Smart Focus Timer
A MINI-PROJECT REPORT
Submitted by

NEHA.R.N 240701354
NISHANI.B 240701363

in partial fulfillment of the award of the degree
of
BACHELOR OF ENGINEERING
IN
COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI
An Autonomous Institute
CHENNAI
NOVEMBER 2025

BONAFIDE CERTIFICATE

Certified that this project “SMART FOCUS TIMER” is the bonafide work of “NEHA.R.N, NISHANI.B” who carried out the project work under my supervision.

SIGNATURE
Mrs Anitha Muthulingam
ASSISTANT PROFESSOR SG

Dept. of Computer Science and Engg,
Rajalakshmi Engineering College
Chennai

This mini project report is submitted for the viva voce examination to be held on _____

INTERNAL EXAMINER EXTERNAL EXAMINER

ABSTRACT

The Smart Focus Timer is a sleek and minimal productivity application built using JavaFX, designed to help users enhance concentration and manage time effectively through a Pomodoro-style timer. The application allows users to set customizable focus durations, track progress through a visually engaging circular progress clock, and manage tasks with built-in options to add, edit, and delete session goals. To create a calming and motivating work environment, the app integrates a built-in music player that offers soothing lofi tracks, along with sound cues for session start, pause, and reset actions. The dark mode interface ensures a distraction-free user experience suitable for prolonged focus sessions. Developed using core Java and JavaFX features like Canvas, FXML, and MediaPlayer, Smart Focus Timer provides a seamless blend of functionality, aesthetics, and user comfort—making it a powerful tool for productivity and mindful time management.

ACKNOWLEDGEMENT

We express our sincere thanks to our beloved and honorable chairman MR. S. MEGANATHAN and the chairperson DR. M.THANGAM MEGANATHAN for their timely support and encouragement.

We are greatly indebted to our respected and honorable principal Dr. S.N. MURUGESAN for his able support and guidance.

No words of gratitude will suffice for the unquestioning support extended to us by our Head Of The Department Dr. E.M. MALATHY and our Deputy Head Of The Department Dr. J. MANORANJINI for being ever supporting force during our project work

We also extend our sincere and hearty thanks to our internal guide Anitha Muthulingam, for her valuable guidance and motivation during the completion of this project.

Our sincere thanks to our family members, friends and other staff members of computer science engineering.

1. NEHA.R.N
2. NISHANI.B

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

The Smart Focus Timer is a productivity-enhancing desktop application designed to help users manage their time efficiently and improve concentration during study or work sessions. It implements the Pomodoro technique, allowing users to set focus durations, take breaks, and maintain a balanced workflow. The system integrates a customizable timer, task management features, and built-in lofi music playback for a calm and focused environment. Developed using Java and JavaFX, the application combines functionality with a minimal, modern user interface to create an engaging and distraction-free productivity tool.

1.2 SCOPE OF THE WORK

The project aims to simplify time management by offering a digital platform that enhances focus and task organization. It enables users to customize focus sessions, track their progress visually, and manage daily goals through an interactive task list. The built-in music player and dark mode interface further enhance the user experience by promoting relaxation and reducing screen strain. The system is suitable for students, professionals, and anyone seeking to boost productivity and maintain consistent focus throughout their work sessions.

1.3 PROBLEM STATEMENT

In today's digital environment, individuals often struggle to maintain concentration due to constant distractions and poor time management habits. Traditional timers or manual tracking methods lack motivation and real-time feedback. There is a need for an intuitive, automated solution that not only tracks time but also encourages consistency through interactive visuals and ambient sound. The Smart Focus Timer addresses this issue by offering a seamless and user-friendly platform that integrates time tracking, task management, and relaxation features to improve productivity and focus.

1.4 AIM AND OBJECTIVES OF THE PROJECT

Aim:

To develop a desktop-based productivity application that enhances focus and time management through a customizable timer and interactive tools.

Objectives:

- To implement a Pomodoro-style timer with customizable focus and break durations.
- To provide a task management system for organizing and prioritizing goals.
- To integrate lofi background music for maintaining a calm and focused atmosphere.

