Word Scrabble

```
import random
import pygame


class SoundToggle:
    def __init__(self, text="Sound", state=True):
        self.name = text
        self.state = state
        self.font = pygame.font.SysFont('Lato', 24, bold=False)

    def create(self, parent):
        if not self.state:
            icon = pygame.image.load('images/toggle-off.png')
            text = self.font.render(self.name, True, (0, 0, 0))
            parent.blit(icon, (825, 15))
            parent.blit(text, (820, 6))
        else:
            icon = pygame.image.load('images/toggle-on.png')
            text = self.font.render(self.name, True, (0, 0, 0))
            parent.blit(icon, (825, 15))
            parent.blit(text, (820, 6))

    def toggle(self, sounds):
        if not self.state:
            self.state = True
            sounds.set_volume(1.0)
        else:
            self.state = False
            sounds.set_volume(0.0)


class Button:
    def __init__(self, x, y, width, height, text, font_size=20):
        self.x = x
        self.y = y
        self.width = width
        self.height = height
        self.text = text
        self.font = pygame.font.SysFont('Arial', font_size, bold=True)
        self.bg = (0, 255, 0)
        self.fg = (0, 0, 0)
        self.highlight_color = (0, 230, 0)
        self.highlighted = False

    def create(self, parent):
        if self.highlighted:
            self.bg = self.highlight_color
        else:
            self.bg = self.bg
        pygame.draw.rect(parent, self.bg, rect=(self.x, self.y, self.width,
self.height), border_radius=10)
        text = self.font.render(self.text, True, self.fg)
        text_rect = text.get_rect()
        text_rect.center = self.x + self.width // 2, self.y + self.height
// 2
        parent.blit(text, text_rect)
```

```
class Letter:
    def __init__(self, let):
        self.letter = let


def submit(word):
    for letter in word:
        word = "".join(letter)
    #     check eng_us dictionary
    us_dict = enchant.Dict("en_US")
    if us_dict.check(word):
        #     and word not in list
        # Play correct sound
        print(f"{word} Found")
    else:
        # Play incorrect sound
        print(f"{word} not found")


# Do only when there is a letter in the field
def remove_from_board(count, x, y):
    pass
    try:
        if temp_num == 7:
            x = 1
        elif temp_num == 6:
            x, y = 220, 98
        elif temp_num == 5:
            x, y = 262, 98
        elif temp_num == 4:
            if (281 <= x <= 350) and (96 <= y <= 165):
                mouse_click1_sound.play()
                temp = clicked_objects[0]
                text_surfaces.pop(0)
                clicked_objects.pop(0)
            elif (361 <= x <= 429) and (96 <= y <= 165):
                mouse_click1_sound.play()
                temp = clicked_objects[1]
                text_surfaces.pop(1)
                clicked_objects.pop(1)
            elif (441 <= x <= 509) and (96 <= y <= 165):
                mouse_click1_sound.play()
                temp = clicked_objects[2]
                text_surfaces.pop(2)
                clicked_objects.pop(2)
            elif (522 <= x <= 589) and (96 <= y <= 165):
                mouse_click1_sound.play()
                temp = clicked_objects[3]
                text_surfaces.pop(3)
                clicked_objects.pop(3)
        elif temp_num == 3:
            if (320 <= x <= 390) and (96 <= y <= 165):
                mouse_click1_sound.play()
