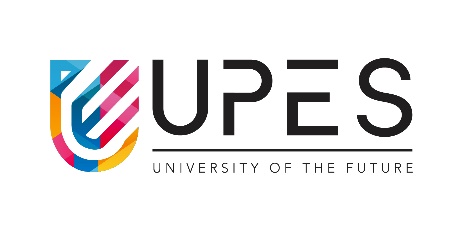
# PROJECT REPORT

**ON**

**Topic Name: The 9-5 Alarm**

|  |  |  |
| --- | --- | --- |
|  | **Submitted By** |  |
|  | SRUJAN V  BATCH – 11  500106950  ***Under the guidance of***  **Ms. Gaytri Bakshi**  (Assistant Professor)  Department of Computer Science |  |



**Department of Cybernetics**

**School of Computer Science**

# UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

**Dehradun-248007**

**April 2023**

# Index

**Contents Page Number**

**Chapter 1: Introduction** 3

**Chapter 2: Problem Statement** 4

**Chapter 3: Objectives** 4

**Chapter 4: Methodology** 5

**Chapter 5: Literature** 7

**Chapter 6: System Requirements** 9

**Chapter 7: Conclusion** 10

**Chapter 8: PERT TABLE** 11

**Chapter 9: References** 13

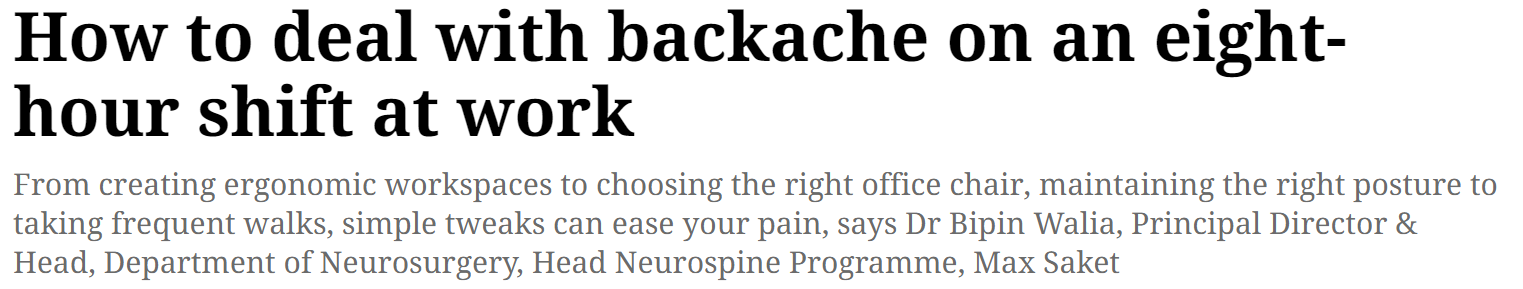
# Chapter 1: Introduction

Ever since the globe was hit with the pandemic in 2020, most of the companies shifted their on-site working systems to a new trend we all have heard of called – ‘Work from Home’, an idea which is being adopted by most firms to allow its employees to work from the comfort of their home. If you think about it, these working methods have nothing much of a difference – working from 9 to 5. Although software engineers who work 9 to 5 receive an amazing pay, they are bound to live in a monotonous work culture. They are glued to their seats and their eyes are fixed on the computer screen constantly exposed to the blue-light display. This takes a toll on their physical health.

There are numerous articles available online which address this issue -

**Text

Description automatically generated**

****

Working an 8-hour shift can be the cause of dry eyes and back pain - because of staring at the screen continuously for long hours every day whilst having bad posture. Thousands of employees suffer from these symptoms because they tend to neglect their physical wellbeing as they are burdened with a lot of work. However, there is a solution to every problem and for this one –

1) Ensure to drink plenty of water through the course of the office hours to stay hydrated.

2) Take a short break every 30 – 45 minutes to look away from the screen and relax your eyes.