- To offer visual progress tracking through a real-time circular progress indicator.
- To ensure a minimal and distraction-free user interface with dark mode support.

CHAPTER 2
SYSTEM SPECIFICATIONS

2.1 Hardware Specifications

Processor	:	Intel i5
Memory Size	:	8 GB (Minimum)
Hard Disk (HDD)	:	500 GB (Minimum)

2.2 Software Specifications

Operating System	:	Windows 10
Front-End	:	JavaFX
Audio Library	:	JavaFX MediaPlayer
Programming Languages	:	Java
Graphics Library	:	JavaFX Canvas
IDE Used	:	IntelliJ IDEA / Eclipse / NetBeans

CHAPTER 3
MODULE DESCRIPTION

The Smart Focus Timer system is divided into three main modules: Timer Management, Task Management, and Media & Visualization. Each module performs a distinct function that contributes to creating an efficient, motivating, and user-friendly productivity environment.

1. TIMER MANAGEMENT MODULE

The Timer Management module forms the core of the Smart Focus Timer. It allows users to start, pause, and reset customizable focus sessions based on the Pomodoro technique. Users can set preferred focus and break durations according to their workflow. The timer is implemented using JavaFX's animation and Canvas APIs, providing real-time circular progress visualization. This module ensures smooth operation through responsive controls and sound alerts for session transitions, helping users maintain consistent focus and structured work intervals.

2. TASK MANAGEMENT MODULE

The Task Management module enables users to organize their work efficiently by adding, editing, and deleting tasks. Each task can represent a specific goal or activity to be completed during focus sessions. The module provides a clean and minimal interface for managing daily or session-based tasks. Data persistence ensures that tasks remain available even after restarting the application. This module promotes productivity and goal tracking, allowing users to prioritize tasks and stay organized throughout their focus cycles.

3. MEDIA & VISUALIZATION MODULE

The Media & Visualization module enhances the user experience by integrating visual and auditory feedback. Using JavaFX MediaPlayer, the system allows users to play built-in lofi background music, creating a calm and motivating atmosphere during sessions. Additionally, the circular progress visualization dynamically reflects session progress in real time, offering a clear visual cue of time remaining. Together, these features make focus sessions more engaging and help users maintain concentration in a distraction-free environment.

