

**B.M.S. COLLEGE OF ENGINEERING BENGALURU**  
Autonomous Institute, Affiliated to VTU



**UNIX SHELL PROGRAMMING**  
**Report on**

**STUDENT CALENDAR NOTIFIER**

*Submitted in partial fulfillment of the requirements for AAT*

Bachelor of Engineering  
in  
Computer Science and Engineering

*Submitted by:*

**Namita Devabasappanavar Anilkumar (1BM23CS202)**  
**Nishta Khariwal (1BM23CS217)**  
**Nithya S (1BM23CS219)**  
**Parash Kashyap (1BM23CS225)**

Department of Computer Science and Engineering  
B.M.S. College of Engineering  
Bull Temple Road, Basavanagudi, Bangalore 560 019  
2024-2025

**B.M.S. COLLEGE OF ENGINEERING**  
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



***DECLARATION***

We, Namita DA (1BM23CS202), Nishta Khariwal (1BM23CS217), Nithya S (1BM23CS219), Parash Kashyap (1BM23CS225), students of 3<sup>rd</sup> Semester, B.E, Department of Computer Science and Engineering, B.M.S. College of Engineering, Bangalore, hereby declare that, this AAT Project entitled "STUDENT CALENDAR NOTIFIER" has been carried out in Department of CSE, B.M.S. College of Engineering, Bangalore during the academic semester September 2024 - January 2025. We also declare that to the best of our knowledge and belief, the Project report is not part of any other report by any other students.

**Signature of the Candidates**

Namita Devabasappanavar Anilkumar (1BM22CS202)

Nishta Khariwal (1BM23CS217)

Nithya S (1BM23CS219)

Parash Kashyap (1BM23CS225)

**B.M.S. COLLEGE OF ENGINEERING**  
**DEPARTMENT OF COMPUTER SCIENCE AND**  
**ENGINEERING**



***CERTIFICATE***

This is to certify that the AAT Project titled “**STUDENT CALENDAR NOTIFIER**” has been carried out by Namita Devabasappanavar Anilkumar (1BM23CS202), Nishta Khariwal (1BM23CS217), Nithya S (1BM23CS219), Parash Kashyap (1BM23CS225) during the academic year 2024-2025.

Signature of the Guide

Signature of Head of the Department

Signature of Examiners with date

1. Internal Examiner

---

2. External Examiner

---

## **ABSTRACT**

The Student Calendar Notifier is a Unix Shell Programming based project designed to automate the and simplify the process of tracking to remind students of important personal and academic events, specifically birthdays and holidays. The system allows students to maintain a calendar file that includes the dates of birthdays, public holidays. The system allows students to manage their academic calendar by setting reminders for holidays and birthdays. Utilizing the Unix shell, the notifier reads from a user-maintained calendar file and section text file which contains names of students and dates of their birthdays, this triggers notifications at specified intervals (e.g., Government holidays like Christmas and Makar Sankranti or any of the students' birthdays. ) using shell scripts. The project takes in user input of the name and section of the user, displays a greeting message and then takes in the date user input. On the basis of the input taken, the program displays whether the date is a holiday, weekend or if any birthdays lie on the day. The project leverages basic Unix tools to parse dates, manage birthday listings, and automate reminders via desktop notifications. This project not only helps in personal organization but also contributes to improving time management by ensuring students never forget important social and academic milestones.

# Table of Contents

1. Introduction	
1.1. Introduction to UNIX	1
1.2. Motivation for the Work	3
2. Methodology and Framework	
2.1. System Requirement	4
2.2. Algorithms, Techniques	4
3. Implementation	
3.1. List of Main UNIX Commands	6
3.2. Uses and its Syntax	8
4. Result and Analysis	12
5. Conclusion and Future Enhancement	18
6. References	19

## List of Figures

Figure 4.1

- Output of the Script 12

Figure 4.2

- Downloaded Files 15

# Chapter 1: Introduction

## 1.1 Introduction to UNIX

UNIX is an innovative or groundbreaking operating system which was developed in the 1970s by Ken Thompson, Dennis Ritchie, and many others at AT&T Laboratories. It is like a backbone for many modern operating systems like Ubuntu, Solaris, Kali Linux, Arch Linux, and also POSIX. Originally, It was designed for developers only, UNIX played a most important role in the development and creation of the software and computing environments. It has all the necessary ingredients, like control structures, loops and variables, that establish a powerful programming language. These features are used to design shell scripts – programs that can also invoke UNIX commands. Many of the system's functions can be controlled and automated by using these shell scripts. Unix follows a modular architecture, in which the functionalities are divided between the two modules: Kernel and Shell. This is division of labour. The kernel interacts with the machine hardware and the shell interacts with the user. Kernel: It is a collection of functions written in language. It is loaded into the memory when the system is booted. Shell: The shell is the command interpreter. It takes the commands given by the user, interprets them and directs them to the kernel for further processing.

### Features of Unix :

#### 1) **Multi User :**

- Multiple users can access the system by connecting to points known as terminals.
- Several users can run multiple programs simultaneously on one system.

#### 2) **Multitasking :**

- The kernel is designed to handle a user's multiple needs. In a multitasking environment, a user sees one job running in the foreground; the rest run in the background.

#### 3) **Pattern Matching :**

- UNIX has very sophisticated pattern matching features.
- Regular Expressions describe a pattern to match, a sequence of characters, not words, within a line of text.

#### 4) **Portable :**

- UNIX can be installed on many hardware platforms.
- The Unix operating system is written in C language, hence it is more portable than other operating systems.

5) **UNIX Toolkit :**

- UNIX offers the ability to add and remove many applications as and when required.
- Tools include:
  - general purpose tools
  - text manipulation tools
  - compilers/interpreters
  - networked applications and
  - system administration tools

6) **Programming Facility :**

- The UNIX shell is also a programming language; it was designed for programmers, not for end users. It has all the necessary ingredients, like control structures, loops and variables, that establish a powerful programming language. These features are used to design shell scripts programs that can also invoke UNIX commands. Many of the system's functions can be controlled and automated by using these shell scripts.

7) **Documentation :**

- The principal on-line help facility available is the man command, which remains the most important references for commands and their configuration files. Apart from the main documentation, there's a vast ocean of UNIX resources available on the internet.