                temp = clicked_objects[0]
                text_surfaces.pop(0)
                clicked_objects.pop(0)
            elif (400 <= x <= 470) and (96 <= y <= 165):
                mouse_click1_sound.play()
                temp = clicked_objects[1]
                text_surfaces.pop(1)
```

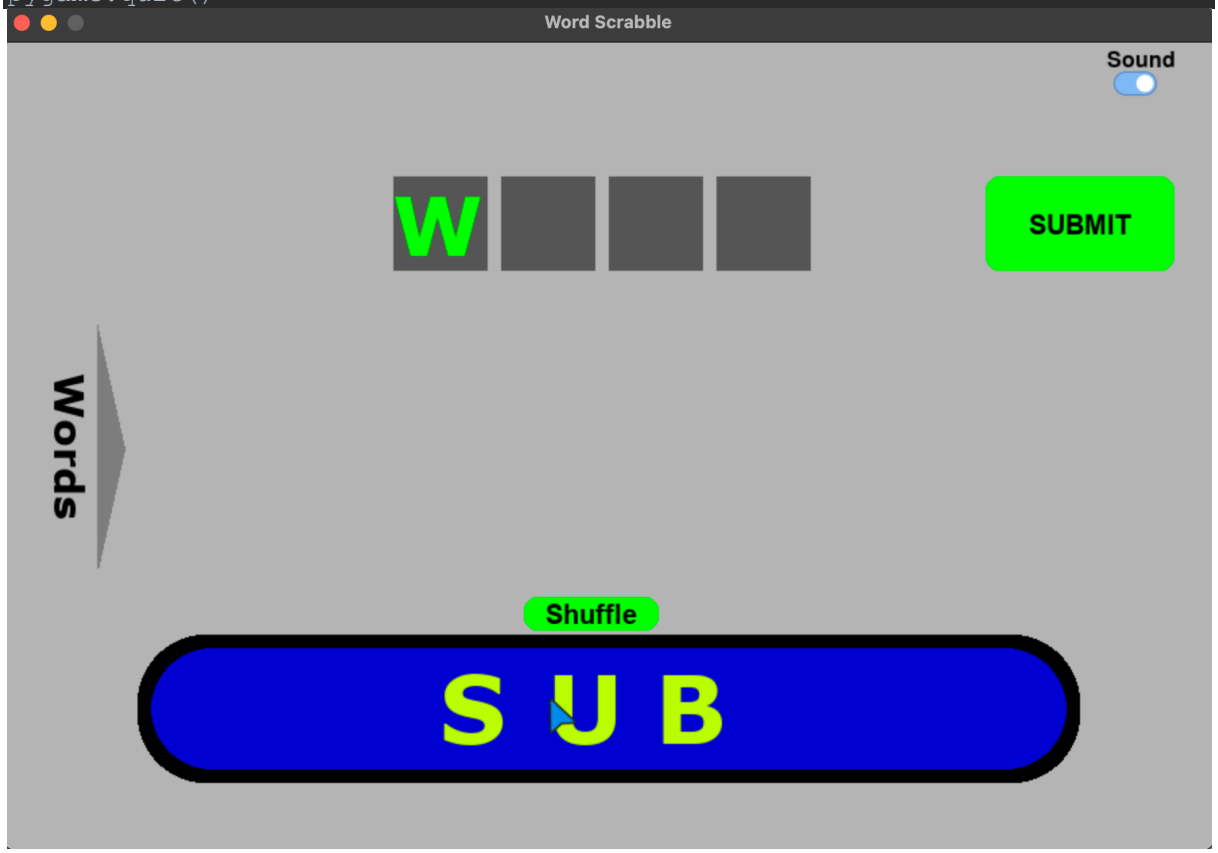
```
                clicked_objects.pop(1)
            elif (480 <= x <= 550) and (96 <= y <= 165):
                mouse_click1_sound.play()
                temp = clicked_objects[2]
                text_surfaces.pop(2)
                clicked_objects.pop(2)
        else:
            return count
        # temp = clicked_objects[0]
        # clicked_objects.pop(0)
        temp = Letter(temp)
        letters_object.append(temp)
        count += 1
        return count
    except IndexError as e:
        print(e)
        print(letters_object)
        print(count)


def add_to_board(showing, clicked):
    w = 900 // 2
    if temp_num == 7:
        xCor, yCor = 183, 98
    elif temp_num == 6:
        xCor, yCor = 220, 98
    elif temp_num == 5:
        xCor, yCor = 262, 98
    elif temp_num == 4:
        xCor, yCor = 302, 98
    elif temp_num == 3:
        xCor, yCor = 340, 98
    for i in range(temp_num):
        board = pygame.draw.rect(window, (85, 85, 85), rect=(w-
(40*temp_num), 100, 70, 70))
        w += 80
    submit = Button(730, 100, 140, 70, 'SUBMIT')
    submit.create(window)
    i = 0
    for text in text_surfaces:
        if clicked_objects[i] == 'I':
            xCor += 8
        elif clicked_objects[i] == 'W':
            xCor -= 13
        elif clicked_objects[i] == 'Q':
            yCor -= 5
        elif clicked_objects[i] == 'J':
            xCor += 3
        elif clicked_objects[i] == 'M':
            xCor -= 4
        window.blit(text, (xCor, yCor))
        if clicked_objects[i] == 'I':
            xCor -= 8
        elif clicked_objects[i] == 'W':
            xCor += 13
        elif clicked_objects[i] == 'Q':
            yCor += 5
        elif clicked_objects[i] == 'J':
            xCor -= 3
        elif clicked_objects[i] == 'M':
            xCor += 4
```

```
        i += 1
        xCor += 80
    font = pygame.font.SysFont('Verdana', 30, bold=True)
    words = font.render('Words', True, (0, 0, 0))
    rotate = pygame.transform.rotate(words, -90)
    rotated_rect = rotate.get_rect()
    rotated_rect.center = (50, 300)
    window.blit(rotate, rotated_rect)
    # Set the arrow color
    arrow_color = (125, 125, 125)
    # Set the points for the body of the arrow
    # body_points = [(40, 30), (40, 30), (40, 220), (40, 230)]
    # Set the points for the head of the arrow
    head_points = [(70, 390), (90, 300), (70, 210)]
    # Draw the body of the arrow
    body = pygame.draw.polygon(window, arrow_color, head_points)
    # Draw the head of the arrow
    head = pygame.draw.polygon(window, arrow_color, head_points, width=0)
    if showing:
        correct_list_rect = pygame.draw.rect(window, (155, 155, 155),
rect=(100, 200, 700, 200), border_radius=10)


def putletters(X, Y, pos):
    letterHolder = pygame.draw.rect(window, (0, 0, 0), rect=(100, 440, 700,
110), border_radius=50)
    letterHolder = pygame.draw.rect(window, bg, rect=(110, 450, 680, 90),
border_radius=50)
    letterHolder.center = (900 // 2, 500)
    counter = 0
    for letter in letters_object:
        # if pos is None:
        text = font.render(letter.letter, True, fg, bg)
        # else:
        #     if counter == pos:
