SRI SIDDHARTHA ACADEMY OF HIGHER EDUCATION

(Deemed to be University, accredited by NAAC with A+ Grade)

Agalakote, Tumakuru – 572107, Karnataka

OOPS-22AM305

Project Assignment Report

on

"POMODORO TIMER AND TASK MANAGER"

By

1.Sudheendra Theertha H P - 22AI056

2.Reem Mohd Bin Mahfooz -22AI048

3.Manu N - 22AI033

4.Alina - 23AI004

Dept. of AI &ML, SSIT, Tumakuru.

SRI SIDDHARTHA INSTITUTE OF TECHNOLOGY

(A constituent College of SSAHE) Maralur, Kunigal Road, Tumakuru-572105: 2024-25

ABSTRACT

The Pomodoro Timer is a productivity application built in Java using the Swing framework. It implements the Pomodoro Technique, a time management method that uses a timer to break work into intervals, typically 25 minutes, separated by short breaks. This technique has its roots in ancient Japanese learning practices. The "kings of knowledge" in Japanese legend were said to have used similar methods to study and gain profound understanding. Their records of experiments on the human mind were long dismissed until an Italian scholar and scientist, Francesco Cirillo, visited Japan and learned about these techniques.Cirillo then invented the Pomodoro method, which consists of two timers - one for 25 minutes of focused work, and another for a 5-minute break. This cycle helps trigger the release of "happy maker" hormones during the 25-minute intervals, motivating the user to study and learn more. While it may feel challenging at first, the simplicity of the Pomodoro method has made it a popular productivity tool for learners seeking to maximize their knowledge acquisition.By breaking work into manageable chunks and incorporating regular breaks, the Pomodoro Timer helps users maintain concentration, reduce burnout, and cultivate the same focused, diligent approach to learning as the legendary Japanese scholars of the past. Its integration with the Swing framework in Java allows for a user-friendly, customizable interface to support a wide range of learners and productivity needs.

LIST OF CONTENTS

I. INTRODUCTION

The Pomodoro Technique was invented by Francesco Cirillo in the late 1980s while he was a university student, struggling to focus on his studies. Frustrated by his inability to maintain concentration, Cirillo developed this time management method using a tomato-shaped kitchen timer (hence "pomodoro", Italian for tomato). The technique revolutionizes productivity by breaking work into focused 25-minute intervals called "pomodoros", separated by short breaks, which helps combat mental fatigue, reduces burnout, and enhances cognitive performance. Key features include structured work sessions that promote deep focus, mandatory short breaks to prevent mental exhaustion, and a systematic approach to tracking tasks and time, allowing individuals to measure and improve their productivity while maintaining a balance between intense work and necessary rest periods.

Project Objectives:

1.Improve Focus: Maximize concentrated work by creating distraction-free 25-minute intervals that train the brain to maintain sustained attention and minimize context switching.

2. Manage Mental Energy: Prevent burnout by implementing structured breaks that allow cognitive recovery, helping maintain consistent productivity throughout workday and reducing mental fatigue.

3.Time optimization is a strategic approach to work management that transforms temporal resources into a precisely measured and dynamically adaptable system. By breaking tasks into measurable units, rigorously tracking completion rates, and systematically analysing personal productivity patterns, individuals can develop a more intentional and data-driven relationship with time. This methodology involves logging task durations, identifying peak performance periods, minimizing interruptions, and continuously refining work strategies based on empirical insights, ultimately converting time from an abstract concept into a controllable and optimizable professional asset.

II. SYSTEM MODEL

In the context of the pomodoro timer Block Diagram, the "System Module" (also referred to as "pomodoro timer system") represents the central component that integrates and coordinates the entire system. It acts as a bridge between the System module, User Module, and Notification Module.