UNIX as a Platform for Student Calendar Notifier :

- UNIX serves as an ideal platform for developing and running the Student Attendance Manager due to its stability, flexibility, and extensive command-line capabilities. As a multitasking and multiuser operating system, UNIX provides a robust environment for managing administrative tasks with minimal resource consumption.

**The Student Calendar Notifier leverages the following features of UNIX:**

1. **Command-Line Interface (CLI):** The terminal-based nature of UNIX allows for efficient execution of commands, file handling, and automation, making it suitable for a lightweight calendar management application.
2. **File Management:** UNIX's powerful file system and utilities like touch, grep, awk, and sed enable effective storage, retrieval, and manipulation of calendar records.
3. **Scripting Capabilities:** Shell scripting and languages like Python or Bash allow developers to automate birthday tracking, holiday generation, and data validation processes.
4. **Resource Efficiency:** UNIX is known for its minimal system requirements and fast performance, making it ideal for institutions with limited computational resources.
5. **Security:** The ability to manage file permissions and secure access to data ensures the confidentiality and integrity of birthday/holiday records.

By utilizing UNIX as a platform, the Student Calendar Notifier combines the power of command line tools with the simplicity of a lightweight application, providing a reliable solution for educational institutions.



## 1.2 Motivation

The motivation behind creating a Student Calendar Notifier stems from the need to help students manage both their academic and personal lives more effectively. Students often juggle multiple responsibilities—such as assignment deadlines, exam dates, extracurricular activities, and social events—while also trying to remember important personal dates like birthdays and holidays. Without an effective system in place, it's easy for important dates to be forgotten, leading to stress, missed opportunities, or even academic penalties.

The motivation to create a Birthday Calendar for students comes from the desire to help students foster and maintain meaningful relationships while balancing their academic responsibilities. Birthdays are significant social events that allow students to celebrate with friends and peers, yet amidst the chaos of academic deadlines and schedules, students often forget or miss these special dates.

The Birthday Calendar aims to address this by providing students with a simple, automated way to remember birthdays. This system ensures that students stay connected with their social circles, acknowledging milestones like birthdays without the stress of having to manually track them. By receiving timely reminders, students can plan small celebrations, send greetings, or even organize group gifts—actions that strengthen friendships and contribute to a more positive social experience during their academic journey. At the end of the day, the goal is to help students stay connected with their friends and make sure they don't miss any important celebrations.

Additionally, using Unix shell programming to develop the notifier serves as a practical exercise in coding and system administration. By automating the process of tracking and reminding about events, the project not only addresses a real-world problem but also teaches important skills in scripting and automation. Ultimately, the Student Calendar Notifier aims to reduce the cognitive load on students, helping them stay organized, reduce stress, and focus on their studies and personal growth.

## Chapter 2. Methodology and Framework

### 2.1. System Requirements:

#### 2.1.1 Hardware Requirements:

1. **Processor:** Any modern CPU (e.g., Intel Core i3, AMD Ryzen 3 or equivalent).
2. **Memory (RAM):** Minimum 2 GB, recommended 4 GB for smoother performance.
3. **Storage:** Around 200 MB free space.
4. **Display:** Basic display with at least 800x600 resolution.
5. **Network:** Stable internet connection for fetching data from APIs. This project can run efficiently on most modern systems with Linux and basic hardware

#### 2.1.2 Software Requirements:

1. **Operating System:** The script is designed for Linux-based operating systems such as Ubuntu, Fedora, and Debian, but can also be run on macOS with minimal changes. For Windows, a Linux subsystem (WSL) or a virtual machine running Linux is required.
2. **Bash Shell:** The script relies on Bash for execution. Bash is a default shell in most Linux distributions and macOS, providing a command-line environment for running scripts.
3. **Zenity:** Zenity is essential for creating graphical user interface (GUI) dialogs in the script. It is used to prompt users for input, display messages, and show errors in a graphical format, making the script more user-friendly.
4. **Internet Access:** Since the script fetches real-time data from external APIs, an active internet connection is required to retrieve weather, air quality, and news data.

### 2.2. Algorithms/Techniques :

1. **Check Dependencies :**
  - Verify if whiptail is installed using command -v whiptail.
  - If not installed, prompt the user to install it and exit the script.
2. **Prompt User for Inputs :**
  - Use whiptail to:
    - Prompt the user to input their name (username).
    - Check if the operation was canceled.
    - Prompt the user to input their class (class).
    - Check if the operation was canceled.
3. **File Existence Check :**
  - Construct the class-specific filename (\${class}.txt).
  - Check if the file exists:
    - If the file is missing, display an error message and exit.

**4. Prompt for Date Input :**

- Use whiptail to:
  - Prompt the user to input a date in DD-MM format (date).
  - Check if the operation was canceled.
- Parse the date into day and month components.
- Validate the date using date -d.

**5. Generate and Display Calendar :**

- Generate a calendar for the given month and year using the cal command.
- Highlight the specific day in the calendar.
- Display the calendar using whiptail.

**6. Greeting Message :**

- Display a greeting message to the user with their name.

**7. Check for Weekends :**

- Determine the day of the week for the given date using date -d.
- If the date falls on a Saturday or Sunday, display a message indicating it's a weekend holiday.

**8. Check for Holidays :**

- Verify the existence of the holidays.txt file.
- If the file exists:
  - Search for the given date in the file.
  - If a match is found, extract the holiday name and display it using whiptail.

**9. Check for Birthdays :**

- Search for the given date in the class-specific file (\${class}.txt).
- If birthdays are found:
  - Extract and display the names of individuals whose birthdays match the date.
- If no birthdays are found: Display a message indicating no birthdays for the date in the class.

**10. Handle Errors Gracefully :**

- Provide meaningful error messages for invalid inputs, missing files, or canceled operations.

## Chapter 3. Implementation

### 3.1 List of main UNIX commands :

1. **whiptail** - Displays dialog boxes for user input and messages.
2. **grep** - Searches for patterns in files.
3. **cut** - Extracts specific fields from text.
4. **sed** - Edits text streams, e.g., highlighting or substitutions.
5. **date** - Handles date validation and formatting.
6. **cal** - Generates calendars for specific months and years.
7. **command -v** - Checks if a command exists on the system.
8. **exit** - Exits the script with a specific status code.