        #
        #         text = font.render(letter.letter, True, (0, 255, 0), bg)
        #     else:
        #         text = font.render(letter.letter, True, fg, bg)
        text_rect = text.get_rect()
        text.convert_alpha()
        text.set_alpha(255)
        text_rect.center = (900 // 2, 500)
        window.blit(text, (X-(34.5*len(letters_object)), Y))
        X += 80
        counter += 1


def check_pos_clicked(count, mousex, mousey):
    pos = None
    ps_clicked = None
    if count == 1:
        for i in range(1):
            if (text_pos_width_1[i][0] <= mousex <= text_pos_width_1[i][1])
and text_pos_height[0] <= mousey <= text_pos_height[1]:
                mouse_click3_sound.play()
                pos = letters_object[i].letter
                ps_clicked = i
    if count == 2:
        for i in range(2):
            if (text_pos_width_2[i][0] <= mousex <= text_pos_width_2[i][1])
```

```
and text_pos_height[0] <= mousey <= text_pos_height[1]:
                mouse_click3_sound.play()
                pos = letters_object[i].letter
                ps_clicked = i
    if count == 3:
        for i in range(3):
            if (text_pos_width_3[i][0] <= mousex <= text_pos_width_3[i][1])
and text_pos_height[0] <= mousey <= text_pos_height[1]:
                mouse_click3_sound.play()
                pos = letters_object[i].letter
                ps_clicked = i
    elif count == 4:
        for i in range(4):
            if (text_pos_width_4[i][0] <= mousex <= text_pos_width_4[i][1])
and text_pos_height[0] <= mousey <= text_pos_height[1]:
                mouse_click3_sound.play()
                pos = letters_object[i].letter
                ps_clicked = i
    elif count == 5:
        for i in range(5):
            if (text_pos_width_5[i][0] <= mousex <= text_pos_width_5[i][1])
and text_pos_height[0] <= mousey <= text_pos_height[1]:
                mouse_click3_sound.play()
                pos = letters_object[i].letter
                ps_clicked = i
    elif count == 6:
        for i in range(6):
            if (text_pos_width_6[i][0] <= mousex <= text_pos_width_6[i][1])
and text_pos_height[0] <= mousey <= text_pos_height[1]:
                mouse_click3_sound.play()
                pos = letters_object[i].letter
                ps_clicked = i
    elif count == 7:
        for i in range(7):
            if (text_pos_width_7[i][0] <= mousex <= text_pos_width_7[i][1])
and text_pos_height[0] <= mousey <= text_pos_height[1]:
                mouse_click3_sound.play()
                pos = letters_object[i].letter
                ps_clicked = i
    if pos is not None and ps_clicked is not None:
        p = letters_object[ps_clicked].letter
        letters_object.pop(ps_clicked)
        count -= 1
        print(f'Count: {count}: POS: {pos}: PS Clicked: {ps_clicked}')
        clicked_objects.append(pos)
        font = pygame.font.SysFont('Verdana', 60, bold=True)
        text = font.render(pos, True, (0, 255, 0))
        text_surfaces.append(text)
        return count, pos, ps_clicked
    else:
        return count, "", 0


def shuffle():
    random.shuffle(letters_object)


change_fg = 0
letters = [chr(i) for i in range(ord('A'), ord('Z') + 1)]
text_pos_width_7 = [(189, 239), (265, 318), (341, 407), (427, 484), (509,
560), (583, 640), (664, 722)]
```

```
text_pos_width_6 = [(211, 263), (291, 345), (372, 428), (447, 494), (526,
587), (609, 666)]
text_pos_width_5 = [(248, 299), (325, 383), (409, 454), (486, 548), (568,
