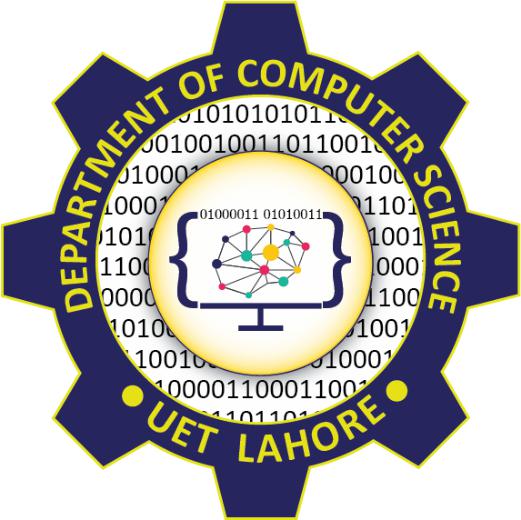
**DSA FINAL PROJECT REPORT**



**Session 2023 - 2027**

**Submitted by:**  
Arwa Mahwash 2023-CS-16

**Submitted To:**

Sir Nazeef-Ul-Haq

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

Table of Contents

[1.Executive Summary 3](#_Toc185209018)

[2.Introduction 4](#_Toc185209019)

[2.1 Background 4](#_Toc185209020)

[2.2 Objectives 4](#_Toc185209021)

[2.3 Project Scope 4](#_Toc185209022)

[2.4 Modifications Implemented 5](#_Toc185209023)

[3. Implementation Details 5](#_Toc185209024)

[3.1 Technologies and Frameworks Used 5](#_Toc185209025)

[3.2 Project Structure and Forms 5](#_Toc185209026)

[3.2.1. Tree View Form 5](#_Toc185209027)

[3.2.2. Add Employee Form 5](#_Toc185209028)

[3.2.3. Remove/Edit Employee Form 6](#_Toc185209029)

[3.2.4. Transfer Employee Form 6](#_Toc185209030)

[3.2.5. Search Form 6](#_Toc185209031)

[3.2.6. List Subordinates Form 6](#_Toc185209032)

[3.2.7. Reporting Chain Form 6](#_Toc185209033)

[3.3 Classes Used 6](#_Toc185209034)

[3.3.1 Tree\_node Class 7](#_Toc185209035)

[3.3.2 OrgTree Class 7](#_Toc185209036)

[3.3.3EmpQueue<T> Class 7](#_Toc185209037)

# 1.Executive Summary

This report presents the design, development, and implementation of an **Organizational Hierarchy Management System**. The application dynamically manages the hierarchical structure of an organization using a tree-based model. It features seven key forms that enable functionalities like adding employees, removing/editing employees, transferring employees, listing subordinates, and searching for employees using BFS or DFS algorithms. Developed in **C#** using **Windows Forms**, the project provides a user-friendly interface for managing organizational hierarchies efficiently. The report details the technologies used, the implementation process, test cases, wireframes, and future advancements.

**Organizational Hierarchy Management System**

# **2.Introduction**

## 2.1 Background

In large organizations, managing hierarchical relationships such as employee roles, reporting chains, and transfers can become complex. A structured tool is needed to simplify operations like employee addition, removal, or searching while maintaining the organizational structure.

## 2.2 Objectives

The main objectives of this project are:

* To provide a dynamic **Tree View** of the organizational structure.
* To allow easy addition, removal, and transfer of employees.
* To search for employees efficiently using BFS and DFS algorithms.
* To display subordinates and reporting chains for employees.
* To implement a user-friendly and interactive **Graphical User Interface (GUI)**.

## 2.3 Project Scope

The scope of this project is:

* Building a Windows-based desktop application for managing organizational hierarchies.
* Implementing tree-based operations for dynamic data manipulation.
* Providing search and visualization features for employee records.
* Handling employee roles, subordinates, and reporting chains.

## 2.4 Modifications Implemented

* Added **role management** during employee transfer.
* Integrated **BFS and DFS** algorithms for searching employees.
* Ensured employees under a transferred manager move directly to the new parent.
* Designed clean and interactive forms for different functionalities.

# 3. Implementation Details

## 3.1 Technologies and Frameworks Used

Table 1:Technologies Used

|  |  |  |
| --- | --- | --- |
| ****Technology/Framework**** | ****Version**** | ****Purpose**** |
| C# (C-Sharp) | .NET 6.0 | Core language for development |
| Windows Forms | WinForms | GUI design and interface creation |
| Visual Studio | 2022 | IDE for project development |
| .NET Framework | 4.8 | Application runtime environment |

## 3.2 Project Structure and Forms

The project consists of **7 forms**:

### 3.2.1. Tree View Form

* Dynamically represents the organization structure as a tree.
* Displays the hierarchy of employees under managers.