Block diagram:

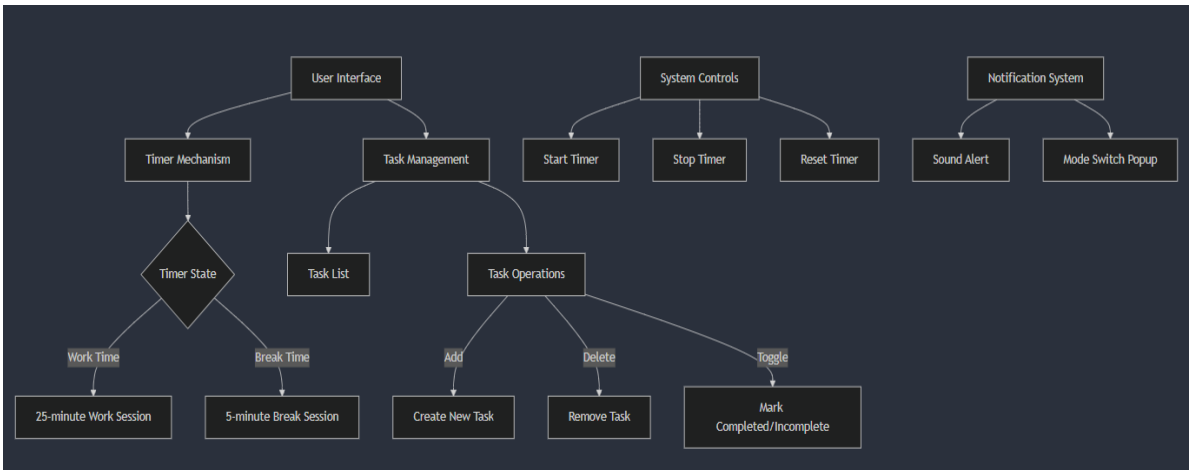

Fig 2.1:Block diagram of currancy cnverter

1. This system brings together a Pomodoro Timer and Task Manager for complete productivity management.

2. Users can maintain focus through customizable 25-minute work sessions with the Pomodoro Timer.

3. Short breaks of 5 minutes follow each work session.

4. Upon completing four pomodoros, users earn a longer break lasting 15-30 minutes.

5. The timer provides both sound and visual alerts to notify users of session changes.

6. Detailed tracking and statistics help users monitor their productivity patterns.

7. The Task Manager component allows for easy task creation, modification, and removal.

8. Tasks can be prioritized to help users focus on what matters most.

9. Each task can be assigned an estimated number of pomodoros for completion.

10. The system tracks completed pomodoros per task to help with time management.

CODE OF THE POMODORO TIMER

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.Timer;
import java.util.TimerTask;

public class PomodoroTimer extends JFrame {
    private JLabel timeLabel;
    private JButton startButton, stopButton, resetButton;
    private JButton addTaskButton;
    private JList<Task> taskList;
    private DefaultListModel<Task> listModel;
    private Timer timer;
    private int timeLeft;
    private boolean isRunning;
    private final int WORK_TIME = 25 * 60; // 25 minutes in seconds
    private final int BREAK_TIME = 5 * 60; // 5 minutes in seconds
    private boolean isWorkTime = true;

    public PomodoroTimer() {
        setTitle("Pomodoro Timer with Task Manager");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(400, 500);
        setLayout(new BorderLayout());
```

```
        // Initialize components
        initializeComponents();

        // Layout setup
        setupLayout();

        timeLeft = WORK_TIME;
        updateTimeLabel();
        isRunning = false;
    }

    @SuppressWarnings("unused")
    private void initializeComponents() {
        // Time display
        timeLabel = new JLabel("25:00", SwingConstants.CENTER);
        timeLabel.setFont(new Font("Arial", Font.BOLD, 40));

        // Control buttons
        startButton = new JButton("Start");
        stopButton = new JButton("Stop");
        resetButton = new JButton("Reset");
        addTaskButton = new JButton("Add Task");

        // Task list
        listModel = new DefaultListModel<>();
        taskList = new JList<>(listModel);
        taskList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
```

```
    // Add mouse listener for task completion toggle
    taskList.addMouseListener(new MouseAdapter() {
        @Override
        public void mouseClicked(MouseEvent e) {
            if (e.getClickCount() == 2) {  // Double click
                int index = taskList.locationToIndex(e.getPoint());
                if (index >= 0) {
                    Task task = listModel.getElementAt(index);
                    task.toggleComplete();
                    taskList.repaint();
                }
            }
        }
    };

    // Add keyboard listener for delete
    taskList.addKeyListener(new java.awt.event.KeyAdapter() {
        @Override
        public void keyPressed(java.awt.event.KeyEvent evt) {
            if (evt.getKeyCode() == java.awt.event.KeyEvent.VK_DELETE) {
                deleteSelectedTask();
            }
        }
    });

