# INTRODUCTION

## Background

Being productive and leaping towards perfection always been a topic of high regard in the present world. We students always work to change for better. Several decades into formal education and we have realized that some practices are better than the other in academics. Practices like to-do listing, session tracking, task scheduling, pomodoro technique, music studying etc. are proven to be more effective. Usage of computer to facilitate and implement these practices is becoming more relevant due to technological reach. So, we aim to compile these practices into a single GUI application using C++ and SFML.

## Motivation

Procrastination is a damaging trait which involves ignoring an unpleasant, but likely more important task, in favor of one that is more enjoyable or easier. It can lead to reduced productivity and cause us to miss out on achieving our goals. Poor organization of tasks can lead to procrastination. Using prioritized To-do lists, creating effective schedules, breaking down a work into time intervals separated by short breaks and working on it etc. are few effective measures to stay organized. In our project we intend to integrate these practices into a single application, which would be of great help to not only us but to many scholars worldwide.

## Problem Definition

All of us, team members, in some way or the other have used many applications for staying organized. We all realized that a single application does not facilitate every technique we wished for. Most applications were available in mobile phones only and some required us to make a payment for accessing the full-fledged application. So, there was a collective need for an application that compiles all the practices that we wished for and in the meantime eradicating all the other problems that we encountered.

## Project Objectives

This project aims to full the following objective:

* To develop an application that compiles to-do list, session tracker, study planner and pomodoro timer using C++ and SFML.

**1.5 Project Application**

The project we purpose put forwards following application:

* Task organization and completion – Prioritized to-do list and session tracker are of great help in organizing daily tasks. Moreover, investing periodic efforts using pomodoro timer sets us in a track for completing that task at the very least. At the end of the day we are organized and we accomplish many things.
* Self-tracking – Session tracker helps us keep track of our involvement in a specific work and then invest efforts accordingly. Similarly, the to-do list reminds us of the targets we’ve set for the day and also the targets we’ve accomplished.

## 1.6 Scope of Project

Productivity tools are handy to everybody worldwide. It is not that this application is limited within scholars only. Everybody willing to stay organized and focused can make a good use of it. On the down side, it is a desktop application and it cannot be used in mobile phones.

# ABSTRACT

# 4.1 Pomodoro timer

Pomodoro timer makes the use of user interactive dial like interface using which the user will be able to set the time for a pomodoro session. The timer starts when the user presses the play button and it ends when user completes the session or user chooses to end the session before the timer completes. User will be rewarded points as per the length of the session provided that they complete it.

## 4.1.1 Basic Interface

Filler Circles

Knob Circle

Main Circle

Time display

Play Button

Reference Point

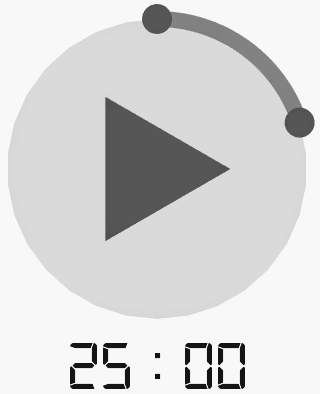


Figure 1: Basic Interface

For setting the timer user will be provided with an interactive dial like interface. The dial consists of 3 main components namely: a main circle, filler circles and a knob circle. By dragging the knob circle users will be able to adjust the timer as per their liking which is displayed just below the dial. Meanwhile, the filler circles visually show the point up to which the knob circle is moved from the reference point.

## 4.1.2 Filler circles

sf::Vector2f is a utility template class for manipulating 2-dimensional vectors defined with two coordinates (x and y) which are its data members that can be accessed directly. sf::Vector3f is similar to sf::Vector2f except for having 3 coordinates(x, y and z).

In the beginning, all the coordinates on the outline of the main circle is calculated using general equation of circle and stored in a C++ vector having datatype as sf::Vector2f.

Then, all the positions in the outline of the main circle where the knob circle would set time of positive whole numbers i.e., 1,2,3 … 120 are calculated and stored in a C++ vector having datatype of sf::Vector3f where x is time and, y and z are the coordinates of the points. Similar process is repeated for to find out positions where knob circle would set the time value in minutes is exactly divisible by five i.e., 0,5,10,15, … , 120 is also found out.

Later, by iterating through vector containing all positions on the outline of the main circle, filler circle is drawn in every iterated position. Iteration stops when the iterated value is equal to the coordinates of the knob circle.

# 4.1.3 Knob Mechanism

Initially the knob circle is placed at the position where the set time for the timer is 25 minutes.

Upon dragging, the knob circle is positioned along the outline of the main circle with respect to the position of the mouse pointer. The coordinates of that position are calculated by using a simple geometric calculation that involves the intersection of the line joining center of the main circle and the outline of the main circle itself which is illustrated below.

When user releases the mouse and stops dragging the knob, its position is set to a point where the timer set was last exactly divisible by 5.

## 4.1.5 Time Evaluations

Time that is set by the user by dragging the knob, is calculated mathematically by evaluating time as a function of degrees. Position of the knob determines the time that will be set for the timer. When the knob is moved to its final limit i.e., when the knob makes full rotation around the main circle making an angle of 360 degrees, the timer will be set to 120 minutes. However, the user will only be enabled to set the time in the multiples of five beginning from 25 minutes to 120 minutes. Calculations involved are illustrated below.

## 4.1.6 Timer mechanism

SFML has an inbuilt class *sf::Clock*. When an object of the class is created the clock starts. The time elapsed after the object is created can be accessed by using *getElapsedTime()* method. To get the elapsed time in seconds we use *getElapsedTime().asSeconds()* method. Moreover, the clock can be restarted using *restart()* method.

When the user starts timer by clicking the play button, the clock is restarted and the timer starts ticking. Initially the timer begins at the time set by the user and eventually it decreases to zero (i.e., 00:00) provided that user doesn’t end the session half way by pressing the stop button. Time that the timer displays at any instant is calculated by subtracting elapsed time from the time set by the user in seconds. Meanwhile, the filler circles give the visual representation of the time remaining.