|  |
| --- |
| **Data Structures and Algorithms** |
| IntelliRoute: Integrated Train and Employee Management System |
| **Course Project Report** |

|  |
| --- |
| **School of Computer Science and Engineering**  **2023-24** |

**Contents**

|  |  |
| --- | --- |
| **Si. No.** | **Topics** |
| 1. | Course and Team Details |
| 2. | Introduction |
| 3. | Problem Definition |
| 4. | Functionality Selection |
| 5. | Functionality Analysis |
| 6. | Conclusion |
| 7. | References |

**1. Course and Team Details**

**1.1 Course details**

|  |  |
| --- | --- |
| **Course Name** | Data Structures and Algorithms |
| **Course Code** | 19ECSC201 |
| **Semester** | III |
| **Division** | B |
| **Year** | 2023-24 |
| **Instructor** | Prakash Hegade |

**1.2 Team Details**

|  |  |  |
| --- | --- | --- |
| **Si. No.** | **Roll No.** | **Name** |
| 1 | 211 | Sanjana Patil |
| 2 | 257 | Pooja Gani |
| 3 | 261 | Nitya Patil |
| 4 | 265 | Shahin Mirakhan |

**1.3 Report Owner**

|  |  |
| --- | --- |
| **Roll No.** | **Name** |
| 265 | Shahin Mirakhan |

**2. Introduction**

The project that was chosen is an integrated train route and employee management system. It falls under the category of software solutions designed to increase the operational effectiveness of human resource management and transportation. This project was inspired by a thorough examination of a white paper that addressed labor management and transportation system automation and optimization. The decision to start this project was motivated by the growing demand for digital transformation in the previously stated fields, especially in light of growing organizational complexity and urbanization. The main goal of the project is to create a solid software system that tackles important issues with employee data management and train route management, ultimately leading to a more efficient and productive operational framework. This system's dual nature highlights its adaptability.

**3. Problem Statement**

**3.1 Domain**

The initiative focuses on two important areas: human resources, particularly personnel management, and transportation management, particularly rail route management. The complex features of staff administration, such as payroll, scheduling, and employee data processing, are noted as a difficulty, as are the manual and sometimes laborious operations involved in managing rail lines, including scheduling, route planning, and station management. The necessity for an automated system to streamline these procedures was emphasized in the white paper, which served as inspiration for developing a solution that would allow for the seamless integration of efficient personnel administration and transportation management.

**3.2 Module Description**

My attention has been on the creation of two essential modules within the project's scope: the Train Route Management System (TMS). The TMS module uses a variety of algorithms and data structures to effectively manage train timetables, station information, and route planning. On the other hand, the UMS module is focused on handling personnel data, such as schedule, time, and pay, and it makes use of unique algorithms to search and organize this data.

**4. Functionality Selection**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Si. No.** | **Functionality Name** | **Known** | **Unknown** | **Principles applicable** | **Algorithms** | **Data Structures** |
|  | **Name the functionality within the module** | **What information do you already know about the module? What kind of data you already have? How much of process information is known?** | **What are the pain points? What information needs to be explored and understood? What are challenges?** | **What are the supporting principles and design techniques?** | **List all the algorithms you will use** | **What are the supporting data structures?** |
| 1 | Route Representation | Network as a graph, where stations are nodes, and connections between stations are edges. | Distances occupancy within time and located area should be closest without overcost. | Abstraction, Modularity | Nodes,vertices. | Graphs |
| 2 | Route Planning | To find the shortest path between two stations, considering the travel time or distance. | Multiple stations with same distance and single path may cause problem for considering route. | Abstraction | Shortest path between points,consider it. | Dijkstra's |
| 3 | Optimal Transfer Points | Optimizes the transfer of routes to minimum transfer points, within minimum cost. |  | Abstraction | Minimum Spanning Tree (MST) . | Prim's or Kruskal's |
| 4 | Real-time Updates | Priortize the latest scheduled route and modify if routes are busy. | Passangers may get informed after leaving the source and moved to destination or already waiting for train/metro. | Abstraction | Priority queue | Heap (priority queue) |
| 5 | User Preferences | As per user prefered there will be the scheduled routine of trains/metros at estimated cost . |  | Abstraction | Trees | Trees (Binary Search Trees, AVL Trees) |
| 6 | Station Information Storage | provides fast access to information based on a key. Here's how you can store station information using a hash table | Newly modified/added information must not upload properly because of servers or connection. | Abstraction | Hashing | Hash Table |
| 7 | Route Identification | To represent various pieces of information, such as station names, locations, and other relevant details. | User Interface of input should be clean and clear to not mess up the source and destination. No conjestion and messy interface. | Abstraction | Efficiently match and identify routes based on user input | String |
| 8 | Load Balancing | Represent the load or occupancy of trains on specific routes. Each element in the array corresponds to a train or route, and the values represent the current load. | Accidents/Clashes may happen if there is no balance of specific routes. | Abstraction | Dijkstra's | Red-black tree |
| 9 | Dynamic Updates | Dynamic update allow for efficient insertion and removal of trains or schedule adjustments ,applied to accommodate varying numbers of trains and schedule modifications. | Work of dynamic updates should be properly mentioned because to notify other stations about modifications. | Modularity | Bubble sort, insertion sort, and selection sort | Dynamic arrays |
| 10 | Caching Mechanisms | Help improve performance, particularly when working with computed or frequently accessible data. |  | Abstraction,Modularity | Bubble sort, insertion sort, and selection sort | Arrays |

