B.Tech - C++ mini project

Group Number : 5
Group Member:
3. Hazel Lakshmana – 150096725155
GITHUB-LINK:https://github.com/hazelnutbytes/cpp-mini-project.git
Cohort: Sam Altman

1. PROGRAM DETAILS

1.1 Title of the Module:
Productivity Timer (Console-Based C++ Application)

1.2 Purpose of the Program:
1. The purpose of this program is to design and develop a console-based Productivity Timer using C++.
2. The application helps users manage focused work sessions and breaks.
3. It includes predefined modes such as Sprint, Break, Custom Timer, and Pomodoro cycles.
4. The system demonstrates core Object-Oriented Programming concepts.
5. The program provides a real-time countdown display with dynamic progress indicators.
6. It enhances productivity using structured time management techniques.

1.3 Target Users:
1. Students preparing for exams

2. Developers practicing productivity techniques
3. Individuals using the Pomodoro method
4. Beginners learning C++ and Object-Oriented Programming

1.4 Tools Used:
1. C++
2. Visual Studio Code
3. macOS Terminal
4. GitHub for version control

2. KEY FEATURES

1. Real-time countdown timer
2. Multiple timer modes (Sprint, Break, Custom, Pomodoro)
3. Function overloading for flexible timer input
4. Inheritance-based mode structure
5. Dynamic progress bar display for minutes and seconds
6. Terminal cursor control for live updating interface
7. Audio alert when timer completes
8. Menu-driven interactive system

3. OOP CONCEPTS USED

3.1 Class Concept:
1. The program uses two classes: Timer and TimerModes.

2. The Timer class acts as the parent class containing the core countdown engine.
3. The TimerModes class acts as the child class that manages different productivity modes.

3.2 Inheritance:

1. TimerModes inherits from Timer.
2. This allows reuse of the countdown logic.
3. It maintains a clear separation between timer engine and user interaction.
4. It demonstrates the IS-A relationship in Object-Oriented Programming.

3.3 Encapsulation:

1. Data members such as remainingSeconds and completed are defined inside the class.
2. These variables are accessed and modified using member functions.
3. This ensures controlled access and structured program design.

3.4 Function Overloading:

1. The startTimer function is overloaded.
2. One version accepts only minutes.
3. Another version accepts both minutes and seconds.
4. This demonstrates compile-time polymorphism.

3.5 Constructor:

1. The Timer class includes a constructor.
2. It initializes remainingSeconds to 0.
3. It initializes completed to false.
4. This ensures the object starts in a valid state.

4. C++ CONCEPTS USED

4.1 Header Files:

1. iostream for input and output operations.
2. iomanip for formatting time display using setw and setfill.
3. unistd.h for sleep function to pause execution.
4. cstdlib for executing system commands.

4.2 Menu-Driven Programming:

1. A do-while loop is used to repeatedly show the menu.
2. A switch-case structure is used to handle user choices.
3. The program runs until the user selects the exit option.

4.3 Loops:

1. A while loop is used for the countdown mechanism.
2. A for loop is used for Pomodoro cycle repetition.
3. A for loop is used to display progress bar blocks.

4.4 Time Formatting:

1. Time is displayed in MM:SS format.
2. setw is used to maintain fixed width.

3. setfill ensures leading zeros are displayed.

4.5 Terminal Control:

1. Carriage return is used to overwrite the same line.
2. Escape sequences are used to move the cursor up.
3. flush ensures immediate output display.
4. This creates a dynamic updating interface.

