Project Report: Console-Based Student Portal

Project Name: Technology College Freshers Portal
Primary Language: Python
Target Environment: Command-Line Interface (CLI)

1. Introduction and Project Goals

This project implements a basic simulation of a student information portal designed for use in a command-line environment. The primary goal is to provide authenticated access for students to retrieve academic information (marks and attendance) and initiate administrative processes (leave applications).

The application is built around a simplified, recursive functional structure to handle continuous user interaction within a single session.

2. Core Features

The portal supports the following functionalities, accessible via a main menu after successful login:

Feature	Menu ID	Description
Attendance	1	Displays subject-wise attendance percentage and allows users to view raw present/absent days.
Marks	2	Displays subject-wise academic marks for the current semester.
Leave Process	3	Initiates a leave request requiring three steps: Proctor verification, reason submission, and date calculation. Leave is only granted if attendance is above 75% in all subjects.
Exit/Re-login	4	Terminates the current

		session or prompts for new login credentials.

3. Implementation Details

3.1. Data Management

Student records are stored in a global dictionary named student_record. This dictionary uses the roll_number as the primary key and contains nested dictionaries for subjects, marks, and attendance data.

3.2. Program Structure

The application's logic is primarily driven by a single recursive function, get_day_name(), which manages the main menu and handles transitions between features. The structure strictly adheres to the requested constraint of using minimal functions.

3.3. Isolated Function: CAPTCHA Verification

The **Captcha Verification** logic is contained within a dedicated function, captcha_verification(). This function generates a random 4-digit number, prompts the user for input, and returns a boolean value (True/False) indicating successful verification.

3.4. Leave Day Calculation

The leave process includes an internal utility function, count_leave_days(), which utilizes the datetime module to calculate the number of weekdays between the user-provided start and end dates.

4. Sample Output Demonstration

The following demonstrates a standard login and attendance check sequence using the provided credentials (student1, 101).

_ _ _ _ ----WELCOME TO TECHNOLOGY COLLAGE FRESHERS PORTAL---- _ _ _ _

ENTER YOUR NAME :> student1
ENTER YOUR ROLLNUMBER :> 101
CAPTCHA---3892
ENTER THE CAPTCHA----3892
_ _ _ _ STUDENT LOGIN VERIFIED_ _ _ _
< ENTER <1> FOR VIEWING YOUR ATTENDENCE >
< ENTER <2> FOR VIEWING YOUR MARKS >
< ENTER <3> FOR YOUR LEAVE PROCESS >

```
< ENTER <4> FOR EXIT >
< ENTER <ANYTHING> FOR EXITING PORTAL >
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

ENTER YOUR CHOICE :1

_ _ ATTENDANCE PORTAL FOR CURRENT SEMESTER_ _
ATTENDENCE PERCENTAGE PER SUBJECT
CALCULAS : 60.00%
PHYSICS : 85.00%
ENGLISH : 95.00%
PYTHON : 90.00%
IF YOU WANT TO SEE TOTAL PRESENT AND ABSENT DAYS ENTER <44> IF NOT ENTER ANY
INTEGERS :44
TOTAL PRESENT DAYS PER SUBJECT
{'CALCULAS': 60, 'PHYSICS': 85, 'ENGLISH': 95, 'PYTHON': 90}
TOTAL ABSENT DAYS PER SUBJECT
{'CALCULAS': 40, 'PHYSICS': 15, 'ENGLISH': 5, 'PYTHON': 10}
TO GO BACK TO HOME PAGE PRESS KEY <1> :
1
_ _ _ _ --ENTER YOUR CHOICE-- _ _ _ _
< ENTER <1> FOR VIEWING YOUR ATTENDENCE >
...
```

5. Conclusion

The Technology College Freshers Portal successfully meets the requirements for simulating a student information system within a constrained environment. It demonstrates core login authentication, basic data retrieval, and a multi-step administrative process (leave application). While the design deliberately omits standard Python error handling (try/except), achieving the goal of functioning with minimal structure, this limitation is clearly documented to ensure safe operation. The project serves as a clear example of prioritizing functional requirements over development best practices under strict constraints.

6. Operational Limitations and Constraints

Due to explicit development constraints, the application operates under severe limitations, primarily related to robustness and input handling:

Constraint	**Technical Impact**	**Risk Assessment**

No Error Handling (try/except)	The code cannot gracefully handle ValueError exceptions.	**High Risk:** The program will crash and terminate immediately if the user inputs non-numeric data (letters, symbols) when an integer (choice, CAPTCHA) is expected.
No Input Validation	Input is converted directly using int(input()) or date.fromisoformat().	**High Risk:** Incorrect date formats in the Leave Process (e.g., MM/DD/YYYY instead of YYYY-MM-DD) will cause a critical ValueError crash.
Recursive Structure	The main menu logic relies on recursive calls to get_day_name().	**Low to Medium Risk:** For extremely prolonged, continuous use, deep recursion could eventually lead to Python's maximum recursion depth being reached, although this is unlikely for typical interactive use.