**5. Functionality Analysis**

**Train Route Management Functionality Analysis**

1. Reading Station and Route Data: This is the first step where the system reads data from a file. The efficiency depends on the size of the file and the data structure used for storing the information. Larger files and inefficient data structures can increase the time complexity.

2. Initializing and Updating Graph Structure: The graph is initialized based on the station and route data. This involves creating vertices for stations and edges for routes. Updating the graph might involve adding or removing stations/routes. The complexity depends on the graph representation (adjacency matrix/list). An adjacency list is more efficient for sparse graphs.

3. Displaying Stations/Routes: This function traverses the graph to list all stations and routes. The efficiency largely depends on the graph traversal method. For a dense graph represented with an adjacency matrix, this can be O(V^2).

4. DFS and BFS Traversals: Depth-First Search (DFS) and Breadth-First Search (BFS) are used for traversing the graph. DFS is implemented with a stack and has a time complexity of O(V + E), while BFS uses a queue and also has a time complexity of O(V + E). These are efficient for exploring routes and connections.

5. Shortest Path (Dijkstra's Algorithm): Crucial for finding the shortest path between stations. Its efficiency is key, especially in large networks. The standard implementation has a time complexity of O(V^2) using an adjacency matrix, but this can be reduced to O(V + E log V) with a priority queue (min-heap) and adjacency list.

**Employee Management Functionality Analysis**

1. Reading Employee Data: Similar to train route management, the efficiency here depends on the file size and the data structure. A well-chosen data structure can significantly reduce read times.

2. Sorting (QuickSort and HeapSort): Both QuickSort and HeapSort are efficient sorting algorithms. QuickSort, with an average time complexity of O(n log n), is particularly effective for sorting employees by attributes like age. HeapSort also has a time complexity of O(n log n) and is useful for repeatedly finding and removing the smallest (or largest) element, like for salary-based operations.

3. Pattern Searching (Rabin-Karp): Rabin-Karp algorithm, with a worst-case time complexity of O(mn) (m is the length of the pattern and n is the length of the text), is used for pattern searching like finding a substring in employee data. Its average and best-case complexity is O(n + m), making it efficient for average cases.

4. Managing Salary Groups (Union-Find and Fenwick Tree): Union-Find helps in grouping employees and managing their hierarchies or departments with almost constant time operations (amortized O(α(n)), α(n) is the inverse Ackermann function, very slow-growing). The Fenwick Tree, or Binary Indexed Tree, is efficient for cumulative salary operations with a time complexity of O(log n) for update and query operations.

Each function within these modules plays a critical role in ensuring the overall system's efficiency and effectiveness. The choice of algorithms and data structures is crucial for handling large datasets and complex operations within acceptable time frames and resource utilization.

**6. Conclusion**

This project provided insightful learning experiences, particularly in applying complex algorithms and data structures to real-world problems. The integration of different systems (TMS and EMS) underlined the importance of modular design in software development. The challenges encountered, such as optimizing route planning and managing employee data efficiently, emphasized the critical role of algorithmic efficiency in system performance.

**7. References**

DSA Resources - <https://tinyurl.com/4ac6jmpv>

ChatGPT - https://chat.openai.com/

**~\*~\*~\*~\*~\*~\*~\*~**