4.6 System Command Execution:

1. The system function is used to execute a terminal command.
2. It plays an alert sound when the timer reaches zero.

=

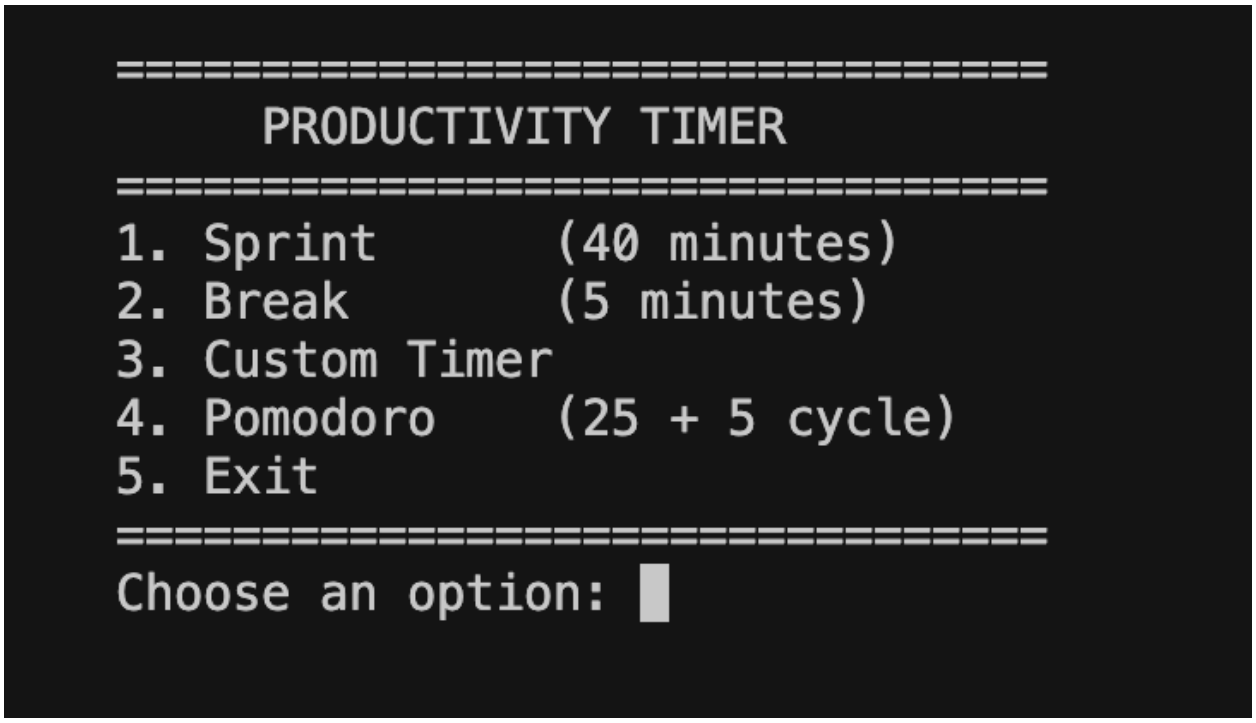
5. PROGRAM FLOW

1. The program starts in the main function.
2. A TimerModes object is created.
3. The run function displays the menu.
4. The user selects a timer mode.
5. The selected mode calls the appropriate startTimer function.
6. The countdown function runs and updates every second.
7. The progress bars update dynamically.
8. When time reaches zero, the alert message is displayed.
9. The sound is played.
10. The menu appears again until the user exits.