CHAPTER 4
SAMPLE CODE 1

```
import javafx.animation.KeyFrame;
import javafx.animation.Timeline;
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.scene.media.Media;
import javafx.scene.media.MediaPlayer;
import javafx.scene.paint.Color;
import javafx.scene.shape.ArcType;
import javafx.stage.Stage;
import javafx.util.Duration;

import java.io.File;

public class Main extends Application {

    private int focusMinutes = 25;
    private int secondsLeft;
    private Timeline timeline;

    private Label timerLabel;
    private Button startBtn, pauseBtn, resetBtn;

    private ListView<CheckBox> taskListView;
    private Canvas clockCanvas;
    private double canvasSize = 150;

    private MediaPlayer mediaPlayer;
    private ComboBox<String> musicSelector;
    private Button playMusicBtn, stopMusicBtn;

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Smart Focus Timer");
```

```
        // --- Focus Time Input ---
        TextField focusTimeField = new TextField("25"); // default 25 minutes
        focusTimeField.setPrefWidth(60);

        Button setFocusBtn = new Button("Set Focus Time");
        setFocusBtn.setOnAction(e -> {
            try {
                int value = Integer.parseInt(focusTimeField.getText());
                if (value <= 0) throw new NumberFormatException();
                focusMinutes = value;
                resetTimer();
            } catch (NumberFormatException ex) {
                Alert alert = new Alert(Alert.AlertType.ERROR, "Please enter a valid positive
number!");
                alert.show();
            }
        });

        HBox selectorBox = new HBox(10, new Label("Focus time (min):"), focusTimeField,
setFocusBtn);
        selectorBox.setPadding(new Insets(10));
        selectorBox.setStyle("-fx-alignment: center;");

        // --- Timer Label ---
        timerLabel = new Label(formatTime(focusMinutes * 60));