3) Getting off the seat, going for a walk with your colleagues around the block, stretching or even checking on your fellow workers can help you get your mind off work, relaxing your back muscles and maintaining healthy posture.

But we are so engrossed in our work that we forget to make such changes, only if there was a way to remind ourselves at regular intervals of time so that we don’t forget. That is where this program swoops in. It helps create health awareness among people who work at a desk for 8 hours every day. It is user friendly and prompts the user to take a break at regular intervals from their devices.

# Chapter 2: Problem Statement

Coding/Programming is a fun tool that has been used for solving various problems not just in the field of computer science but also in other domains as well. Using the concepts of programming, I have come up with a code that aims to solve a real-life problem. This code rings an alarm at regular time intervals as desired by the user prompting him/her to take a break from their work for a few minutes.

**The Problem Statement for the code –**

To create health awareness among people working a job from 9 a.m. to 5 p.m. (or longer).

# Chapter 3: Objectives

**Objective of the code –**

1) To prompt users to take a short break from their work instead of working for long hours continuously.

2) To keep and maintain an active log of the user’s activity which is stored permanently on the device and is accessible at the user’s will.

# Chapter 4: Methodology

Before we dive into the actual working of the program, let us look at what are the modules and functions which have been used to make this code functional.

1) **Pygame** - Pygame is a Python library designed for creating video games. It provides functionality for creating graphics, handling user input, playing sounds and music, and other features commonly found in video games. Pygame is built on top of the SDL library.

\*SDL (Simple DirectMedia Layer) is a cross-platform development library designed to provide low-level access to audio, keyboard, mouse, joystick, and graphics hardware via OpenGL and Direct3D. It is used in game development, multimedia, and other applications that require access to low-level hardware functionality.

2) **mixer** - The Pygame Mixer module is a part of the Pygame library used for loading and playing audio files in different formats like WAV, MP3, OGG, MIDI, etc. It provides functions for loading, playing, pausing, stopping, and fading in/out of audio files. It also allows you to set the volume and channels for each sound. Here are the five basic steps I applied in the code to play audio –

i) Starting the mixer: **mixer.init()** by this we can initialize the mixer module from **pygame** and once it has been initialized, we can load and play audio files.

ii) Loading the song: **mixer.music.load(audio.mp3)** - a method call from the Pygame library's mixer module, where `**audio.mp3**` is a variable that contains the filename or path of an MP3 audio file. The `**load()`** method is used to load an audio file into Pygame’ s mixer.

iii) Setting the volume: **mixer.music.set\_volume(0.6)** is a method used in the Pygame mixer module to set the volume of the currently loaded music file. The volume can be set between 0.0 (silent) and 1.0 (full volume) inclusive. In the above example, `mixer.music.set\_volume(0.6)` sets the volume to 60% of the maximum volume of the device.

iv) Playing the audio: **mixer.music.play()** is a method in the Pygame mixer module that starts playing the currently loaded music file. When called, it begins playing the music from the beginning of the file. If there is already music playing, it will be stopped and replaced by the new file.

v) Stopping the audio: **mixer.music.stop()** stops the playback of the current music file being played by **`mixer.music.play()`.**

3) **time** – it is a module in Python that provides various time-related functions. It allows you to measure time, sleep for a specific duration, and get the current time in different formats. Some of the functions which I have used in this program from `time` module are:

- **time()**: Returns the current time in seconds since the Epoch (January 1st, 1970, 00:00:00 UTC).

- **ctime()**: Converts a time in seconds to a string representing a local time. The time argument is optional and defaults to the current time.

- **strftime()**: Converts a struct\_time to a string according to a format specification.

# 

# Chapter 5: Literature

In this chapter of the report, we will understand the functions used in the code and how they perform, basically, the whole working of the whole code. In the initial lines of the code, I imported the mixer module from the pygame library and imported the time module to get access to the in-built functions of each module.



The first function defined in the code is for ringing the alarm.