### 3.2.2. Add Employee Form

* **Inputs**: Employee ID, Role, Parent ID.
* Adds a new employee under the specified parent.
* **Button**: Submit.

### 3.2.3. Remove/Edit Employee Form

* Two sections:
  + **Remove Section**: Removes an employee by ID.
  + **Edit Section**: Updates an employee's name.
* **Buttons**: Remove, Edit.

### 3.2.4. Transfer Employee Form

* **Inputs**: Employee ID, Current Parent ID, New Parent ID.
* Transfers an employee to a new parent.
* Updates the role dynamically if a manager is transferred.
* **Button**: Transfer.

### 3.2.5. Search Form

* **Inputs**: Search Text, Filter (Name/ID), Search Type (BFS/DFS).
* Searches for employees using BFS or DFS algorithms.
* Results displayed in a **Grid View**.
* **Buttons**: Search.

### 3.2.6. List Subordinates Form

* **Input**: Employee ID.
* Displays a list of all direct subordinates.
* Results displayed in a **Grid View**.

### 3.2.7. Reporting Chain Form

* **Input**: Employee ID.
* Displays the reporting chain of the specified employee up to the root.
* Results displayed in a **Grid View**.

## Classes Used

### 3.3.1 Tree\_node Class

* **Purpose**: Represents a node in the organizational tree. Each node corresponds to an employee or position in the hierarchy.
* **Properties**:
  + - Name: Stores the name of the person or position.
    - Role: Defines the role of the node (e.g., CEO, Manager, Employee).
    - ID: Unique identifier for each node.
    - Children: A list to hold subordinate nodes.
* **Constructor**: Initializes the node with name, role, and ID, and creates an empty list of children.

### 3.3.2 OrgTree Class

* **Purpose**: Manages the hierarchical structure of the organization using Tree\_node objects as nodes.
* **Properties**:

Root: The topmost node of the tree (e.g., the CEO).

* **Methods**:
* FindNode: Searches for a node in the tree based on its ID.
* AddNode: Adds a new node as a child of a specified parent node.
* RemoveNode: Deletes a node and its subtree, ensuring the root node cannot be removed directly.
* DFS: Performs a depth-first search to find nodes matching a search term and filter.
* BFS: Implements a breadth-first search using a queue to find nodes matching a search term and filter.
* TransferEmp: Transfers an employee node from one parent to another, with conditions to manage roles (e.g., Manager to Employee).
* GetReportChain: Retrieves the reporting chain of a specific employee up to the root.
* EditNode: Updates the name of an existing node.

### 3.3.3EmpQueue<T> Class

1. **Purpose**: A generic queue implementation used in the BFS method of OrgTree. It is implemented using a linked list.
2. **Functionality**: Allows enqueue and dequeue operations to traverse the organizational tree level by level.

# ****4. Test Cases****

## 4.1 SubmitBTN\_Click\_1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case** | **ParentID** | **NameEmp** | **Role** | **Parent Role** | **Expected Output** | **Remarks** |
| 1 | 101 | John | Manager | CEO | "Employee Added Successfully" | Valid input where CEO can only add Managers |
| 2 | 101 | John | Employee | CEO | "If the parent is a CEO, the new employee's role must be Manager." | CEO cannot add Employee directly |
| 3 | 201 | Mike | Employee | Manager | "Employee Added Successfully" | Manager can add Employee |
| 4 | 201 | Mike | Manager | Manager | "If the parent is a Manager, the new employee's role must be Employee." | Invalid role for Manager's child |
| 5 | 301 | Alex | Employee | (Invalid Role) | "The parent must be a CEO or Manager to add a new employee." | Parent node role invalid |
| 6 |  | Alex | Employee | - | "Please fill all fields." | Missing ParentID |

## 4.2 RemoveBTN\_Click\_1

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Cases** | **ID\_node** | **Expected Output** | **Remarks** |
| 1 | 101 | "Employee deleted Successfully" | Valid node deletion |
| 2 | 102 | "Employee deleted Successfully" | Valid node deletion |
| 3 | - | "Please fill all fields." | Missing ID\_node input |