### Source Code

```
#!/bin/bash

# Check if whiptail is installed
if ! command -v whiptail &>/dev/null; then
    echo "whiptail is not installed. Install it using: sudo apt install whiptail"
    exit 1
fi

# Prompt the user to enter their name
username=$(whiptail --inputbox "Enter your name:" 8 40 --title "User Input" 3>&1 1>&2 2>&3)

# Check if the user canceled
if [[ $? -ne 0 ]]; then
    echo "Operation canceled."
    exit 1
fi

# Prompt the user to enter their class
class=$(whiptail --inputbox "Enter your class (e.g.3d):" 8 40 --title "Class Input" 3>&1 1>&2 2>&3)

if [[ $? -ne 0 ]]; then
    echo "Operation canceled."
    exit 1
fi

# Look for class.txt file
filename="{class}.txt"
holidays="holidays.txt"

if [[ -f "$filename" ]]; then
    # Prompt the user to enter a date
    date=$(whiptail --inputbox "Enter the date (in format DD-MM):" 8 40 --title "Date Input" 3>&1
1>&2 2>&3)
```

```

if [[ $? -ne 0 ]]; then
    echo "Operation canceled."
    exit 1
fi

# Extract the day and month from the date
day=$(echo "$date" | cut -d'-' -f1 | sed 's/^0//')
month=$(echo "$date" | cut -d'-' -f2)
# Validate the date input
if ! date -d "2025-$month-$day" &>/dev/null; then
    whiptail --msgbox "Invalid date format. Please enter a valid date (DD-MM)." 8 40 --title
>Error"
    exit 1
fi

#whiptail --msgbox "Calendar for $month 2025:\n\n$cal_output" 20 60 --title "Calendar"
# Generate the calendar for the given month and year
cal_output=$(cal "$month" 2025 | sed "s/\b$day\b/[$day]/")

# Use whiptail to display the calendar
whiptail --msgbox "Calendar for $month 2025:\n\n$cal_output" 20 60 --title "Calendar"

#whiptail --msgbox "Calendar for $month 2025:\n\n$cal_output" 20 60 --title "Calendar"

# Greet the user
whiptail --msgbox "Hi $username!" 8 40 --title "Greeting"
# Check for weekends
day_name=$(date -d "2025-$month-$day" +"%A")
if [[ "$day_name" == "Saturday" || "$day_name" == "Sunday" ]]; then
    whiptail --msgbox "It's a holiday today because it's the weekend!" 8 40 --title "Weekend"
fi

# Check for holidays
if [[ -f "$holidays" ]]; then
    holiday=$(grep "$date" "$holidays" | cut -d' ' -f1)
    if [[ -n "$holiday" ]]; then
        whiptail --msgbox "It's a holiday today because it's $holiday." 8 40 --title "Holiday"
    fi
fi

# Check for birthdays
birthday_matches=$(grep "$date" "$filename" | cut -d' ' -f1)
if [[ -n "$birthday_matches" ]]; then
    whiptail --msgbox "It's the birthday of:\n\n$birthday_matches" 12 40 --title "Birthdays"
else
    whiptail --msgbox "There are no birthdays today in $class." 8 40 --title "No Birthdays"
fi
else
    whiptail --msgbox "The file for $class does not exist. Please check the class name." 8 40 --title
>Error"
fi

```

## 3.2 Uses and its Syntax

### 1. File and Directory Operations :

#### 1. -f (File Test Operator)

- **Purpose:** Tests if a file exists and is a regular file.
- **Usage in Script:**  
bash  
if [[ -f "\$filename" ]]; then
- Checks if the class-specific file (\${class}.txt) or the holidays.txt file exists.

#### 2. grep

- **Purpose:** Searches for a specific pattern in a file.
- **Usage in Script:**  
bash  
grep "\$date" "\$filename"
- Searches for birthdays or holidays in the respective files based on the input date.
- **Example:** If 03-05 is the input, it searches for occurrences of 03-05 in the file.

#### 3. cut

- **Purpose:** Extracts specific fields or segments from text based on a delimiter.
- **Usage in Script:**  
bash  
cut -d '-' -f1
- Extracts the day (f1) or month (f2) from the DD-MM formatted date.
- bash  
cut -d ' ' -f1
- Extracts the holiday or name from the file where the date is matched.

#### 4. sed

- **Purpose:** Performs text substitution or manipulation in a stream of text.
- **Usage in Script:**  
bash  
sed "s/\b\$day\b/[\$day]/"
- Highlights the day in the calendar output by enclosing it in brackets (e.g., [15]).

---

### 2. User Interaction :

#### 1. whiptail

- **Purpose:** Provides a graphical dialog box interface for user input, messages, or confirmations.
- **Usage in Script:** bash

whiptail --inputbox "Enter your name:" 8 40

- Prompts the user to input their name, class, or date.

## 2. Message Box:

- bash  
whiptail --msgbox "Hi \$username!" 8 40
- Displays greetings or information to the user.

## 3. echo

- **Purpose: Prints text to the terminal**
  - **Usage in Script:**  
bash  
echo "whiptail is not installed."
  - Informs the user if a command is missing or displays error messages.
- 

## 3. Process and Error Handling :

### 1. command -v

- **Purpose:** Checks if a command exists in the system.
- **Usage in Script:**  
bash  
command -v whiptail &>/dev/null
- Verifies if whiptail is installed.

### 2. exit

- **Purpose:** Exits the script with a specific status code.
- **Usage in Script:**  
bash
- exit 1
- Terminates the script when a critical issue occurs or the user cancels an operation.

### 3. \$?

- **Purpose:** Captures the exit status of the last executed command.
- **Usage in Script:**  
bash  
if [[ \$? -ne 0 ]]; then
- Determines if the previous command was successful (0) or failed (non-zero).

### 4. if

- **Purpose:** Executes commands conditionally based on success or failure.
- **Usage in Script:**  
bash  
if [[ -f "\$filename" ]]; then
- Executes different blocks of code based on the existence of a file.