Text

Description automatically generated

This is a Python function that plays music using the Pygame library's mixer module. It takes two arguments, "music" and "volume", where "music" is the path to the audio file to be played, and "volume" is the volume at which the music should be played. The function starts by initializing the mixer and loading the music file using the `mixer.music.load()` method. It then sets the volume of the music using the `mixer.music.set\_volume()` method and plays the music using the `mixer.music.play()` method. The function then enters an infinite loop where it asks the user for input to stop the music. If the user inputs "yes", the music is stopped, and the function returns False. If the user inputs "q", the music is stopped, and the function returns True. If the user inputs anything else, the message "wrong input!!" is printed to the console.

The next function is responsible for creating a log of the user and storing it in the device.

Text

Description automatically generated

It takes two arguments, "**msg**" and "**name**", where "**msg**" is the message to be logged, and "**name**" is the name of the log file to which the message should be written based on the activity which is later declared in the main code. The function starts by opening the log file in append mode using the `**with open(name+".txt", "a") as file:` statement**. The `**with**` statement ensures that the file is closed after the block is executed. The `a` mode appends the message to the end of the file if it already exists and creates the file if it does not exist. The function then writes the message to the file using the `**file.write()`** method. The message includes the current time, which is obtained using the `**time.ctime()`** function. This gives us the current time it has been recorded. Finally, the function prints the message "**Log saved!!**" to the console to indicate that the message has been successfully logged to the file.

The last function in the code is responsible for displaying the contents of a log file.

Text

Description automatically generated

It takes in a string argument called `**log**`, which is the name of a file that contains logged information. The function then attempts to open and read the contents of the file and prints them to the console. If the file does not exist or there is an error in opening it, the function prints a message indicating that the file does not exist. Overall, the `**disp**` function is designed to display the contents of a file to the user, and handle errors gracefully if the file does not exist.

The main code makes use of all the functions defined and explained above to implement the whole program.

**Chapter 6: System Requirements**

Software Requirements

* Operating System: Windows 11/10/8/7 (32-bit or 64-bit)
* Software: VS Code (with pygame installed)

\*To install pygame, open **cmd** and type: ‘pip install pygame’, without which the program will not run successfully.

* Language used: Python.

Hardware Requirements

* Processor: Dual Core 2.7 GHz or better
* RAM: 512 MB or higher
* Disk Space: 512 MB

**Chapter 7: Conclusion**

This code reminds the user to take breaks while working and logs the breaks they take. The program asks for the user's name to create a file in their name, and then offers a menu with options - to start an alarm for taking breaks, open previous activity, and to exit the program.

If the user chooses to start the alarm, they are asked for the time intervals for drinking water and relaxing their eyes. The program then plays an audio file to remind the user to take a break after the specified intervals and logs the break they took. The program will continue to remind the user to take breaks until they exit the program or until it is no longer office hours. If the user chooses to open previous logs, they are given a sub-menu to choose whether they want to view their logs for drinking water breaks, logs for relaxing their eyes, or to exit the sub-menu. If the user chooses to exit the program, the program will end.

Overall, this program is used for reminding people to take breaks while working and keeping track of the breaks they take.

**Chapter 8: PERT Table**

[Day 1] Brainstorming Ideas

Duration: 15 minutes

[Day 1] Project Planning & Downloading Audio Files

Duration: 6-5 hours

[Day 2] Learning Libraries and Modules

Duration: 2-3 hours

[Day 3 & 4] Coding

Duration: 2 days

[Day 5 & 6] Report Making & PPT

Duration: 6 hours

[Day 7] Testing with other users,

Improvising and Final

Code Deployment

**Chapter 9: References**

<https://www.geeksforgeeks.org/python-time-module/> - To learn about the time module.

<https://www.geeksforgeeks.org/python-playing-audio-file-in-pygame/> - To learn about the pygame library and mixer module.

<https://chat.openai.com/> - for error rectification and improvising.

<https://www.youtube.com/> - used it for finding audio files and then downloaded it.

# 

# 