        timerLabel.setStyle("-fx-font-size: 36px; -fx-text-fill: #ffffff;");

        // --- Clock Canvas ---
        clockCanvas = new Canvas(canvasSize, canvasSize);
        drawClock(0);

        VBox timerBox = new VBox(clockCanvas, timerLabel);
        timerBox.setPadding(new Insets(20));
        timerBox.setStyle("-fx-alignment: center; -fx-spacing: 10;");

        // --- Buttons ---
        startBtn = new Button("Start");
        pauseBtn = new Button("Pause");
        resetBtn = new Button("Reset");
        HBox controlsBox = new HBox(10, startBtn, pauseBtn, resetBtn);
        controlsBox.setPadding(new Insets(10));
        controlsBox.setStyle("-fx-alignment: center;");

        // --- Task List ---
```

```
        taskListView = new ListView<>();
        taskListView.setStyle("-fx-control-inner-background: #2b2b2b; -fx-text-fill: #ffffff;");

        Button addTaskBtn = new Button("Add Task");
        Button editTaskBtn = new Button("Edit Task");
        Button deleteTaskBtn = new Button("Delete Task");
        HBox taskButtons = new HBox(10, addTaskBtn, editTaskBtn, deleteTaskBtn);
        taskButtons.setPadding(new Insets(5));
        taskButtons.setStyle("-fx-alignment: center;");

        VBox tasksBox = new VBox(new Label("Tasks"), taskListView, taskButtons);
        tasksBox.setPadding(new Insets(10));
        tasksBox.setStyle("-fx-background-color: #2b2b2b; -fx-text-fill: #ffffff;");
        tasksBox.setSpacing(5);

        // --- Music Selection (only lofi tracks) ---
        musicSelector = new ComboBox<>();
        musicSelector.getItems().addAll("lofi1.mp3", "lofi2.mp3", "lofi3.mp3");
        musicSelector.setValue("lofi1.mp3");

        playMusicBtn = new Button("Play Music");
        stopMusicBtn = new Button("Stop Music");
        HBox musicBox = new HBox(10, new Label("Music:"), musicSelector, playMusicBtn,
stopMusicBtn);
        musicBox.setPadding(new Insets(10));
