horizontal line

**Nhlakanipho Mathibela**

Student No : ST10460431

APPR7112 POE

# Implementation of service request feature

A binary tree has been utilized as a fundamental data structure in the creation of the service request functionality. This structure enables effective management and organization of service issues. This document details the rationale behind design decisions, the capabilities, and the benefits of implementing a binary tree for this purpose.

## **Rationale for Employing a Binary Tree**

The main reason for incorporating a binary tree structure in the service request functionality is to enhance the efficiency of handling, accessing, and modifying service request data. The binary tree organization enables:

### **Structured Data Management**

Service requests are organized in a tree-like structure, facilitating the management of request relationships.

**Streamlined Retrieval**

Rapid location of particular service requests using multiple search parameters. **Real-time Modifications**

Smooth addition and removal of service requests as their status evolves.

## Design of the Binary Tree

### **Structure**

The binary tree is designed as a binary search tree (BST), where each node represents a service request. Each node contains the following attributes:

**Issue ID:**  A unique identifier for the service request.

**Location:** The location related to the service request.

**Category:** The category of the service request (e.g., bug, feature request).

**Description:** A brief description of the issue.

**Image Path:** A path to any associated images or files.

**Status:** The current status of the service request (e.g., New, Open, Resolved).

### **Operation:** The binary tree supports several key operations:

**Insertion:** New service requests are inserted into the tree based on their unique identifiers, ensuring that the tree remains organized and searchable.

**Deletion:** When a service request is resolved or closed, it can be removed from the tree, maintaining the integrity of the data structure.

**Searching:** Users can quickly find service requests by traversing the tree based on specific criteria, such as issue ID or status.

**Traversal:** The tree can be traversed using in-order, pre-order, or post-order methods to generate reports or display requests in a user-friendly manner.

## Advantages of Implementing a Binary Tree Structure

### 1. Performance

Operational Efficiency: Search, insert, and delete functions in a binary search tree have an average-case time complexity of O(log n), ensuring swift access to service requests as the dataset expands.

Flexible Memory Allocation: Binary trees utilize memory dynamically, offering superior efficiency compared to static data structures such as arrays.

### 2. Adaptability

Accommodating Expansion: The binary tree structure can effectively manage an increasing number of service requests while maintaining optimal performance.

Self-Adjusting Variants: When implemented as a self-balancing tree (e.g., AVL or Red-Black tree), the structure can automatically adjust to preserve efficiency during frequent updates.

### 3. Advanced Query Capabilities

Refined Search and Arrangement: The binary tree structure enables efficient filtering and sorting of service requests by status, priority, or other attributes, allowing users to concentrate on relevant inquiries.

Consolidated Reporting: The tree structure facilitates data aggregation for reporting purposes, such as tallying the number of requests in each status or category.

# Project Completion

## Project Overview

The Service Request Management System is engineered to enable users to submit complaints across various domains, including road issues, waste disposal, and utility services. This platform allows individuals to file service requests, examine existing ones, and monitor their progress. The project employs a binary tree structure for efficient request management, ensuring swift access and organized data handling.

## Principal Learning Outcomes Comprehension of Data Structures:

Acquired comprehensive knowledge of binary trees, their characteristics, and their use in managing hierarchical information. Mastered the implementation of binary search trees (BSTs) for efficient searching, adding, and removing operations. Database Interaction: Cultivated expertise in MySQL database communication using C# and the MySqlConnector library. Mastered the execution of CRUD (Create, Read, Update, Delete) operations.

## User Interface Creation:

Improved proficiency in crafting user interfaces with Windows Forms, emphasizing usability and user experience. Incorporated various interface elements such as text fields, buttons, and list views to enhance user interaction. Obstacles Encountered Data

## Structure Implementation:

Issue:

Constructing the binary tree structure and maintaining its balance during additions and removals.

### Resolution:

Explored self-balancing binary trees (e.g., AVL trees) and implemented a basic binary search tree for effective service request management.

## Database Connectivity:

### Issue:

Establishing a stable connection to the MySQL database and managing exceptions during database operations.

### Resolution:

Incorporated error handling and connection management strategies to ensure robust database interactions.

# Coding Methodologies

## Object-Oriented Programming (OOP):

Employed OOP concepts to construct classes and models for service requests, events, and notifications. Wrapped data and functions within classes to enhance code reusability and ease of maintenance.

## Event-Driven Programming:

Adopted event-driven programming strategies to manage user interactions and react to occurrences like mouse clicks and form submissions. Developed event handlers to control application flow based on user actions.

## Data Binding:

Leveraged data binding methods to link UI components with data sources, enabling dynamic updates and enhancing user experience. Incorporated list views and data grids for efficient display of service requests and events.

# Newly Acquired Skills

## C# Programming:

Enhanced expertise in C# programming, with emphasis on advanced topics such as LINQ, collections, and error handling.

## Database Administration:

Obtained knowledge in database design and administration, including schema development, data normalization, and query enhancement.

## UI/UX Design:

Cultivated skills in user interface design principles, emphasizing the creation of intuitive and user-friendly applications.

# Tech Recomendations

## ASP.NET Core:

For expanding the application to include a web-based interface, ASP.NET Core is an excellent choice. It enables the creation of web APIs or comprehensive web applications, allowing users to access the system from any internet-connected device. This enhancement improves both user experience and accessibility. ASP.NET Core's reputation for high performance and scalability makes it well-suited for contemporary web applications.

## SignalR: As a library for ASP.NET

SignalR introduces real-time web capabilities to applications. This technology is particularly useful for implementing features such as instant updates for announcements or events, eliminating the need for page refreshes. By providing immediate information, SignalR boosts user engagement.

## Docker:

Utilizing Docker for application containerization simplifies deployment and dependency management. This approach enhances the application's ability to scale and its portability, ensuring consistent performance across various environments (development, testing, production). This is particularly advantageous for applications that may require deployment in different cloud settings.

## Frontend Frameworks (e.g., React, Angular, or Blazor):

When expanding the application to include a more dynamic user interface, employing a modern frontend framework can significantly enhance the user experience. React or Angular can be used to develop responsive, single-page applications that improve interactivity. Alternatively, Blazor, a .NET-based framework, allows for the creation of interactive web UIs using C# instead of JavaScript, potentially offering better compatibility with the existing .NET stack.