620)]
text_pos_width_4 = [(281, 335), (360, 410), (439, 490), (520, 580)]
text_pos_width_3 = [(314, 372), (395, 444), (465, 533)]
text_pos_width_2 = [(346, 401), (428, 478)]
text_pos_width_1 = [(383, 443)]
text_pos_height = [464, 524]
random.shuffle(letters)
letters_object = []
text_surfaces = []
numbers = [3, 4, 5, 6, 7]
# Initialize pygame
pygame.init()
pygame.mixer.init()

# Create a window
window = pygame.display.set_mode((900, 600))
# Set the window title
pygame.display.set_caption("Word Scrabble")

# Cosmetics - Graphics
cursor_img = pygame.image.load("images/blue-pointer.png")
pygame.mouse.set_visible(False)
# Cosmetics - Audio
mouse_click1_sound = pygame.mixer.Sound('audio/mixkit-light-button-
2580.wav')
mouse_click2_sound = pygame.mixer.Sound('audio/mixkit-sci-fi-click-
900.wav')
mouse_click3_sound = pygame.mixer.Sound('audio/mixkit-message-pop-alert-
2354.mp3')
background_music = pygame.mixer.Sound('audio/In-DreamLand.wav')
background_music.set_volume(0.1)
# Draw a circle on the surface
fill = (180, 180, 180)
fg = (200, 250, 0)
bg = (0, 0, 200)
font = pygame.font.SysFont('Verdana', 70, bold=True)
clicked_objects = []
num = random.choice(numbers)
num = 4
temp_num = num

for i in range(num):
    letter = Letter(random.choice(letters))
    letters_object.append(letter)
# Useful Images/Icons
sound = SoundToggle()
refresh = Button(387, 412, 100, 25, text='Shuffle')
refresh_pos = [(387, 487), (412, 437)]
x, y = 900 // 2.1, 450
# Run the game loop
pos_clicked = None
running = True
show = False
letter_clicked = None
while running:
    window.fill(fill)
    putletters(x, y, pos_clicked)
    # Handle events
```

```
    for event in pygame.event.get():
        refresh.create(window)
        sound.create(window)
        add_to_board(show, letter_clicked)
        letter_clicked = None
        if event.type == pygame.MOUSEBUTTONDOWN:
            num, letter_clicked, pos_clicked = check_pos_clicked(num,
event.pos[0], event.pos[1])
            print(f"{num}: {letter_clicked}: {pos_clicked}: {event.pos}")
            # Add letter to board
            Cx, Cy = event.pos
            if (720 <= Cx <= 860) and (97 <= Cy <= 166):
                submit(clicked_objects)
            if (320 <= Cx <= 550) and (96 <= Cy <= 165):
                num = remove_from_board(num, event.pos[0], event.pos[1])
            if (27 <= Cx <= 78) and (209 <= Cy <= 388):
                show = not show
            if (387 <= Cx <= 487) and (412 <= Cy <= 437):
                mouse_click2_sound.play()
                shuffle()
            if (816 <= Cx <= 847) and (21 <= Cy <= 34):
                sound.toggle(mouse_click2_sound)
                sound.toggle(mouse_click1_sound)
                sound.toggle(mouse_click3_sound)
        mouse_pos = pygame.mouse.get_pos()
        window.blit(cursor_img, mouse_pos)
        if event.type == pygame.QUIT:
            running = False
        pygame.display.update()
# Close the window
pygame.quit()
```



RMC Appointments

```
import Login
import datetime
import os
import sys
import tkinter.ttk
from tkinter import *
from tkinter import messagebox
from tkinter import ttk
import mysql.connector
import pyautogui
from mysql.connector import Error
from tkcalendar import Calendar

device_res_width, device_res_height = pyautogui.size()
x = Login.main()
if x[0] == '' or x[1] == '':
    exit(123)