#### 5. [[ ... ]]

- **Purpose:** Evaluates conditions like file existence, string comparison, or logical operators.
- **Usage in Script:**  
bash  
[[ -f "\$filename" ]]
- Checks if a file exists.

### 4. Date and Time Manipulation :

#### 1. date

- **Purpose:** Handles date formatting and calculations.
- **Usage in Script:**
- **Convert Day of Week:**  
bash  
date -d "2025-\$month-\$day" +"%A"
- Determines the day name (e.g., Monday) for a given date

#### 2. Validate Date:

bash  
date -d "2025-\$month-\$day" &>/dev/null - Checks if the entered date is valid.

### 5. Text Manipulation

#### 1. (Pipe)

- **Purpose:** Passes the output of one command as input to another.
- **Usage in Script:**  
bash  
cal "\$month" 2025 | sed "s/\b\$day\b/[\$day]/"
- Sends the calendar output to sed for highlighting the day.

#### 2. cal

- **Purpose:** Displays a calendar for a specific month and year.
- **Usage in Script:**  
bash  
cal "\$month" 2025



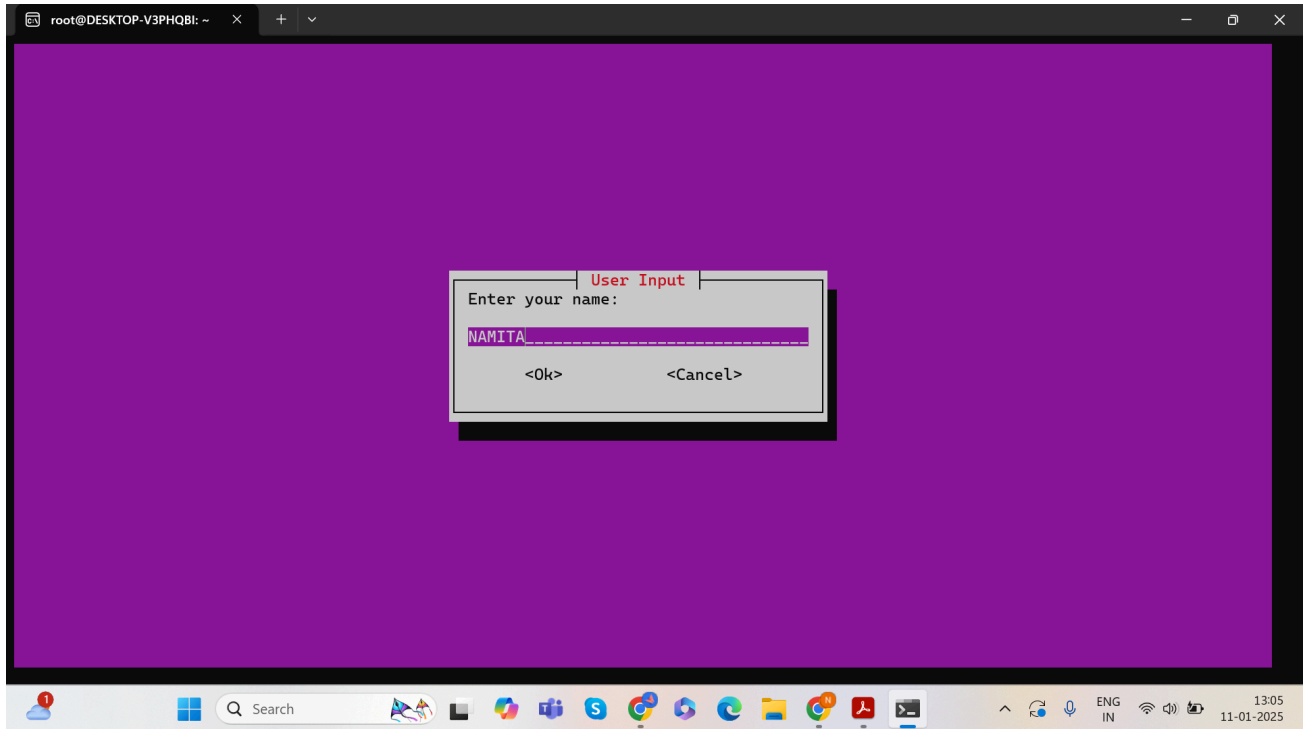
- Generates a calendar for the given month and highlights the specified day.
- 

