Project Based Learning

Title of Project: Departmental Calendar

|  |  |  |
| --- | --- | --- |
| **Sr.No** | **Name of Students** | **PRNNo** |
| 1 | Sakshi Sanap | B24IT1004 |
| 2 | Nilakshi Patil | B24IT1013 |
| 3 | Rutuja Deskmukh | B24IT1015 |
| 4 | Tanushree Patil | B24IT1049 |
| 5 |  |  |

**Date: Faculty In-Charge**

# Research:

The design of the calendar system involves research into existing solutions, technologies, and algorithms that can be

applied in this context. Key areas of research include:

Date and Time Functions in C: The C programming language provides a time.h library, which offers various functions for

manipulating dates and times. Functions like localtime(), mktime(), and strftime() can be used to calculate and format dates

and times efficiently.

Algorithm for Day of the Week: The key to building a calendar is calculating the day of the week for a given date. One

commonly used algorithm is Zeller’s Congruence, which can be used to determine the weekday for any given date.

Graphical Interface in C: While C does not have built-in graphical libraries like higher-level languages,

ncurses (for terminal-based UIs) or SDL (for graphical UIs) can be explored. The goal is to create a clear, readable calendar layout with appointments listed on the specific dates

# 2. Analysis :

The first step in the development of the Department Calendar is understanding the needs and requirements of the

application. A calendar system is essential for organizing appointments, meetings, events, and reminders, and can be

applied in both personal and professional contexts. For the department use case, the calendar tool needs to be capable

of:

Displaying a calendar view for different months and years.

Adding, editing, and deleting appointments.

Setting reminders for specific events.

User-friendly interface with input handling and graphical display of calendar data.

The core features and functionality expected from the Department Calendar are:

Date Manipulation: Handling leap years, day of the week calculations, and month/year navigation.

Appointment Management: Users should be able to add, view, and edit appointments on specific dates.

Reminder Notifications: The system should notify users of upcoming appointments and events.

Graphical Display: The calendar should be presented in a visually readable format, even in a text-based console interface.

**3) Ideate**

The ideation phase involves brainstorming potential features, interfaces, and solutions for the Department Calendar.

Here are the main ideas and concepts:

User Input and Navigation:

Input Methods: Users will input data via the terminal (command line). They will navigate through the calendar using arrow

keys or specific commands.

Adding Appointments: Users can add appointments via a simple text interface where they input the date, time, and

description.

Graphical User Interface:

Text-Based UI: For simplicity, the calendar can be displayed as a text-based grid. Each day will be represented in a grid

format where appointments are listed below the respective day.

Month/Year Navigation: Users can move forward or backward through months and years using simple commands or arrow

keys.

# Build

# In the build phase, we implement the ideas and solutions gathered in the ideation phase. The tools and technologies used

# will include:

# C Programming:

# time.h library for date and time manipulations.

# File Handling: Use file operations for storing and reading appointment data.

# Basic Data Structures: Arrays or linked lists to manage calendar events.

# Graphical Interface:

# If opting for a text-based interface, we could use ncurses to create a more dynamic user interface.

# For a graphical UI, we could explore SDL2 or OpenGL, but this might be an overkill for a basic calendar.

# Algorithm Implementation:

# Day Calculation: Implement Zeller’s Congruence to calculate the starting weekday of a month.

# Calendar Display: Construct a 7x6 matrix (7 days per week and 6 rows for the longest months) to visually

# Represent the calendar.

# Test:

# Once the calendar tool is built, testing is essential to ensure the system works correctly. The testing phase involves:

# Unit Testing: Test individual components like date manipulation functions, file input/output, appointment addition and deletion.

# System Testing: Test the full calendar workflow: displaying a calendar, adding/removing appointments, and ensuring the reminder system functions.

# Edge Case Testing: Test edge cases like leap years, invalid date inputs, and large date ranges.

# Test performance by handling a large number of appointments and long-term usage.

# 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test case id | Test case desc | Test date | Expected Dates | Status(Pass/Fail) | Additional Notes |
| TC01 | Add a single event to a month | 2024-01-01 | New year | Pass | Matches |
| TC02 | Add multiple events to the same month | 2024-08-15 | Independence Day | Pass | Matches |
| TC03 | Handle leap year adjustment for february | 2024-02-14 | Displays 29 days | Pass | Matches |
| TC04 | Check display of events | 2024-11-01 | Diwali | Pass | Matches |
| TC05 | Test of semester | 2024-12-09 | Semester exams displayed in may | Pass | Matches |
| TC06 | Max event limit handling | 2024-12-01 | Displays Max event limit | Pass | Matches |

# 

# 

# Implement :

# The Department Calendar Project is a practical and valuable tool that applies core programming concepts to solve a real-world problem. Through the analysis, research, ideation, build, and testing phases, the project will result in a functional, user-friendly calendar system. By leveraging C programming, the system will be efficient, portable, and customizable for the department’s scheduling needs.

# 

# .