    // Add action listeners for buttons - removed unused lambda parameters
    startButton.addActionListener(__ -> startTimer());
    stopButton.addActionListener(__ -> stopTimer());

    resetButton.addActionListener(__ -> resetTimer());

    addTaskButton.addActionListener(__ -> addNewTask());

}


private void deleteSelectedTask() {
    int selectedIndex = taskList.getSelectedIndex();
    if (selectedIndex != -1) {
        listModel.remove(selectedIndex);
    }
}

private void setupLayout() {
```

```
        // Timer panel
        JPanel timerPanel = new JPanel(new BorderLayout());
        timerPanel.add(timeLabel, BorderLayout.CENTER);

        // Button panel
        JPanel buttonPanel = new JPanel(new FlowLayout());
        buttonPanel.add(startButton);
        buttonPanel.add(stopButton);
        buttonPanel.add(resetButton);

        // Combine timer and buttons
        JPanel topPanel = new JPanel(new BorderLayout());
        topPanel.add(timerPanel, BorderLayout.CENTER);
        topPanel.add(buttonPanel, BorderLayout.SOUTH);

        // Task panel with instructions
        JPanel taskPanel = new JPanel(new BorderLayout());
        JLabel instructionsLabel = new JLabel(
            "<html><center>Double-click to toggle task completion<br>Press Delete to
remove task</center></html>",
            SwingConstants.CENTER
        );
        taskPanel.add(instructionsLabel, BorderLayout.NORTH);
        taskPanel.add(new JScrollPane(taskList), BorderLayout.CENTER);
        taskPanel.add(addTaskButton, BorderLayout.SOUTH);

        // Add to frame
        add(topPanel, BorderLayout.NORTH);
        add(taskPanel, BorderLayout.CENTER);
    }

    private void startTimer() {
        if (!isRunning) {
            isRunning = true;
            timer = new Timer();
            timer.scheduleAtFixedRate(new TimerTask() {
                @Override
                public void run() {
                    if (timeLeft > 0) {
                        timeLeft--;
                        SwingUtilities.invokeLater(() -> updateTimeLabel());
                    } else {
                        timer.cancel();
                        isRunning = false;
                        SwingUtilities.invokeLater(() -> switchMode());
```

```
                }
            }
        }, 1000, 1000);
    }
}

private void stopTimer() {
    if (isRunning) {
        timer.cancel();
        isRunning = false;
    }
}

private void resetTimer() {
    stopTimer();
    timeLeft = isWorkTime ? WORK_TIME : BREAK_TIME;
    updateTimeLabel();
}

private void switchMode() {
    isWorkTime = !isWorkTime;
    timeLeft = isWorkTime ? WORK_TIME : BREAK_TIME;
    updateTimeLabel();
    String message = isWorkTime ? "Work Time!" : "Break Time!";
    Toolkit.getDefaultToolkit().beep();  // Add sound notification
    JOptionPane.showMessageDialog(this, message);
}

private void updateTimeLabel() {
    int minutes = timeLeft / 60;
    int seconds = timeLeft % 60;
    timeLabel.setText(String.format("%02d:%02d", minutes, seconds));
}

private void addNewTask() {
    String taskName = JOptionPane.showInputDialog(this, "Enter task name:");
    if (taskName != null && !taskName.trim().isEmpty()) {
        Task newTask = new Task(taskName);
        listModel.addElement(newTask);
    }
}

public static void main(String[] args) {
    try {
        // Set system look and feel
```

```
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        } catch (Exception e) {
            e.printStackTrace();
        }

        SwingUtilities.invokeLater(() -> {
            new PomodoroTimer().setVisible(true);
        });
    }
}

class Task {
    private String name;
    private boolean completed;

    public Task(String name) {
        this.name = name;
        this.completed = false;
    }