# try:
#     test_server = mysql.connector.connect(
#         host='localhost',
#         user=x[0],
#         password=x[1]
#     )
#     if test_server.is_connected():
#         print("server is connected")
#         test_server.close()
# except:
#     messagebox.showerror('Error', 'Server is not connected')
#     exit(1)

master = Tk()
master.title("RMC Appointments")
master.geometry(f'{int(device_res_width / 1.5)}x{int(device_res_height /
1.4)}')
master.minsize(400, 300)
master.maxsize(device_res_width, device_res_height)
master.attributes('-transparent', True)
master.config(bg='systemTransparent')


class Personnel:
    def __init__(self, name, num_appointments, s_time_hours, e_time_hours,
s_time_mins, e_time_mins):
        self.times_avail = []
        self.name = name
        self.appointments = num_appointments
        self.start_time_hours = s_time_hours
        self.end_time_hours = e_time_hours
        self.start_time_mins = s_time_mins
        self.end_time_mins = e_time_mins

    def set_time_slots(self):
        hours = self.end_time_hours - self.start_time_hours
        counter = 1
        for i in range(self.start_time_hours, self.end_time_hours):
            ifstate = True
            if i <= 9:
```

```
                i = '0' + str(i)
            if self.start_time_mins == 0:
                mins = '00'
                self.times_avail.append(str(i) + ":" + str(mins))
                if counter % 2 == 1 and self.start_time_mins == 0:
                    mins = self.start_time_mins + 30
                    self.times_avail.append(str(i) + ":" + str(mins))
                elif counter % 2 == 1 and self.start_time_mins == 30:
                    mins = self.start_time_mins - 30
                elif counter % 2 == 0 and self.start_time_mins == 0:
                    mins = self.start_time_mins + 30
                    self.times_avail.append(str(i) + ":" + str(mins))
                elif counter % 2 == 0 and self.start_time_mins == 30:
                    mins = self.start_time_mins - 30
                else:
                    mins = 0
                if self.end_time_mins == 30 and i == self.end_time_hours -
1:
                    mins = '00'
                    i = int(i)
                    if i < 9:
                        self.times_avail.append('0' + str(i + 1) + ":" +
str(mins))
                    else:
                        self.times_avail.append(str(i + 1) + ":" +
str(mins))
            if self.start_time_mins == 30:
                ifstate = False
                mins = '30'
                self.times_avail.append(str(i) + ":" + str(mins))
                if self.end_time_mins == 0 and i == self.end_time_hours -
1:
                    break
                if counter % 2 == 1 and self.start_time_mins == 0:
                    mins = '30'
                elif counter % 2 == 1 and self.start_time_mins == 30:
                    mins = '00'
                    i = int(i)
                    if i < 9:
                        self.times_avail.append('0' + str(i + 1) + ":" +
str(mins))
                    else:
                        self.times_avail.append(str(i + 1) + ":" +
str(mins))
                elif counter % 2 == 0 and self.start_time_mins == 0:
                    mins = '30'
                elif counter % 2 == 0 and self.start_time_mins == 30:
                    mins = '00'
                    i = int(i)
                    if i < 9:
                        self.times_avail.append('0' + str(i + 1) + ":" +
str(mins))
                    else:
                        self.times_avail.append(str(i + 1) + ":" +
