**Bytexl's guided project**

**Final Project report on**

# ORGANIZATION HIERARCHY

| | |
|---|---|
| **Name of the educator** | **Rajesh D. Nagawade** |
| **Project title** | **ORGANIZATION HIERARCHY** |
| **Tools / platforms used** | **C Language** |

**About the project:**

This projects represents hierarchy of an Organization. The reporting structure of employees is represented in a customized data structure. Escalation service is provided for unresolved issues. Add, Remove, Search - employee and Print Organization Structure is implemented.

**System requirements:**

C language compiler on Linux or Windows

**Functional requirements:** NIL

**User interface requirements if any:** Runs on console

**Inputs and Outputs:**

1) Input: Run the program

Output: Many EMP ID added message gets displayed and Main gets displayed

2) Input: Choose the option New Employee

Output: Many employees got added message should appear and Main Menu gets displayed again

3) Input: Choose Search Manager Option and 15 Sagar - say employee id of manager

Output: Sagar Manager data gets displayed

4) Input: Choose Escalation and Id of 15

Output: Escalation Level 1 and Level 2 for 15 gets displayed

5) Input: Choose second level employee id and escalation

Output: Escalation Level 1 gets displayed and level 2 does not exist - gets displayed.

6) Input: Choose first level employee id and escalation

Output: Escalation Level 1 and Level 2 does not exist - gets displayed.

**List of subsystems:** NIL

**Other Applications relevant to your project:**

PUMIS, Wipro Backbone, GreytHR

This can be used in employee management systems.

**Designing of Test cases:**

Every Menu Option is chosen with relevant input and expected output is the test case for that input.

**Future Work:**

1) Maintain Project Info and efforts tracking
2) Maintain workdays and leave days for each employee
3) Suggest for leave approval and keep track of leave sanctioned
4) Convey work days and leave days to HR
5) Each employee to report to HR additionally

**References:**

**GreytHR – provides employee details**

**PUIMS – maintains tasks assigned to employees**

**Wipro Employee Backbone System – provides escalation matrix and leave approvals**

**Reflection of the project creation:**

- **Technical challenge:** *Scanning a sub reporting employee in an array inside a node was a challenge.*

- **Existing knowledge** of DSA helped to work with TREE Operations.

- **Benefits individually noticed –** were focusing on a technical bug and alternative ways to solve it.

- **Algorithmic Decisions / Logic / Data Design**
  1) Get node, Create, Search, In Order Traversal, Insert and Delete functions of BST will be used.
  2) Every Node offered by get node – represents an Employee.
  3) Pointer to such employee node will be placed on the tree only if he/she is a manager.
  4) Pointers to reporting employees will be maintained in an array at the manager's node level.
  5) To find escalation level 1 – we have to find employee in these arrays.
  6) To find escalation level 2 – we have to find parent of the manager.
  7) Each node will have following data fields.
     a. Name
     b. Emp ID – used for positioning in the tree
     c. Skill Set – used for project allocation
     d. Active / Free Field
     e. Workdays
     f. Leaves Sanctioned
     g. Reporting Employees - Array of pointers
     h. Pointer to Parent - Thread
  8) To avoid continuous data entry for testing – get node inputs will be taken from statically declared arrays. These can (in future scope) be taken from database or user interface.
  9) Wise decisions are done to keep the tree balanced – enhancing the search and work on the organizational structure.