    public void toggleComplete() {
        completed = !completed;
    }

    @Override
    public String toString() {
        return (completed ? "[✓] " : "[ ] ") + name;
    }
}
```

III . FLOW CHART FOR THE CODE

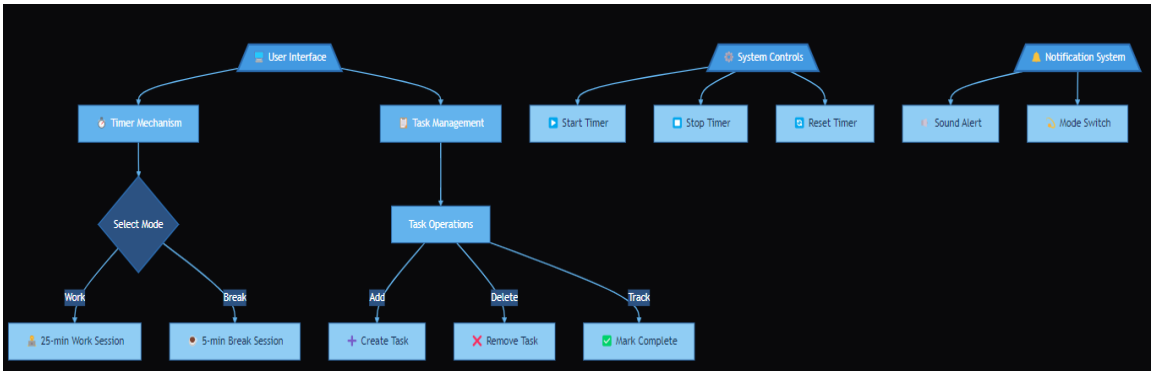

Fig 3.1: Flow chart of the pomodoro timer

1. User Interface: This is the main entry point for the user to interact with the application.

2. Timer Mechanism: This component manages the core functionality of the Pomodoro timer, handling the transitions between work and break sessions.

3. Task Management: This component is responsible for managing the task list, including operations to add, delete, and toggle task completion.

4. System Controls: This component provides the user interface for controlling the timer, with buttons to start, stop, and reset the timer.

5. Notification System: This component is responsible for providing notifications to the user, such as sound alerts and mode switch popups, to indicate changes in the timer state.

The diagram shows the various interactions between these components. For example, the User Interface interacts with the Timer Mechanism and Task Management components to control the timer and manage tasks. The System Controls component triggers actions in the Timer Mechanism, while the Notification System receives information from the Timer Mechanism to alert the user.

III. RESULT (OUTPUT SCREENSHOT)

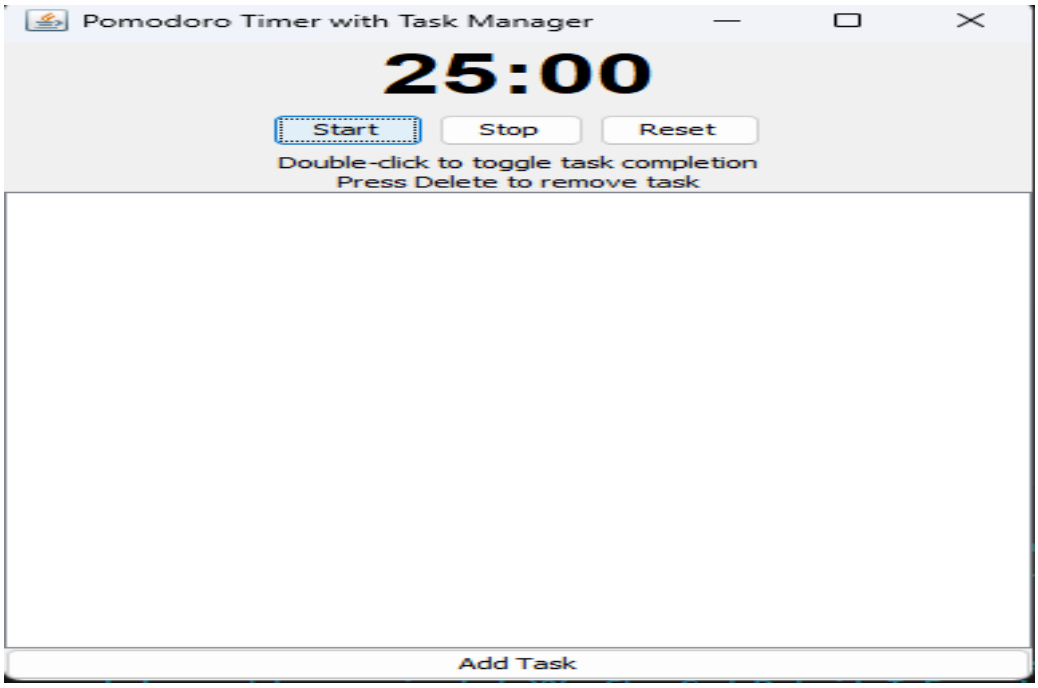

Fig 4.1.: pop-up page

Figure 4.1 shows pop-up of the page just after run to start with timer of 25 minutes.

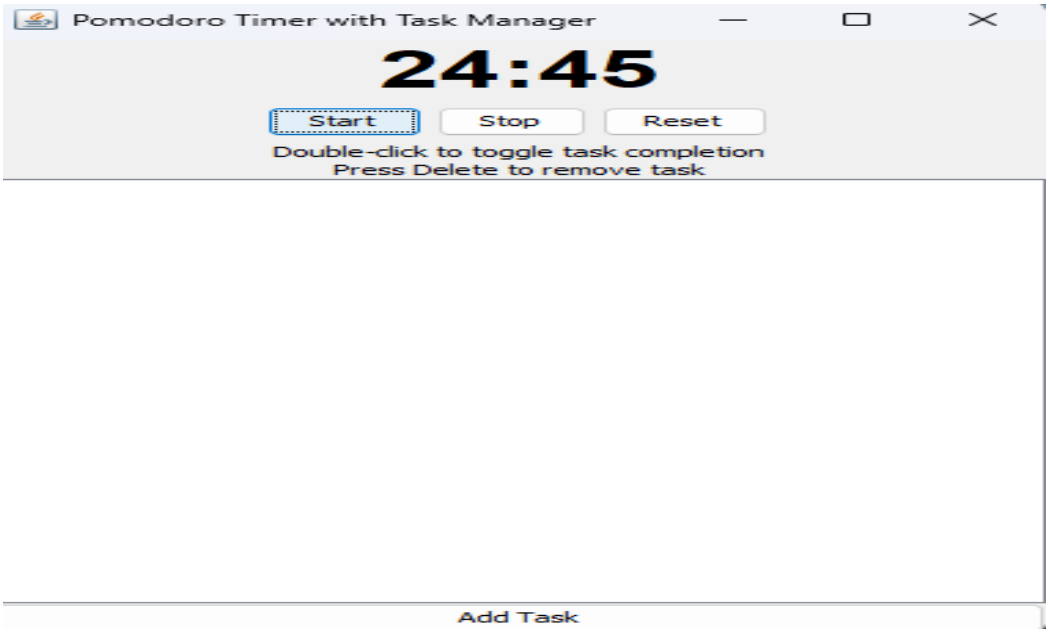

Fig 4.2: 25min timer starts by pressing start

Figure 4.2 shows that 25 min timer is started by pressing startand also have stop and reset for convinence

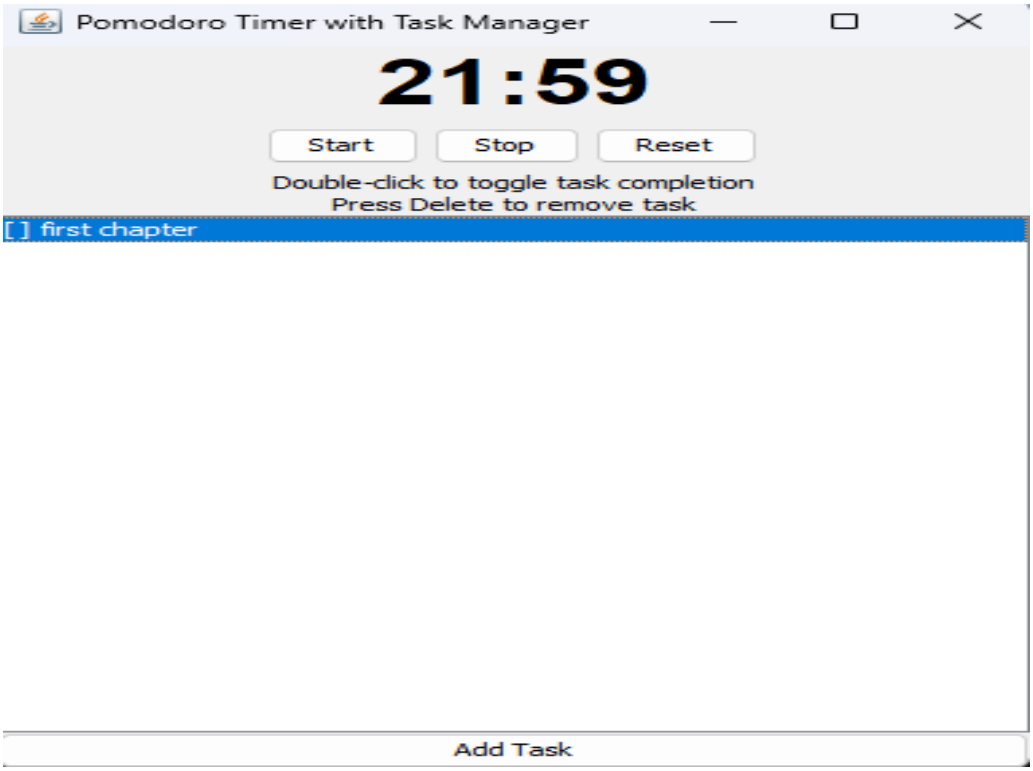

Fig4.3: new task can be added and deleted

Figure 4.3 shows that the timer also has options for adding and removing the tasks for more convinence

V. ADVANTAGES