## 4.3 EditBtn\_Click\_1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Cases** | **ID\_node** | **NewName** | **Expected Output** | **Remarks** |
| 1 | 101 | Mark | "Employee name updated successfully!" | Valid name update |
| 2 | 999 | Mark | "Employee ID not found." | Node does not exist |
| 3 | - | Mark | "Please fill in both Employee ID and New Name." | Missing empId input |

## 4. 4 ReportBTN\_Click

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Cases** | **ID\_node** | **NewName** | **Expected Output** | **Remarks** |
| 1 | 101 | Mark | "Employee name updated successfully!" | Valid name update |
| 2 | 999 | Mark | "Employee ID not found." | Node does not exist |
| 3 | - | Mark | "Please fill in both Employee ID and New Name." | Missing empId input |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Cases** | **ID\_node** | **NewName** | **Expected Output** | **Remarks** |
| 1 | 101 | Mark | "Employee name updated successfully!" | Valid name update |
| 2 | 999 | Mark | "Employee ID not found." | Node does not exist |
| 3 | - | Mark | "Please fill in both Employee ID and New Name." | Missing empId input |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Cases** | **ID\_node** | **NewName** | **Expected Output** | **Remarks** |
| 1 | 101 | Mark | "Employee name updated successfully!" | Valid name update |
| 2 | 999 | Mark | "Employee ID not found." | Node does not exist |
| 3 | - | Mark | "Please fill in both Employee ID and New Name." | Missing empId input |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Cases** | **Entered ID** | **Reporting Chain Exist** | **Expected Output** | **Remarks** |
| 1 | 101 | Yes | Display reporting chain in ReportTree | Valid ID with a reporting chain |
| 2 | 999 | No | "No reporting chain found for the entered Employee ID." | ID does not exist or chain is empty |

## 4.5 SerachBTN\_Click

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Cases** | **ID\_node** | **NewName** | **Expected Output** | **Remarks** |
| 1 | 101 | Mark | "Employee name updated successfully!" | Valid name update |
| 2 | 999 | Mark | "Employee ID not found." | Node does not exist |
| 3 | - | Mark | "Please fill in both Employee ID and New Name." | Missing empId input |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Cases** | **Search Item** | **Filter** | **Search Type** | **Matches Exist** | **Expected Output** |
| 1 | John | Name | BFS | Yes | Results displayed in searchGrid |
| 2 |  | Name | DFS | - | "Please enter a search term |
| 3 | John | Role | DFS | No | "No results found for the given search criteria. |

## 4.6 TranferBtn\_Click\_1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Cases** | **Emp id** | **Current Parent** | **New Parent** | **Transfer Successful** | **Expected Output** |
| 1 | 101 | 201 | 301 | Yes | Employee transferred successfully." |
| 2 |  | 201 | 301 | N0 | Please fill all the fields." |
| 3 | 291 | - | - | No | Please fill all the fields." |

