**Implementation Overview:**

The implemented system is an in-memory file system that supports basic file and directory operations. It includes functionalities such as creating directories (mkdir), changing the current directory (cd), listing directory contents (ls), creating empty files (touch), writing text to a file (echo), displaying file contents (cat), moving (mv) and copying (cp) files or directories, removing files or directories (rm), searching for a pattern in a file (grep - bonus feature), and saving/loading the file system state.

**Data Structures Used:**

1**. Nested Dictionary (file\_system):**

**Purpose:**

* The primary data structure used to represent the hierarchical structure of the file system.

**Structure:**

* Keys represent directory and file names.
* Values can be either a nested dictionary (representing a directory) or an empty string (representing a file).

Example:

{

'/': {

'home': {

'user': {

'documents': {},

'downloads': {

'file1.txt': '',

'file2.txt': ''

}

}

},

'work': {

'project': {

'report.docx': '',

'presentation.pptx': ''

}

}

}

}

**Usage:**

Supports efficient navigation, creation, and deletion of directories and files.

Mimics the hierarchical structure of a file system.

**Advantages:**

Provides a clear representation of the file system structure.

Enables easy traversal and manipulation of directories and files.

**Considerations:**

Nested dictionaries might lead to increased memory usage for large file systems.

This nested dictionary structure serves as the backbone of the in-memory file system, capturing the relationships between directories and files in a way that is both intuitive and efficient for file system operations. The design choice of using a dictionary aligns with the need for quick access and modification of entries in the file system

**Design Decisions:**

**Class Structure:**

The system is organized into two classes: InMemoryFileSystem and CommandLineInterface. This separation allows for better encapsulation and separation of concerns. The InMemoryFileSystem class handles file system operations, while the CommandLineInterface class manages user interactions through a command-line interface.

**Directory Navigation:**

Directory navigation supports relative paths (.., ../), absolute paths (/), and navigating to a specified path. Special symbols like .. allow moving to the parent directory.

**Error Handling:**

The system includes error handling to address cases where operations are performed on non-existing directories/files, attempting to create duplicate files/directories, etc.

**Infinite Loop with Exit Code:**

The system runs in an infinite loop, and the user can exit the file system by typing the exit command in the terminal.