## 6. Redirection :

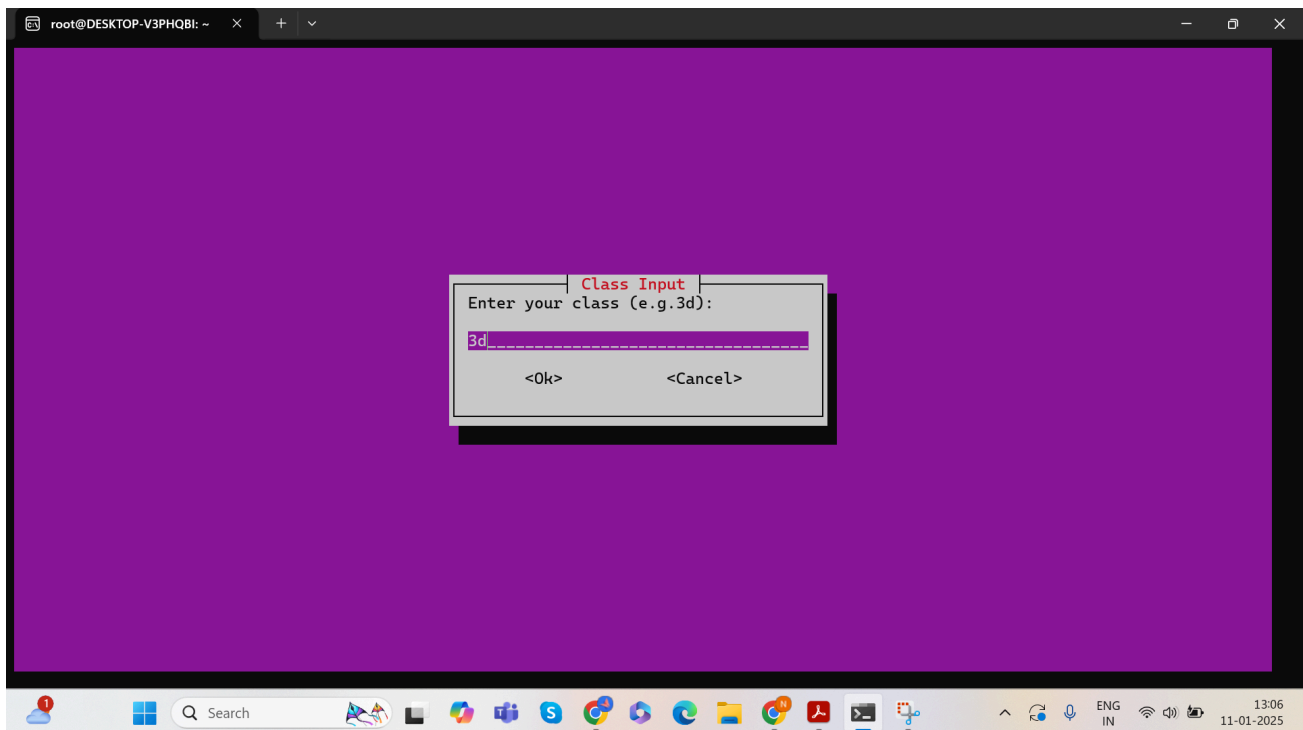
### 1. **&>/dev/null**

- **Purpose:** Suppresses output (both stdout and stderr) by redirecting it to /dev/null.
  - **Usage in Script:**  
bash  
command -v whiptail &>/dev/null
  - Ensures no output is shown when checking for the existence of whiptail.
-