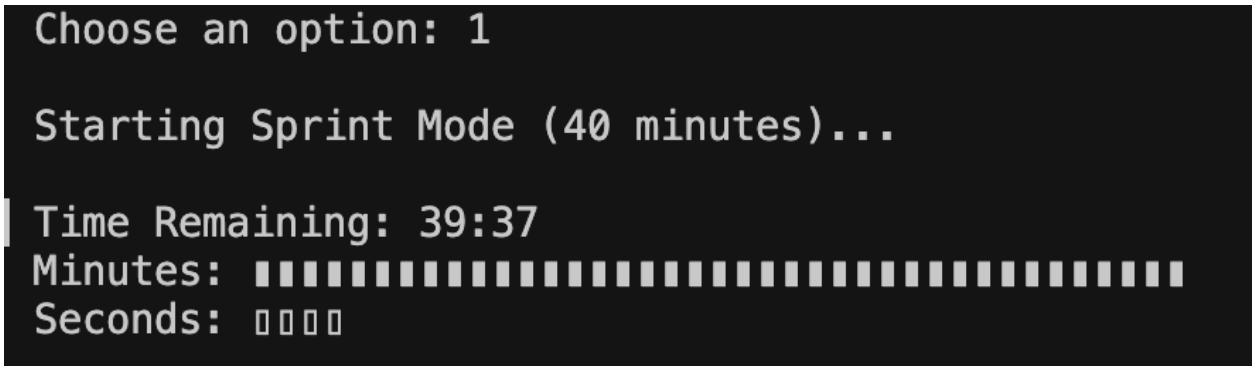
B.Tech - C++ mini project

6. SCREENSHOTS

1. Main Menu Interface



2. Sprint Mode Countdown



B.Tech - C++ mini project

---

3. Custom Timer Input

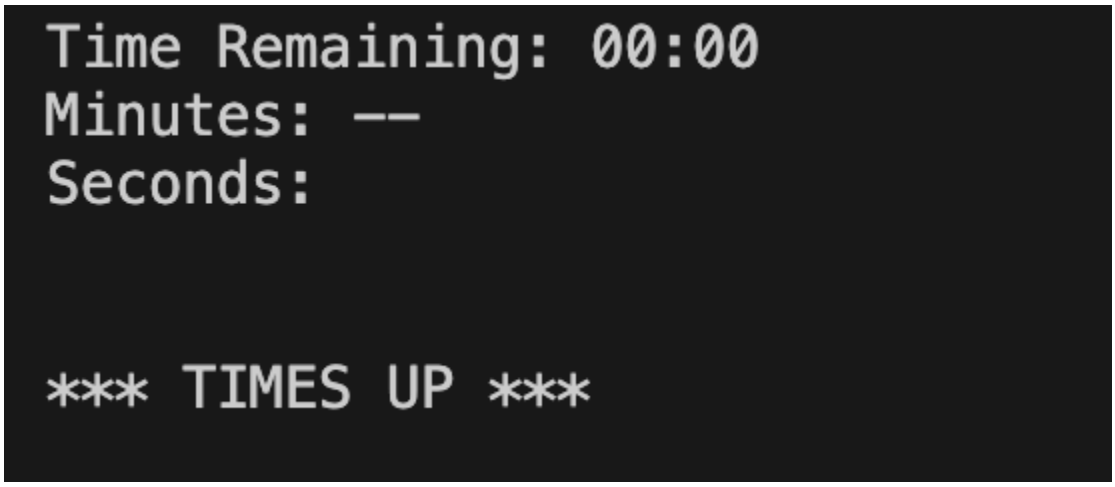
```
Choose an option: 3

Enter minutes: 0
Add seconds too? (y/n): y
Enter seconds: 10

Starting Custom Timer (0m 10s)...

Time Remaining: 00:04
Minutes: --
Seconds: ▯
```

4. Pomodoro Cycle Execution

```
==================================
Choose an option: 4

How many Pomodoro cycles? 1

--- Pomodoro Cycle 1 of 1 ---

WORK phase (25 minutes)...

Time Remaining: 24:53
Minutes: ▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮▮
Seconds: ▯▯▯▯▯▯
```

B.Tech - C++ mini project

5. Time Completion Alert Screen



7. DECLARATION

I hereby declare that this practical work entitled
“Design and Development of a Productivity Timer using C++”
is my original work carried out by me.
This work represents my individual contribution to the group project and has not been copied from any source nor submitted earlier for any examination.

Name of Student: Hazel Lakshmana
Roll Number: 150096725155
Cohort Name: Sam Altman
Signature of Student: hazel
Date: 22/02/2026

B.Tech - C++ mini project

---

8. FUTURE SCOPE

1. Adding pause and resume functionality.
2. Implementing file handling to store timer history.
3. Creating a graphical user interface version.
4. Making the program cross-platform.
5. Adding productivity statistics and analytics.
6. Integrating database support.
7. Adding user authentication and profiles.