A

JAVA Project

On

STOPWATCH

Submitted in partial fulfilment of the requirement

for the award of the degree of

Bachelor of Technology

In

INFORMATION TECHNOLOGY

II YAER - I SEMESTER

By

THOUTI HARSHITHA (24J41A1259)

Under the guidance of

Mrs.D.Pushpa

Information technology department

MALLAREDDY ENGINNERING COLLEGE

UGC Autonomous Institution, Approved by AICTE, & Affiliated to JNTUH).

(Accredited by NAAC with 'A++' Grade (III Cycle),

Maisammaguda (H), Medchal-Malkajgiri District, Secunderabad,Telangana500100

www.mrec.ac.in

Information Technology Department

CERTIFICATE

This is to certify that the project entitled "**Stopwatch using Java**" is a bonafide work done in partial fulfilment for the award of degree of Bachelor of Technology in Information Technology of Malla Reddy Engineering College done by me during the academic year 2025-2026 under the guidance and supervision.

THOUTI HARSHITHA **24J41A1259**

Internal guide.

Mrs. D. Pushpa

Head of the Department

Dr. M. Deena Babu

ABSTRACT

This project is a Java-based GUI stopwatch built using Swing. It includes Start, Pause, Stop, and Reset buttons to control time displayed in HH:MM:SS format. The program uses a Timer and ActionListener to update and manage events, demonstrating simple and effective GUI programming in Java. It also showcases event-driven programming by managing user interactions through button actions. The intuitive layout and real-time updates make it a practical example for beginners learning Java Swing and timer-based applications.

INTRODUCTION

The Stopwatch Project is a user-friendly Java GUI application built using the Swing framework. It allows users to measure elapsed time precisely in hours, minutes, and seconds. The interface includes essential control buttons — Start, Pause, Stop, and Reset — providing full functionality for time tracking. The use of a javax.swing.Timer ensures that the stopwatch updates every second, displaying the current elapsed time dynamically.

This project highlights the implementation of event-driven programming and object-oriented design in Java. By integrating GUI components such as JFrame, JLabel, and JButton, it offers an interactive and visually appealing experience. The Stopwatch Project serves as an excellent example for beginners to understand how logic and interface design can work together in building real-time desktop applications.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my teacher and guide for their valuable support and guidance throughout the development of this Stopwatch project. Their encouragement and suggestions helped me to understand Java programming concepts and implement them effectively. I am also thankful for their cooperation and motivation during this project. This project has enhanced my knowledge of object-oriented programming and improved my practical coding skills in Java.

EXISTING SYSTEM

In the existing system, time measurement is often done manually using physical stopwatches or basic digital timers. These methods require manual start and stop, which may lead to human errors and inaccurate results. Most existing stopwatch systems lack advanced features such as lap time tracking, pause/resume, or data storage. Traditional systems don't offer user-friendly interfaces or integration with computer applications.

PROPOSED SYSTEM

The proposed system introduces a Java-based stopwatch application developed using the Swing framework. It allows users to start, pause, stop, and reset time with an easy-to-use graphical interface. The system uses a Timer to update time every second and ActionListener to handle user actions efficiently. This application offers better flexibility, accuracy, and a clear understanding of event-driven GUI programming in Java. It provides a practical way to learn how real-time applications can be built using simple programming logic.

SOFTWARE REQUIREMENTS:

1. **Operating System:** Windows / macOS / Linux
2. **Programming Language:** Java
3. **Java Version**: JDK 8 or above (recommended JDK 17+)
4. **IDE / Editor:** Visual Studio Code, Eclipse, or IntelliJ IDEA
5. **Libraries Used:** Java Swing (javax.swing, java.awt, java.awt.event)
6. **Execution Environment:** Java Virtual Machine (JVM)

HARDWARE REQUIREMENTS:

1. **Processor:** Intel Core i3 or higher
2. **RAM:** Minimum 2 GB (4 GB recommended)
3. **Hard Disk Space:** At least 100 MB of free space
4. **Display:** 1024 × 768 resolution or higher
5. **Input Devices:** Keyboard and Mouse
6. **System Type:** 32-bit or 64-bit compatible system