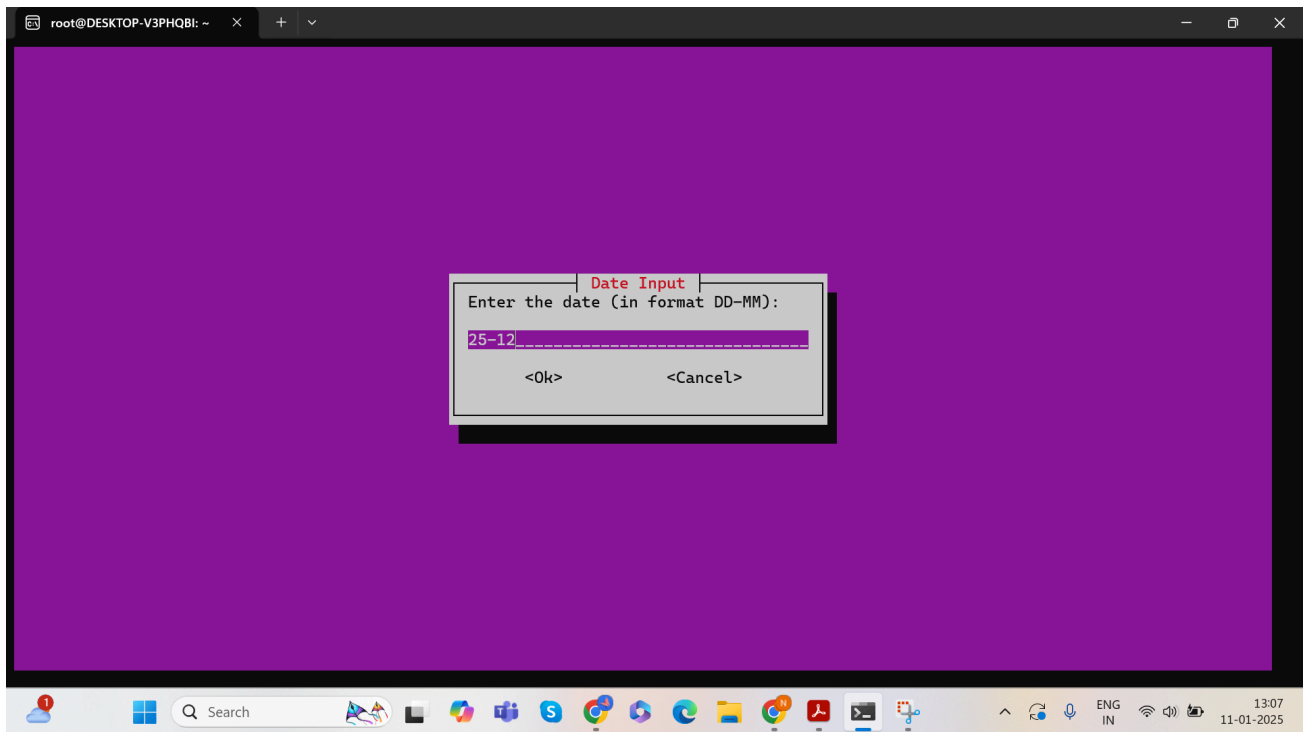
## Chapter 4. Result and Analysis



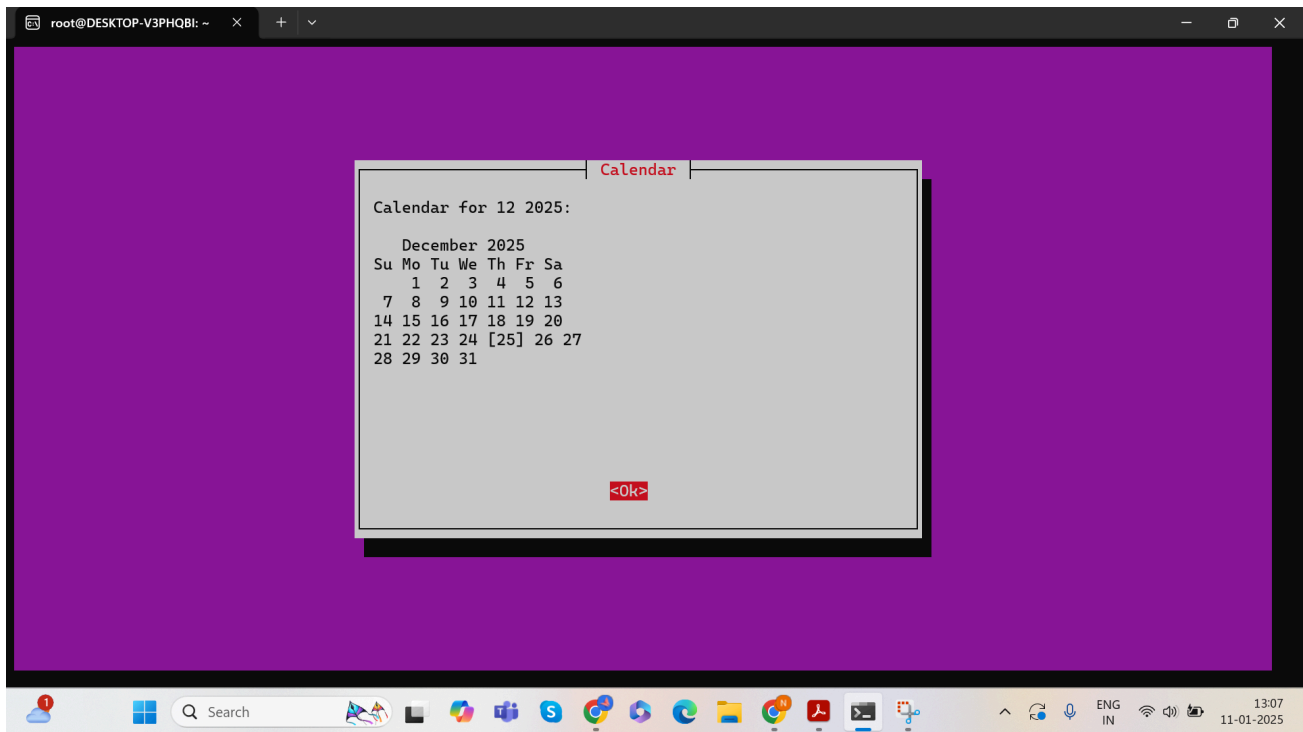
**Fig. 4.1.1 : Users have to enter their name.**



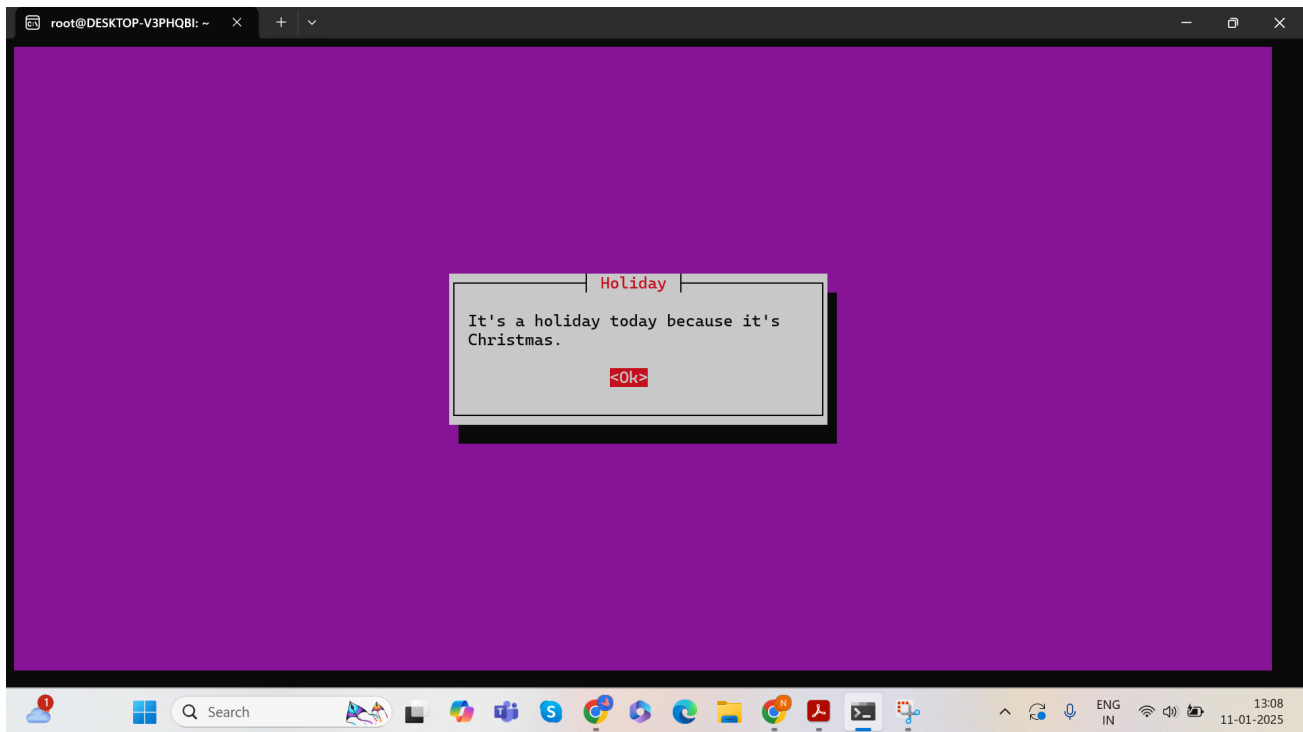
**Fig. 4.1.2 : Users have to enter their class, if the wrong class is entered, the program exits.**