        musicBox.setStyle("-fx-alignment: center;");

        // --- Main Layout ---
        BorderPane root = new BorderPane();
        root.setTop(new VBox(selectorBox, timerBox, musicBox));
        root.setCenter(tasksBox);
        root.setBottom(controlsBox);
        root.setStyle("-fx-background-color: #1e1e1e;");

        Scene scene = new Scene(root, 450, 600);
        primaryStage.setScene(scene);
        primaryStage.show();

        // --- Timer Timeline ---
        secondsLeft = focusMinutes * 60;
        timeline = new Timeline(new KeyFrame(Duration.seconds(1), e -> tick()));
        timeline.setCycleCount(Timeline.INDEFINITE);

        // --- Button Actions ---
```

```
    startBtn.setOnAction(e -> {
        playSound("session_start.wav");
        timeline.play();
    });
    pauseBtn.setOnAction(e -> {
        playSound("session_start.wav");
        timeline.pause();
    });
    resetBtn.setOnAction(e -> {
        playSound("session_start.wav");
        resetTimer();
    });

    addTaskBtn.setOnAction(e -> addTask());
    editTaskBtn.setOnAction(e -> editTask());
    deleteTaskBtn.setOnAction(e -> deleteTask());

    playMusicBtn.setOnAction(e -> playMusic(musicSelector.getValue()));
    stopMusicBtn.setOnAction(e -> stopMusic());
}

private void tick() {
    if (secondsLeft > 0) {
        secondsLeft--;
        timerLabel.setText(formatTime(secondsLeft));
        drawClock(1 - ((double) secondsLeft / (focusMinutes * 60)));
    } else {
        timeline.pause();
        Alert alert = new Alert(Alert.AlertType.INFORMATION, "Focus session over!");
        alert.show();
    }
}

private void resetTimer() {
    timeline.stop();
    secondsLeft = focusMinutes * 60;
    timerLabel.setText(formatTime(secondsLeft));
    drawClock(0);
}

private String formatTime(int totalSeconds) {
    int minutes = totalSeconds / 60;
    int seconds = totalSeconds % 60;