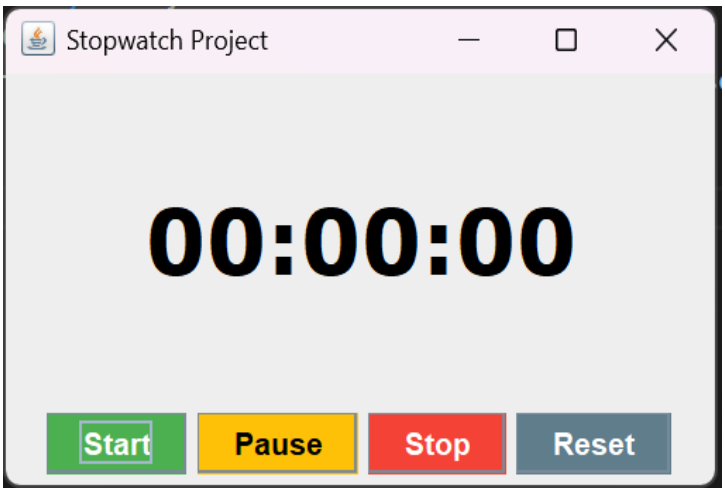
SOURCE CODE

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class Project extends JFrame implements ActionListener {
    private JLabel timeLabel;
    private JButton startButton, stopButton, pauseButton, resetButton;
    private Timer timer;
    private int elapsedTime = 0; // milliseconds
    private boolean running = false;
    private boolean paused = false;

    public Project() {
        setTitle("Stopwatch Project");
        setSize(350, 220);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());
        setLocationRelativeTo(null);

        // Time display label
        timeLabel = new JLabel("00:00:00", SwingConstants.CENTER);
        timeLabel.setFont(new Font("Verdana", Font.BOLD, 40));
        timeLabel.setForeground(Color.BLACK);
        add(timeLabel, BorderLayout.CENTER);

        // Buttons panel
        JPanel buttonPanel = new JPanel();
        buttonPanel.setLayout(new FlowLayout());

        startButton = new JButton("Start");
        pauseButton = new JButton("Pause");
        stopButton = new JButton("Stop");
        resetButton = new JButton("Reset");

        startButton.setBackground(new Color(76, 175, 80)); // green
        startButton.setForeground(Color.WHITE);
        pauseButton.setBackground(new Color(255, 193, 7)); // yellow
        pauseButton.setForeground(Color.BLACK);
        stopButton.setBackground(new Color(244, 67, 54)); // red
```

```
    stopButton.setForeground(Color.WHITE);
    resetButton.setBackground(new Color(96, 125, 139)); // gray-blue
    resetButton.setForeground(Color.WHITE);

    Font buttonFont = new Font("Arial", Font.BOLD, 14);
    startButton.setFont(buttonFont);
    pauseButton.setFont(buttonFont);
    stopButton.setFont(buttonFont);
    resetButton.setFont(buttonFont);

    buttonPanel.add(startButton);
    buttonPanel.add(pauseButton);
    buttonPanel.add(stopButton);
    buttonPanel.add(resetButton);

    add(buttonPanel, BorderLayout.SOUTH);

    // Timer updates every second (1000 ms)
    timer = new Timer(1000, new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            elapsedTime += 1000;
            int hours = elapsedTime / 3600000;
            int minutes = (elapsedTime / 60000) % 60;
            int seconds = (elapsedTime / 1000) % 60;
            timeLabel.setText(String.format("%02d:%02d:%02d", hours, minutes, seconds));
        }
    });

    // Button actions
    startButton.addActionListener(this);
    pauseButton.addActionListener(this);
    stopButton.addActionListener(this);
    resetButton.addActionListener(this);

    setVisible(true);
}

@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == startButton) {
        if (!running) {
```

```
            timer.start();
            running = true;
            paused = false;
        }
    } else if (e.getSource() == pauseButton) {
        if (running && !paused) {
            timer.stop();
            paused = true;
        } else if (paused) {
            timer.start();
            paused = false;
        }
    } else if (e.getSource() == stopButton) {
        timer.stop();
        running = false;
        paused = false;
        elapsedTime = 0;
        timeLabel.setText("00:00:00");
    } else if (e.getSource() == resetButton) {
        timer.stop();
        elapsedTime = 0;
        running = false;
        paused = false;
        timeLabel.setText("00:00:00");
    }
}

public static void main(String[] args) {
    new Project();
}
}
```

OUTPUT:



CONCLUSION

The Stopwatch Project is a Java-based GUI application developed using the Swing framework. It provides basic functions such as start, pause, stop, and reset. The program displays time in hours, minutes, and seconds with a simple interface. It makes use of components like JFrame, JPanel, JButton, and JLabel. Event handling is managed through the ActionListener interface and a Timer. The project helps understand GUI design and event-driven programming in Java. It is useful for learning how real-time applications work. Overall, the project is efficient, interactive, and easy to use.