str(mins))
                else:
                    mins = '30'
            counter += 1
        if self.end_time_mins == 0:
            self.end_time_mins = '00'
        endtime = str(self.end_time_hours) + ':' + str(self.end_time_mins)
```

```
        self.times_avail.append(endtime)


person1 = Personnel('Dr E. Nepaul', '5', 9, 14, 30, 0)
person1.set_time_slots()


# print(person1.times_avail)


def add_availability_btn():
    avail = Toplevel(master)
    avail.title('Add Doctor Availability')
    try:
        for i in range(9):
            Grid.columnconfigure(avail, i, weight=1)
            Grid.rowconfigure(avail, i + 1, weight=1)
    except:
        print("Index Error")

    def submit_change(num_appointments, day, start_time, end_time):
        name = str(doctor_listvar.get())
        # specialisation = str(service_listvar.get())
        num_appointments = num_appointments
        day_appt = day
        start_time = start_time
        end_time = end_time

    # service_list = ("Gynaecologist", "Orthodontist", "Ultrasound",
"Mammogram", "Pap Smear", "Covid Test",
    #                 "Vaccinations", "Obstetrician")
    # service_listvar = StringVar(avail)
    # service_listvar.set('Gynaecologist')
    doctor_list = ("Dr L. Morris", "Dr E. Nepaul", "Dr McCrae", "Sasha")
    doctor_listvar = StringVar(avail)
    doctor_listvar.set('Dr E. Nepaul')
    # service = OptionMenu(avail, service_listvar, *service_list)
    doctor = OptionMenu(avail, doctor_listvar, *doctor_list)
    num_of_appointmentsL = Label(avail, text='Number of Appointments')
    num_of_appointmentsE = Entry(avail, relief='groove', width=5)
    start = Label(avail, text='From', font='Arial 18 bold')
    end = Label(avail, text='To', font='Arial 18 bold')
    workdays = Label(avail, text='Days', font='Arial 18 bold')
    days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
'Saturday', 'Sunday']
    times = ('N/A', '1:00', '1:30', '2:00', '2:30', '3:00', '3:30', '4:00',
'4:30', '5:00', '5:30', '6:00',
             '6:30', '7:00', '7:30', '8:00', '8:30', '9:30', '10:00',
'10:30', '11:00', '11:30', '12:00', '12:30')
    timestart = []
    timestop = []
    time_startL = Label(avail, text='Start Time')
    time_stopL = Label(avail, text='End Time')

    localelist = ('AM', 'PM')
    locale1V = StringVar(avail)
    locale1V.set('AM')
    locale1 = OptionMenu(avail, locale1V, *localelist)
    locale1.config(width=1)
    locale2V = StringVar(avail)
    locale2V.set('PM')
```

```
    locale2 = OptionMenu(avail, locale2V, *localelist)
    locale2.config(width=1)
    locale3V = StringVar(avail)
    locale3V.set('AM')
    locale3 = OptionMenu(avail, locale3V, *localelist)
    locale3.config(width=1)
    locale4V = StringVar(avail)
    locale4V.set('PM')
    locale4 = OptionMenu(avail, locale4V, *localelist)
    locale4.config(width=1)
    locale5V = StringVar(avail)
    locale5V.set('AM')
    locale5 = OptionMenu(avail, locale5V, *localelist)
    locale5.config(width=1)
    locale6V = StringVar(avail)
    locale6V.set('PM')
    locale6 = OptionMenu(avail, locale6V, *localelist)
    locale6.config(width=1)
    locale7V = StringVar(avail)
    locale7V.set('AM')
    locale7 = OptionMenu(avail, locale7V, *localelist)
    locale7.config(width=1)
    locale8V = StringVar(avail)
    locale8V.set('PM')
    locale8 = OptionMenu(avail, locale8V, *localelist)
    locale8.config(width=1)
    locale9V = StringVar(avail)
    locale9V.set('AM')