    return String.format("%02d:%02d", minutes, seconds);
```

```
}

private void drawClock(double progress) {
    GraphicsContext gc = clockCanvas.getGraphicsContext2D();
    gc.clearRect(0, 0, canvasSize, canvasSize);

    gc.setStroke(Color.GRAY);
    gc.setLineWidth(8);
    gc.strokeOval(5,5,canvasSize-10, canvasSize-10);

    gc.setStroke(Color.LIMEGREEN);
    gc.setLineWidth(8);
    gc.strokeArc(5,5,canvasSize-10,canvasSize-10,90,-360*progress,ArcType.OPEN);
}

// --- Task Methods ---
private void addTask() {
    TextInputDialog dialog = new TextInputDialog();
    dialog.setHeaderText("Add a new task");
    dialog.setTitle("Add Task");
    dialog.setContentText("Task:");
    dialog.showAndWait().ifPresent(name -> {
        CheckBox cb = new CheckBox(name);
        cb.setStyle("-fx-text-fill: #ffffff;");
        taskListView.getItems().add(cb);
    });
}

private void editTask() {
    CheckBox selected = taskListView.getSelectionModel().getSelectedItem();
    if (selected != null) {
        TextInputDialog dialog = new TextInputDialog(selected.getText());
        dialog.setHeaderText("Edit task");
        dialog.showAndWait().ifPresent(newName -> selected.setText(newName));
    }
}

private void deleteTask() {
    int selectedIndex = taskListView.getSelectionModel().getSelectedIndex();
    if (selectedIndex >= 0) {
        taskListView.getItems().remove(selectedIndex);
    }
}
```

```
    // --- Sound Methods ---
    private void playSound(String filename) {
        try {
            Media sound = new Media(new File("src/resources/sounds/" +
filename).toURI().toString());
            MediaPlayer mp = new MediaPlayer(sound);
            mp.play();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    // --- Music Methods ---
    private void playMusic(String filename) {
        stopMusic();
        try {
            Media media = new Media(new File("src/resources/sounds/" +
filename).toURI().toString());