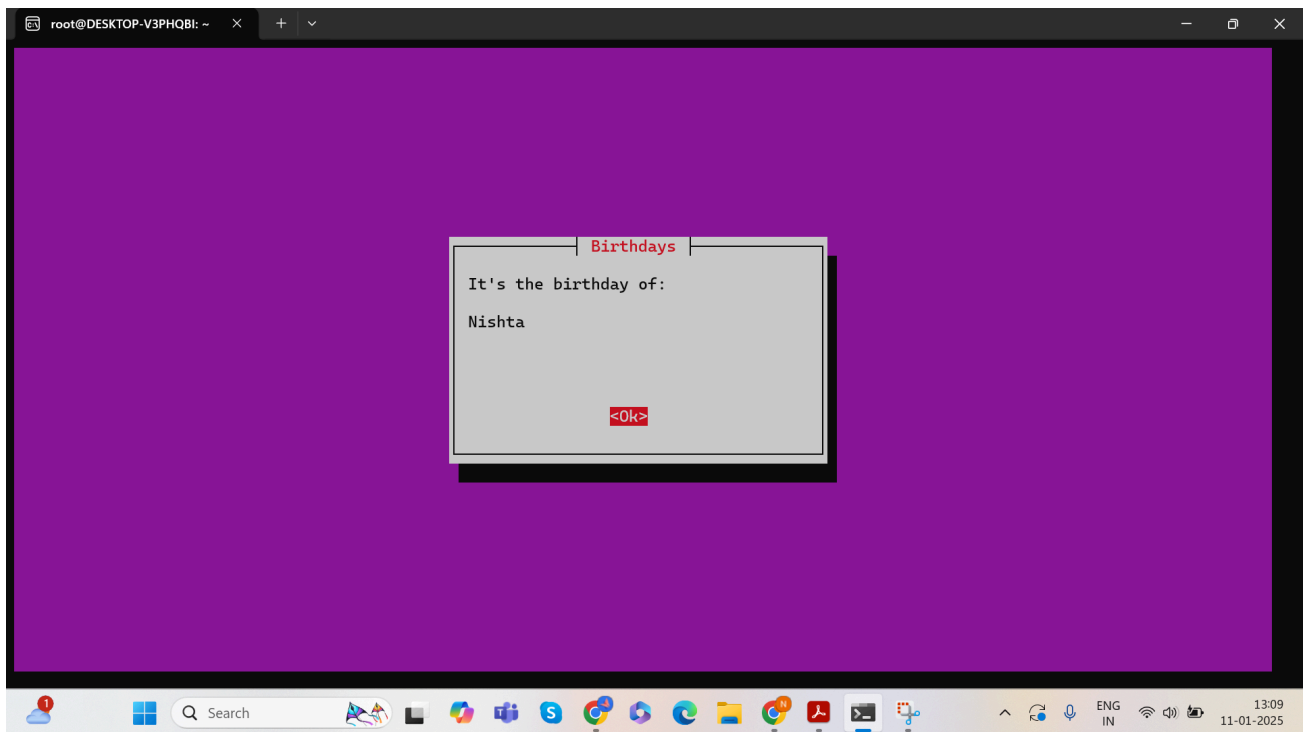
**Fig. 4.1.3 : Users are asked to enter the date of their choice, if invalid date is entered, the program exits.**