# 5. Wireframes

## 5.1 TreeView:

Figure 1:TreeView Form

## 5.2 Add Employee

Figure 2:Add Employee Form

## 5.3 Remove Employee

Figure 3:Remove Employee

## 5.4 Transfer Employee

Figure 4:Transfer Employee Form

## 5.5 Search

Figure 5: Search Form

## 5.6 Reporting Chain:

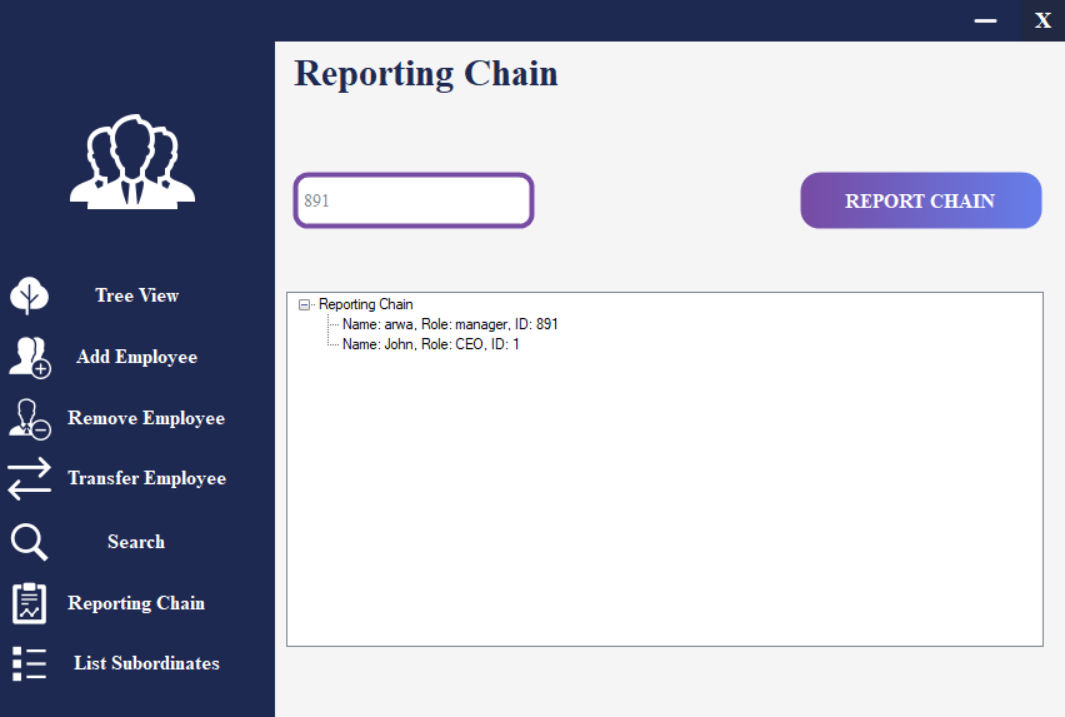


Figure 6:Reporting Chain Form

## 5.7 List Subordinates

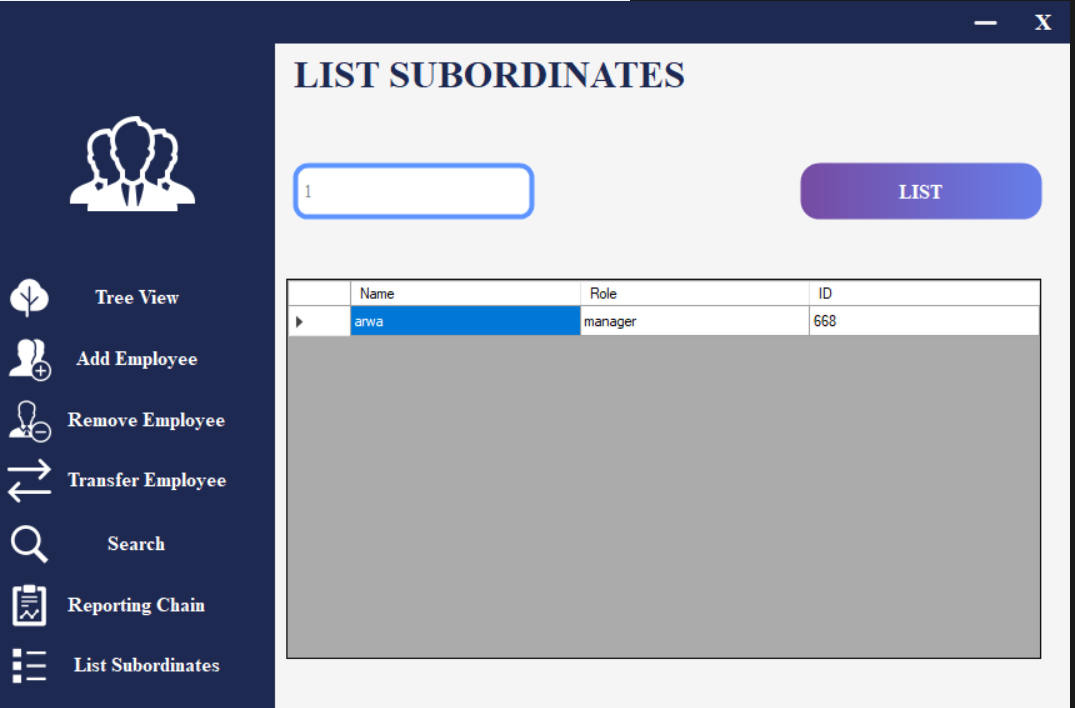


Figure 7:List Subordinate Form

# 6. Conclusion

The **Organizational Hierarchy Management System** successfully provides a user-friendly interface for managing complex hierarchical structures. It efficiently supports operations like employee addition, removal, transfer, search, and subordinate management. The integration of BFS and DFS search algorithms improves usability and search efficiency.

# 7. Future Advancements

* **Database Integration**: Store and retrieve data from an external database.
* **Web-based Application**: Develop a cloud-based version accessible via browsers.
* **Advanced Search**: Add search filters like role, date of joining, etc.
* **Role Visualization**: Implement color coding for different roles in the Tree View.
* **Export Data**: Allow exporting employee data to Excel or PDF files.