            mediaPlayer = new MediaPlayer(media);
            mediaPlayer.setCycleCount(MediaPlayer.INDEFINITE);
            mediaPlayer.play();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void stopMusic() {
        if (mediaPlayer != null) {
            mediaPlayer.stop();
        }
    }

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

CHAPTER 5

SCREENSHOTS

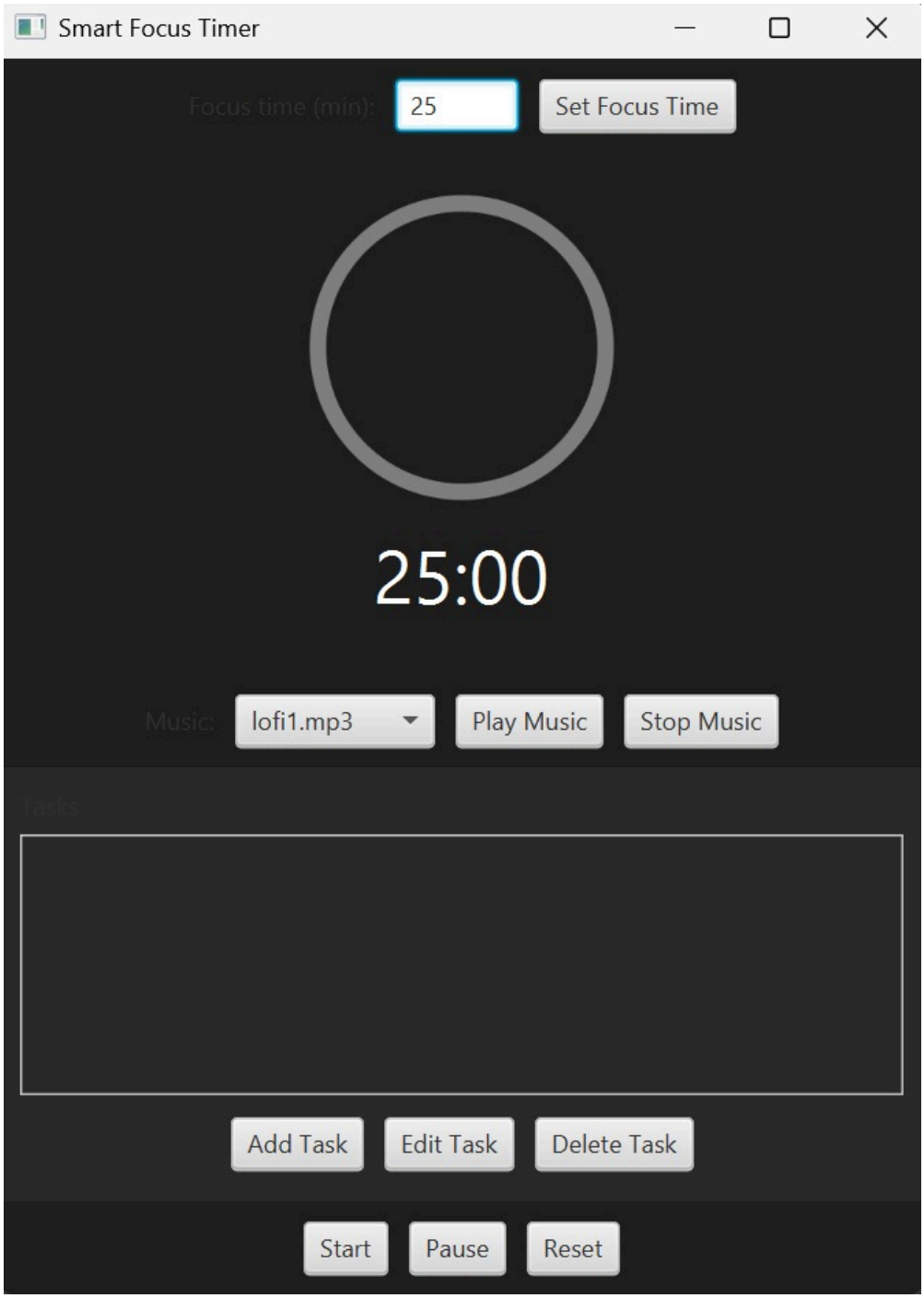

Fig 5.1 User Interface

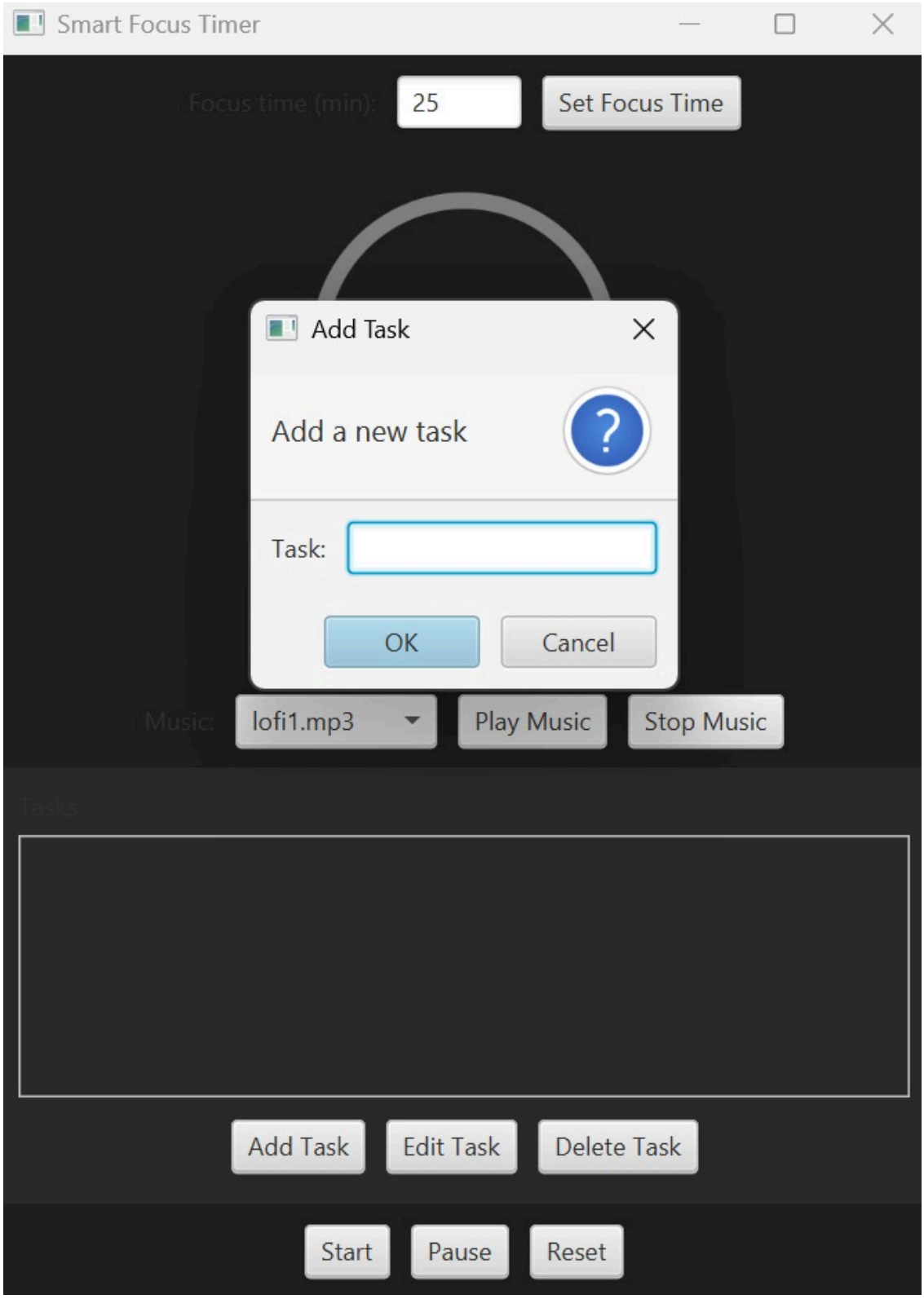

Fig 5.2 Add Tasks

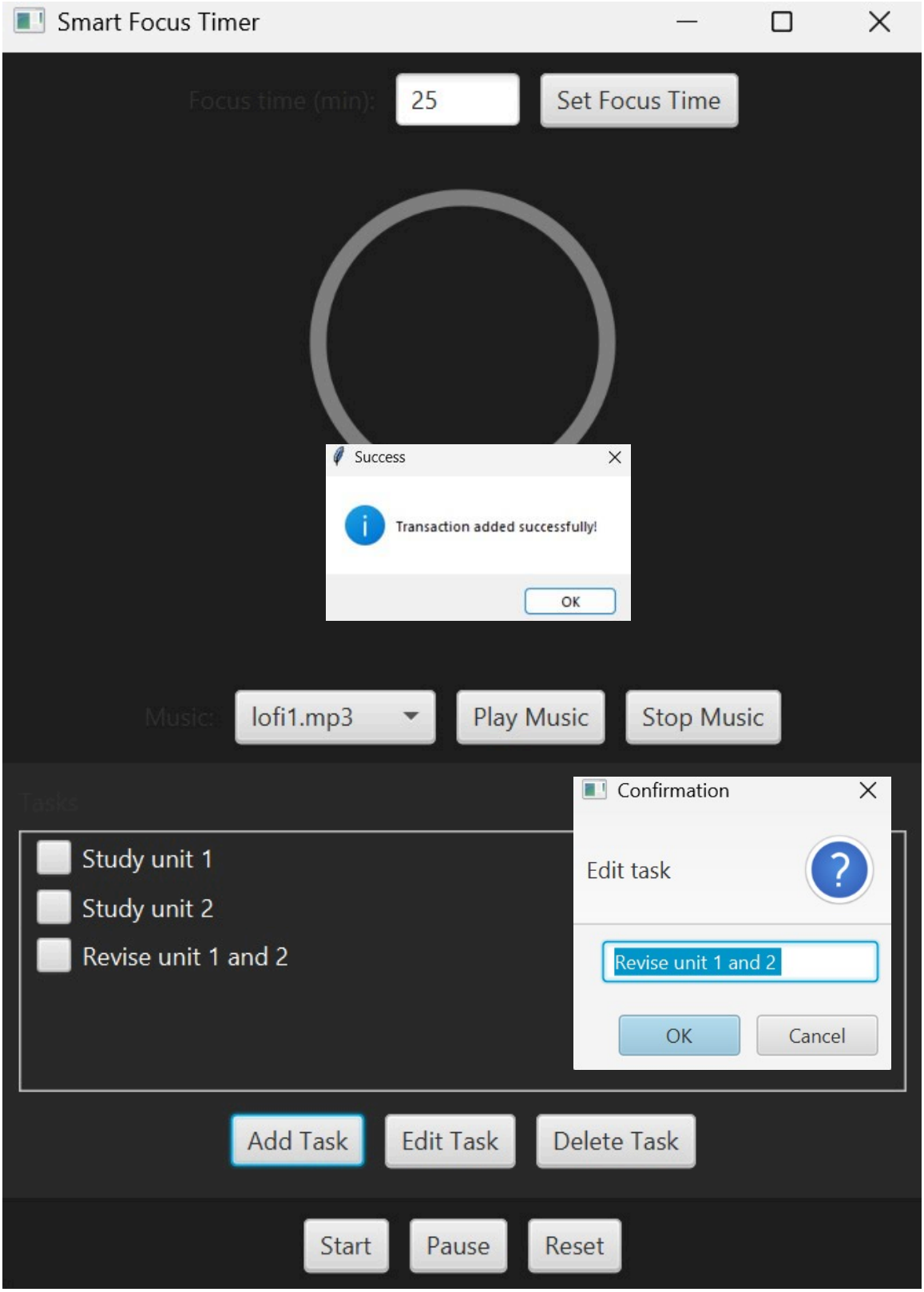

Fig 5.3 Edit & DeleteTasks

CHAPTER 6
CONCLUSION AND FUTURE ENHANCEMENT

The Smart Focus Timer offers an effective and engaging solution for improving productivity through time management and focus enhancement. With its customizable Pomodoro-style timer, task organization features, and built-in lofi music player, it creates an ideal environment for maintaining concentration and completing goals efficiently. The system's minimalist design and real-time visual feedback make focus sessions both structured and enjoyable. In the future, the application can be enhanced with features such as cloud-based task synchronization, AI-driven productivity insights, mobile app integration, and session history tracking, making it an even more powerful and intelligent productivity companion.

REFERENCES

1. https://docs.oracle.com/javase/8/javafx/api/
2. https://openjfx.io/?utm_source=chatgpt.com
3. https://www.geeksforgeeks.org/javafx-tutorial/
4. https://www.tutorialspoint.com/javafx/index.htm
5. https://www.geeksforgeeks.org/javafx-mediaplayer-class/
6. https://www.javatpoint.com/javafx
7. https://docs.oracle.com/javase/8/javafx/graphics-tutorial/javafx-3d-graphics.htm
8. https://www.geeksforgeeks.org/javafx-canvas-class/
9. https://code.makery.ch/library/javafx-tutorial/
10. https://www.oracle.com/java/technologies/javase/javafx-overview.html