**Fig. 4.1.4 : The entered months' calendar appears, along with the date highlighted.**

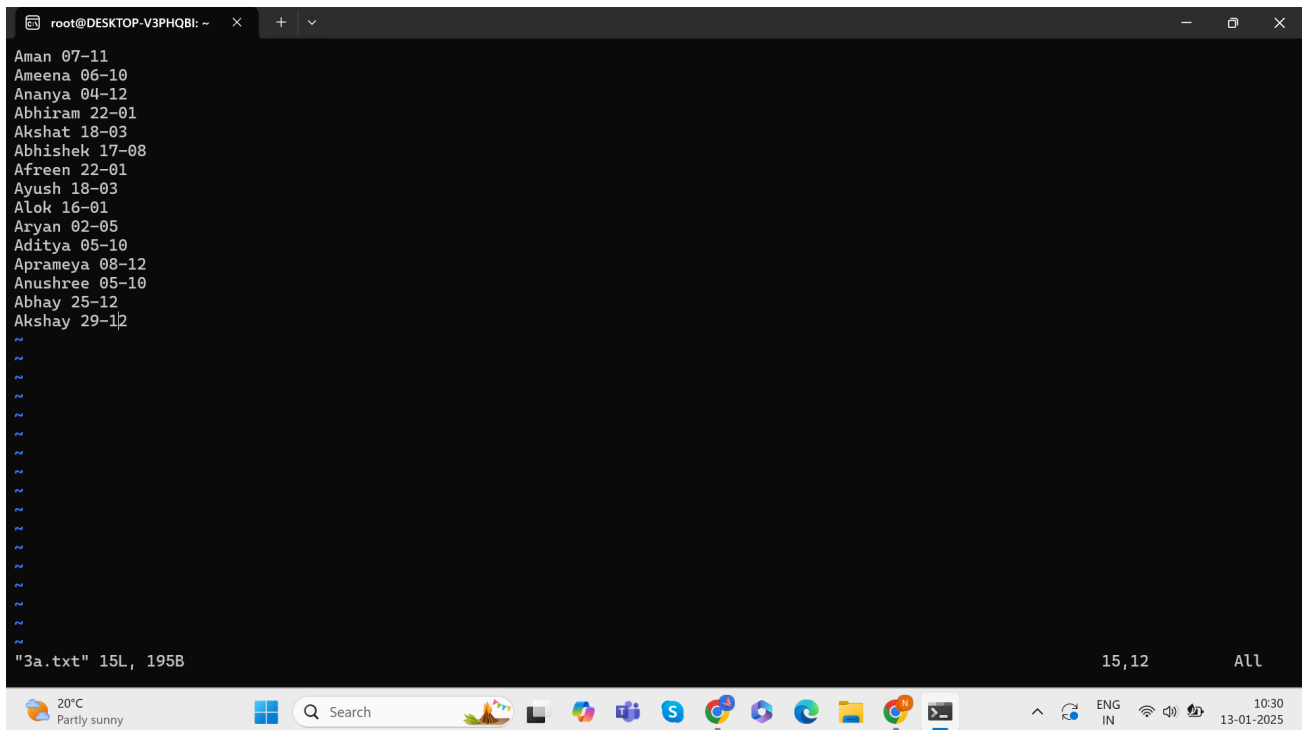


**Fig. 4.1.5 :** If the entered date is a holiday, the above message displays along with the name of the festival.



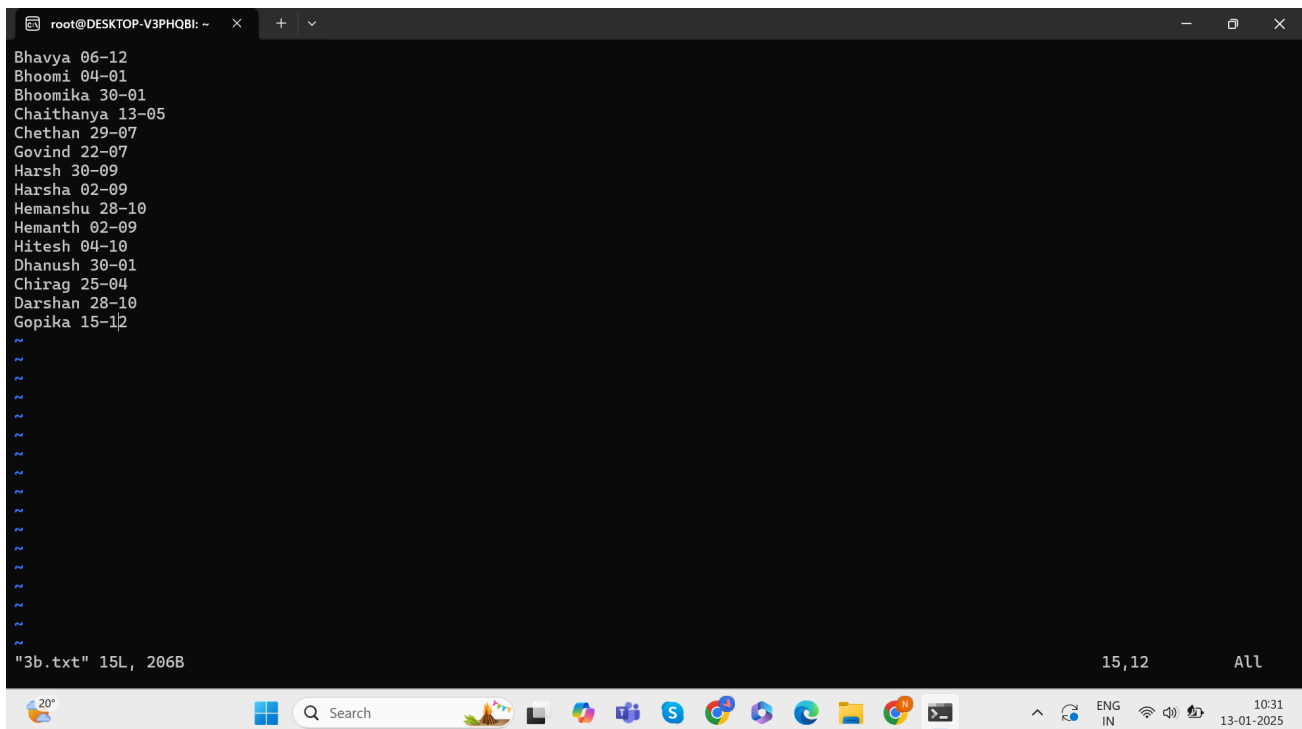
**Fig. 4.1.6 :** If the entered date is a birthday as well, the above message is displayed along with the name.

## Downloaded files



```
root@DESKTOP-V3PHQBI: ~
Aman 07-11
Ameena 06-10
Ananya 04-12
Abhiram 22-01
Akshat 18-03
Abhishek 17-08
Afreem 22-01
Ayush 18-03
Alok 16-01
Aryan 02-05
Aditya 05-10
Aprameya 08-12
Anushree 05-10
Abhay 25-12
Akshay 29-12
"3a.txt" 15L, 195B
```

**Fig 4.2.1 3a.txt file stores names and birthdays of students of 3a.**



```
root@DESKTOP-V3PHQBI: ~
Bhavya 06-12
Bhoomi 04-01
Bhoomika 30-01
Chaithanya 13-05
Chethan 29-07
Govind 22-07
Harsh 30-09
Harsha 02-09
Hemanshu 28-10
Hemanth 02-09
Hitesh 04-10
Dhanush 30-01
Chirag 25-04
Darshan 28-10
Gopika 15-12
"3b.txt" 15L, 206B
```

**Fig 4.2.2 3b.txt file stores names and birthdays of students of 3b**

[illegible]

**Fig 4.2.3 3c.txt file stores names and birthdays of students of 3c.**

[illegible]

**Fig 4.2.4 3d.txt file stores names and birthdays of students of 3d**



## Chapter 5. Conclusion and Future Enhancement

### 5.1 Conclusion

Ultimately, the Student Calendar Notifier is more than just a tool for event tracking—it empowers students to manage their social and academic responsibilities in a balanced way. With timely reminders, students can focus more on their studies and personal growth, while ensuring they don't miss out on important moments in their lives. The project is a great example of how simple automation can improve everyday tasks, making life just a little bit easier and more organized.

The Student Calendar Notifier project offers a practical and efficient solution for students to stay organized and manage their time effectively by automating reminders for important academic deadlines, personal events like birthdays, and holidays. By leveraging Unix shell programming, this system not only simplifies the process of tracking events but also helps students reduce the mental clutter of remembering every detail, while offering valuable hands-on experience in scripting and automation.

### 5.2 Future Enhancement

#### 1. **Multi-Platform Support:**

While the notifier is currently built for Unix-based systems, expanding its functionality to work on different platforms like Windows and macOS would make it more versatile. This could involve using cross-platform programming languages like Python, or adapting the Unix scripts for compatibility with different operating systems.

#### 2. **Mobile App Version:**

A mobile app could be developed to push notifications directly to students' smartphones for reminders, ensuring that they never miss an event even when they are on the go. It could sync with cloud-based services to ensure the calendar is always up to date, no matter where the student is.

#### 3. **Voice Assistant Integration:**

Integrating the calendar with voice assistants like Google Assistant, Siri, or Alexa could allow students to add events, get reminders, or check upcoming events through simple voice commands, making the notifier more hands-free and convenient.

These future enhancements would not only make the Student Calendar Notifier more user-friendly and adaptable but would also expand its reach and functionality, ensuring that it remains a helpful tool for students in managing their academic and personal lives efficiently.



## References :

1. <https://www.geeksforgeeks.org/introduction-to-unix-system/>
2. <https://stackoverflow.com/questions/8317989/calendar-in-linux-shell-programming-problems>
3. <https://blogs.rajinformatica.com/unix-commands/>
4. <https://www.micahlerner.com/2021/07/14/unix-shell-programming-the-next-50-years.html>
5. <https://www.careers360.com/careers/unix-developer>
6. <https://www.slideshare.net/slideshow/shell-programming-10848411/10848411>

