**Research and Solution Report**

**1. Introduction** This document outlines the solution for the Recursive File Structure test, including research on possible implementations, the superiority of our approach, database design, text file manipulation, and web interface development using PHP without third-party frameworks.

**2. Research on Possible Solutions** Two main approaches exist for handling recursive file structures in programming:

* **Flat Storage with Parent References**: Each file and folder is stored as a record with a reference to its parent.
* **Hierarchical/Nested Storage (Adjacency List, Nested Set, Materialized Path, Closure Table)**: Data is stored in a tree structure format to facilitate recursive queries.

The chosen approach is **Adjacency List**, where each file/folder has a reference to its parent. This approach is simple, efficient, and easy to query for retrieving hierarchical data.

**3. Superiority of my Approach**

* **Scalability**: The adjacency list model is efficient for searching and inserting new data.
* **Ease of Querying**: Recursive queries (CTE) allow quick retrieval of a full directory path.
* **Flexibility**: The structure can be extended dynamically without significant overhead.
* **Optimized Storage**: Reduces redundant data storage compared to nested sets or materialized paths.

**4. Database Design** A MySQL database schema is designed to store the file system structure.

**Table: file\_system**

|  |  |  |
| --- | --- | --- |
| **Column** | **Type** | **Description** |
| id | INT (AUTO) | Unique identifier |
| name | VARCHAR(255) | Name of the file/folder |
| parent\_id | INT | Reference to parent folder (NULL for root) |
| type | ENUM('file', 'folder') | Type (File/Folder) |
| path | TEXT | Full path |

Indexes are placed on parent\_id and name for optimized searching.