Works

1. Compares adjacent elements and swaps them if needed.
2. Repeats the process until the list is sorted.

Time Complexity

1. Worst & Average Case: $O(n^2)$
2. Best Case: $O(n)$ (when already sorted)

Why Use Bubble Sort?

1. Simple to implement.
2. Works well for small datasets.
3. Ensures stable sorting by maintaining the relative order.

BINARY SEARCH

How Binary Search Works

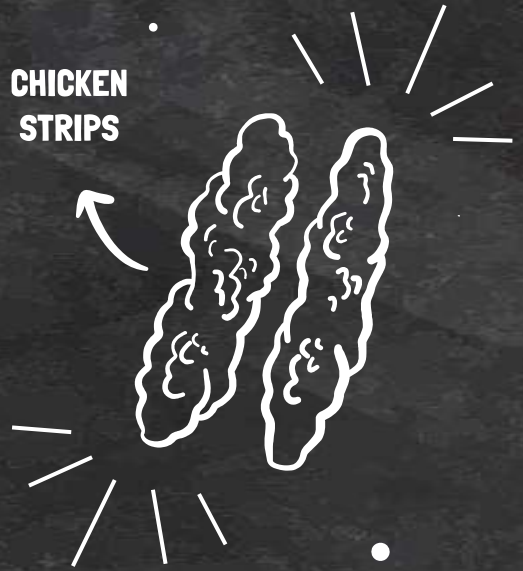
1. Requires a sorted list.
2. Divides the list into two halves.
3. Checks if the middle element matches the search value.
4. Continues searching in the relevant half recursively or iteratively.

Time Complexity

1. Best Case: O(1)
2. Worst & Average Case: O(log n)

Why Use Binary Search?

1. Faster than linear search; especially for large datasets.
2. Improves efficiency in searching for specific price ranges.



$5.7 Trillion

The global restaurant industry is projected to reach by 2027.

REAL-WORLD APPLICATIONS

01 **RESTAURANT AGGREGATOR**
Used in apps like Zomato, Swiggy, and Uber Eats.

02 **BUDGET-FRIENDLY SEARCHES**
Helps users find restaurants within their price range.

03 **POS (POINT OF SALE)**
Used in restaurant management and hotel industries.

04 **FOOD DELIVERY APPS**
Efficient data sorting and retrieval for better user experience.

ADVANTAGES & LIMITATIONS

ADVANTAGES

- Faster searches using Binary Search.
- Simple sorting method with Bubble Sort.
- Customizable filters enhance user experience.

LIMITATIONS

- Bubble Sort is inefficient for large datasets.
- Binary Search requires pre-sorted data.
- Less efficient than database-driven applications.

KNOW THE MARKET

$5.7 Trillion

The global restaurant market

$1 Trillion

The online food delivery industry

60%

Over 60% of global restaurant sales

80%

Restaurants uses tech-driven management systems

TIME COMPLEXITY FOR MY CODE

Sorting (Bubble Sort):
Binary Search:
n-1
Σi= n(n - 1) = O(n2)
2
i=1
log2 n =
O(log n)
Filtering (Linear Scan):
Total Complexity:
O(n)
O(n2) + O(log n) + O(n) = O(n2)
Optimized Complexity (Using MergeSort/QuickSort):
O(n log n) + O(log n) + O(n) = O(n log n)