1. Technical Advantages:

- User-friendly GUI implementation using Java Swing
- Modular code structure for easy maintenance
- Real-time timer updates using Java Timer class
- Persistent task management system
- Event-driven architecture for responsive interactions
- Cross-platform compatibility through Java
- Simple yet effective notification system

2. Productivity Advantages:

a.) Time Management:

- Structured 25-minute work intervals
- Automated break reminders
- Clear visual time tracking
- Prevents overworking

b.) Task Organization:

- Visual task listing
- Easy task addition and deletion
- Task completion tracking
- Quick task status updates
- Organized workflow management

c.) User Benefits:

- Improved focus and concentration
- Reduced mental fatigue
- Better work-break balance
- Enhanced productivity tracking
- Visual progress monitoring
- Reduced procrastination
- Better time awareness

d.) Learning Benefits:

- Practical experience with GUI development
- Understanding of event handling
- Implementation of design patterns
- Experience with timer mechanisms
- Data structure usage (DefaultListModel)
- User interface design principles
- State management implementation

V. CONCLUSION

The Pomodoro Timer and Task Manager project represents a robust implementation of the renowned time management technique, combining essential productivity features with a user-friendly interface built using Java Swing. The application successfully integrates timer functionality with task management capabilities, allowing users to maintain focused work sessions while organizing their tasks effectively. The system implements core features such as 25-minute work intervals, 5-minute breaks, task tracking, and notification alerts, all within a clean and intuitive interface.

Future Enhancements: -

the project could be expanded to include several advanced features: a statistics dashboard showing daily/weekly productivity metrics with graphical representations; data persistence using a database to store task and session history; customizable timer intervals to accommodate different work preferences; category tagging for tasks to enable better organization; a priority system for task management; integration with calendar applications for schedule synchronization; cloud synchronization to access tasks across multiple devices; detailed reporting functionality to analyze productivity patterns; support for multiple concurrent projects; dark/light theme options for better user experience; mobile app integration for on-the-go access; break activity suggestions; and achievement badges or gamification elements to increase motivation. Additionally, implementing a team collaboration feature would allow multiple users to share tasks and track group progress, making it suitable for both individual and team productivity management. The project could also benefit from adding sound customization options, keyboard shortcuts for power users, and automation features like automatic task scheduling based on historical patterns.

IV.REFERENCES

- Coding empire project about java language : https://youtu.be/_Px3DcV1I0Y?si=lstPQQLRmJm1mCPm

- The Effectiveness of Pomodoro Technique on Students' Descriptive Text Writing Quality,by **Widya Eka Septiani, Sulistyaningsih Sulistyaningsih, Abd. Syakur published in 2024 jan:**https://www.researchgate.net/publication/360789500_The_Effectiveness_of_Pomodoro_Technique_on_Students'_Descriptive_Text_Writing_Quality.
- A Learning Assessment Applying Pomodoro Technique as A Productivity Tool for Online Learning, Jefferson Costales, Janice Abellana, Joel Gracia, Madhavi Devaraj published in 08 February 2022 from ACM digital library,Newdelhi, https://dl.acm.org/doi/abs/10.1145/3498765.3498844
- Pomodoro Technique by Francesco Cirillo , Date of publication : 19 October 2006,: http://friend.ucsd.edu/reasonableexpectations/downloads/Cirillo%20--%20Pomodoro%20Technique.pdf