CODE FOR THE RESTAURANT MANAGEMENT SYSTEM

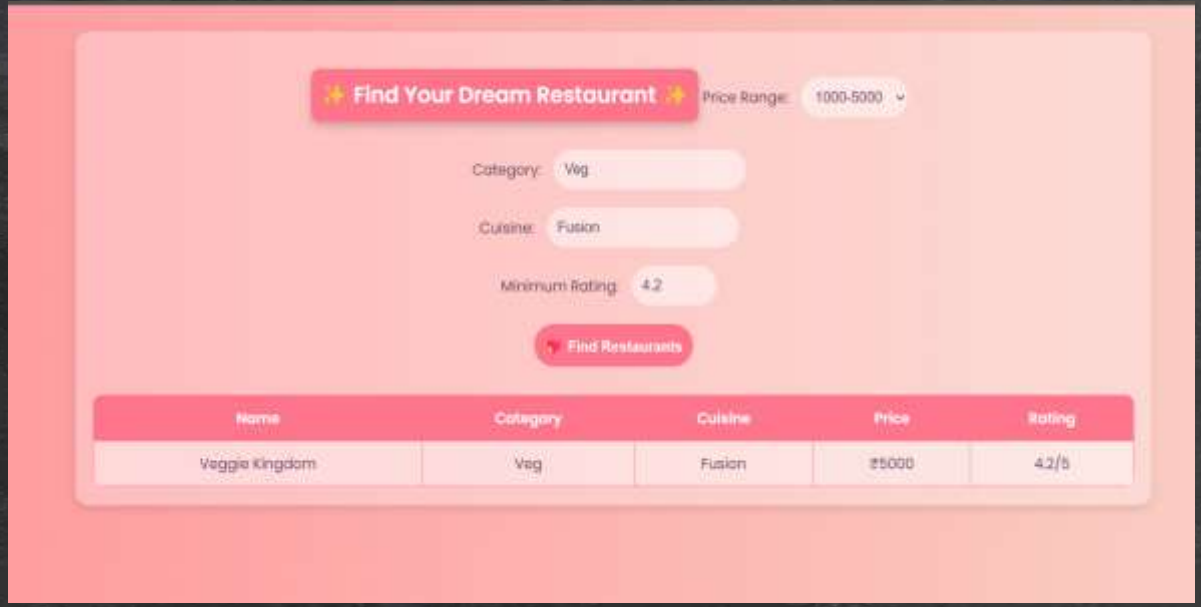
main.c

Share Run Output Clear

```c
#include <stdio.h>
#include <string.h>

typedef struct {
    char name[50];
    int price;
    char category[10];  // "Veg" or "Non-Veg"
    char cuisine[20];   // Cuisine type
    float rating;       // Rating (1.0 - 5.0)
} Restaurant;

// Function to sort restaurants based on price using Bubble Sort
void sortRestaurants(Restaurant arr[], int n) {
    int i, j;
    Restaurant temp;
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (arr[j].price > arr[j + 1].price) {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}
```

```
Select a price range:
1. 1000-1500
2. 1000-2000
3. 1000-2500
4. 1000-3000
5. 1000-4000
6. 1000-5000
7. 1000-6000
8. 1000-7000
Enter your choice (1-8): 6
Enter category (Veg / Non-Veg / Any): Veg
Enter cuisine type (Any / Indian / Italian / Chinese / Japanese / BBQ /
    Mediterranean / Thai / Vegan / Mexican / Seafood / French / Continental
    / South Indian / North Indian / Fusion): Fusion
Enter minimum rating (e.g., 4.0 or '0' to skip): 4.2

Matching Restaurants:
Veggie Kingdom (Veg, Fusion) - ₹5000 | Rating: 4.2/5


=== Code Execution Successful ===
```

FRONTEND OF THE PROJECT



BACKEND OF THE PROJECT

```
<script>
    const restaurants = [

{ name: "Tandoori Flames", price: 1000, category: "Non-Veg", cuisine: "Indian", rating: 4.5 },
{ name: "Herb & Spice", price: 1200, category: "Veg", cuisine: "Italian", rating: 4.3 },
{ name: "The Royal Feast", price: 1500, category: "Non-Veg", cuisine: "Continental", rating: 4.8 },
{ name: "Sushi Heaven", price: 2000, category: "Non-Veg", cuisine: "Japanese", rating: 4.6 },
{ name: "Green Leaf", price: 2500, category: "Veg", cuisine: "Organic", rating: 4.0 },
{ name: "Meat Lovers Hub", price: 3000, category: "Non-Veg", cuisine: "BBQ", rating: 4.9 },
{ name: "Golden Chopsticks", price: 3500, category: "Non-Veg", cuisine: "Chinese", rating: 4.7 },
{ name: "Paneer Palace", price: 4000, category: "Veg", cuisine: "North Indian", rating: 4.4 },
{ name: "Ocean's Delight", price: 4500, category: "Non-Veg", cuisine: "Seafood", rating: 4.6 },
{ name: "Veggie Kingdom", price: 5000, category: "Veg", cuisine: "Fusion", rating: 4.2 },
{ name: "Spice Symphony", price: 2200, category: "Veg", cuisine: "Thai", rating: 4.3 },
{ name: "La Fiesta", price: 3200, category: "Non-Veg", cuisine: "Mexican", rating: 4.6 },
{ name: "Curry House", price: 2700, category: "Veg", cuisine: "Indian", rating: 4.5 },
{ name: "Sea Breeze", price: 4800, category: "Non-Veg", cuisine: "Seafood", rating: 4.7 },
{ name: "Sunset Grill", price: 5500, category: "Non-Veg", cuisine: "Steakhouse", rating: 4.8 },
{ name: "Royal Vegan Bistro", price: 6000, category: "Veg", cuisine: "Vegan", rating: 4.1 },
{ name: "Fine Dining Deluxe", price: 6500, category: "Non-Veg", cuisine: "French", rating: 4.9 },
{ name: "Exotic Bites", price: 7000, category: "Veg", cuisine: "Mediterranean", rating: 4.5 },
{ name: "Bistro Delight", price: 1800, category: "Veg", cuisine: "European", rating: 4.2 },
{ name: "Flavors of Punjab", price: 2800, category: "Non-Veg", cuisine: "Punjabi", rating: 4.7 },
{ name: "Tokyo Bites", price: 3800, category: "Non-Veg", cuisine: "Japanese", rating: 4.6 },
{ name: "The Breakfast Spot", price: 900, category: "Veg", cuisine: "American", rating: 4.3 },
{ name: "Himalayan Bliss", price: 2300, category: "Veg", cuisine: "Nepalese", rating: 4.4 },
{ name: "The Kebab Factory", price: 2600, category: "Non-Veg", cuisine: "Middle Eastern", rating: 4.8 },
{ name: "Greek Tavern", price: 3400, category: "Veg", cuisine: "Greek", rating: 4.5 },
{ name: "Southern Comfort", price: 2900, category: "Non-Veg", cuisine: "South Indian", rating: 4.6 },
{ name: "Turkish Delight", price: 3700, category: "Non-Veg", cuisine: "Turkish", rating: 4.7 },
{ name: "Vegan Bliss", price: 4100, category: "Veg", cuisine: "Vegan", rating: 4.2 },
{ name: "Sizzling Szechuan", price: 4300, category: "Non-Veg", cuisine: "Chinese", rating: 4.5 }
```

```
            transition: 0.3s;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>✨ Find Your Dream Restaurant ✨</h2>
        <label>Price Range:</label>
        <select id="priceRange">
            <option value="1000-1500">1000-1500</option>
            <option value="1000-2000">1000-2000</option>
            <option value="1000-2500">1000-2500</option>
            <option value="1000-3000">1000-3000</option>
            <option value="1000-4000">1000-4000</option>
            <option value="1000-5000">1000-5000</option>
            <option value="1000-6000">1000-6000</option>
            <option value="1000-7000">1000-7000</option>
        </select>
        <br>
        <label>Category:</label>
        <input type="text" id="category" placeholder="Veg / Non-Veg / Any">
        <br>
        <label>Cuisine:</label>
        <input type="text" id="cuisine" placeholder="Indian, Chinese, etc.">
        <br>
        <label>Minimum Rating:</label>
        <input type="number" id="rating" step="0.1" min="0" max="5" placeholder="4.0 or 0 to skip">
        <br>
        <button onclick="filterRestaurants()">🍽️ Find Restaurants</button>

        <table>
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Category</th>
```

```
<script>
    function filterRestaurants() {
        let maxPrice = parseInt(priceRange[1]);
        let category = document.getElementById("category").value.toLowerCase();
        let cuisine = document.getElementById("cuisine").value.toLowerCase();
        let minRating = parseFloat(document.getElementById("rating").value) || 0;

        let filtered = restaurants.filter(r =>
            r.price >= minPrice && r.price <= maxPrice &&
            (category === "any" || r.category.toLowerCase() === category) &&
            (cuisine === "any" || r.cuisine.toLowerCase().includes(cuisine)) &&
            r.rating >= minRating
        );

        displayRestaurants(filtered);
    }

    function displayRestaurants(restaurants) {
        let table = document.getElementById("restaurantTable");
        table.innerHTML = "";

        if (restaurants.length === 0) {
            table.innerHTML = "<tr><td colspan='5'>No restaurants found.</td></tr>";
            return;
        }

        restaurants.forEach(r => {
            let row = `<tr>
                <td>${r.name}</td>
                <td>${r.category}</td>
                <td>${r.cuisine}</td>
                <td>₹${r.price}</td>
                <td>${r.rating}/5</td>
            </tr>`;
```

THANKS!