    locale9 = OptionMenu(avail, locale9V, *localelist)
    locale9.config(width=1)
    locale10V = StringVar(avail)
    locale10V.set('PM')
    locale10 = OptionMenu(avail, locale10V, *localelist)
    locale10.config(width=1)
    locale11V = StringVar(avail)
    locale11V.set('AM')
    locale11 = OptionMenu(avail, locale11V, *localelist)
    locale11.config(width=1)
    locale12V = StringVar(avail)
    locale12V.set('PM')
    locale12 = OptionMenu(avail, locale12V, *localelist)
    locale12.config(width=1)
    locale13V = StringVar(avail)
    locale13V.set('AM')
    locale13 = OptionMenu(avail, locale13V, *localelist)
    locale13.config(width=1)
    locale14V = StringVar(avail)
    locale14V.set('PM')
    locale14 = OptionMenu(avail, locale14V, *localelist)
    locale14.config(width=1)

    monday_startV = StringVar()
    monday_start = ttk.Combobox(avail, textvariable=monday_startV, width=5)
    monday_start['values'] = times
    monday_stopV = StringVar()
    monday_stop = ttk.Combobox(avail, textvariable=monday_stopV, width=5)
    monday_stop['values'] = times

    tuesday_startV = StringVar()
    tuesday_start = ttk.Combobox(avail, textvariable=tuesday_startV,
width=5)
```

```
    tuesday_start['values'] = times
    tuesday_stopV = StringVar()
    tuesday_stop = ttk.Combobox(avail, textvariable=tuesday_stopV, width=5)
    tuesday_stop['values'] = times

    wednesday_startV = StringVar()
    wednesday_start = ttk.Combobox(avail, textvariable=wednesday_startV,
width=5)
    wednesday_start['values'] = times
    wednesday_stopV = StringVar()
    wednesday_stop = ttk.Combobox(avail, textvariable=wednesday_stopV,
width=5)
    wednesday_stop['values'] = times

    thursday_startV = StringVar()
    thursday_start = ttk.Combobox(avail, textvariable=thursday_startV,
width=5)
    thursday_start['values'] = times
    thursday_stopV = StringVar()
    thursday_stop = ttk.Combobox(avail, textvariable=thursday_stopV,
width=5)
    thursday_stop['values'] = times

    friday_startV = StringVar()
    friday_start = ttk.Combobox(avail, textvariable=friday_startV, width=5)
    friday_start['values'] = times
    friday_stopV = StringVar()
    friday_stop = ttk.Combobox(avail, textvariable=friday_stopV, width=5)
    friday_stop['values'] = times

    saturday_startV = StringVar()
    saturday_start = ttk.Combobox(avail, textvariable=saturday_startV,
width=5)
    saturday_start['values'] = times
    saturday_stopV = StringVar()
    saturday_stop = ttk.Combobox(avail, textvariable=saturday_stopV,
width=5)
    saturday_stop['values'] = times

    sunday_startV = StringVar()
    sunday_start = ttk.Combobox(avail, textvariable=sunday_startV, width=5)
    sunday_start['values'] = times
    sunday_stopV = StringVar()
    sunday_stop = ttk.Combobox(avail, textvariable=sunday_stopV, width=5)
    sunday_stop['values'] = times

    for i in range(7):
        day = Label(avail, text=days[i], justify='left', borderwidth=2,
relief='sunken', width=10, font='Arial 16 bold')
        day.grid(column=1, row=i + 3, pady=5, ipadx=50, padx=2)
        colon = Label(avail, text=':')
        colon.grid(column=2, row=i + 3, pady=5)
        dash = Label(avail, text='-')
        dash.grid(column=5, row=i + 3, pady=5)

    monday_start.grid(column=3, row=3, pady=5)
    locale1.grid(column=4, row=3, padx=10)
    monday_stop.grid(column=6, row=3, pady=5, padx=1)
    locale2.grid(column=7, row=3, padx=10)
    tuesday_start.grid(column=3, row=4, pady=5)
    locale3.grid(column=4, row=4)
```

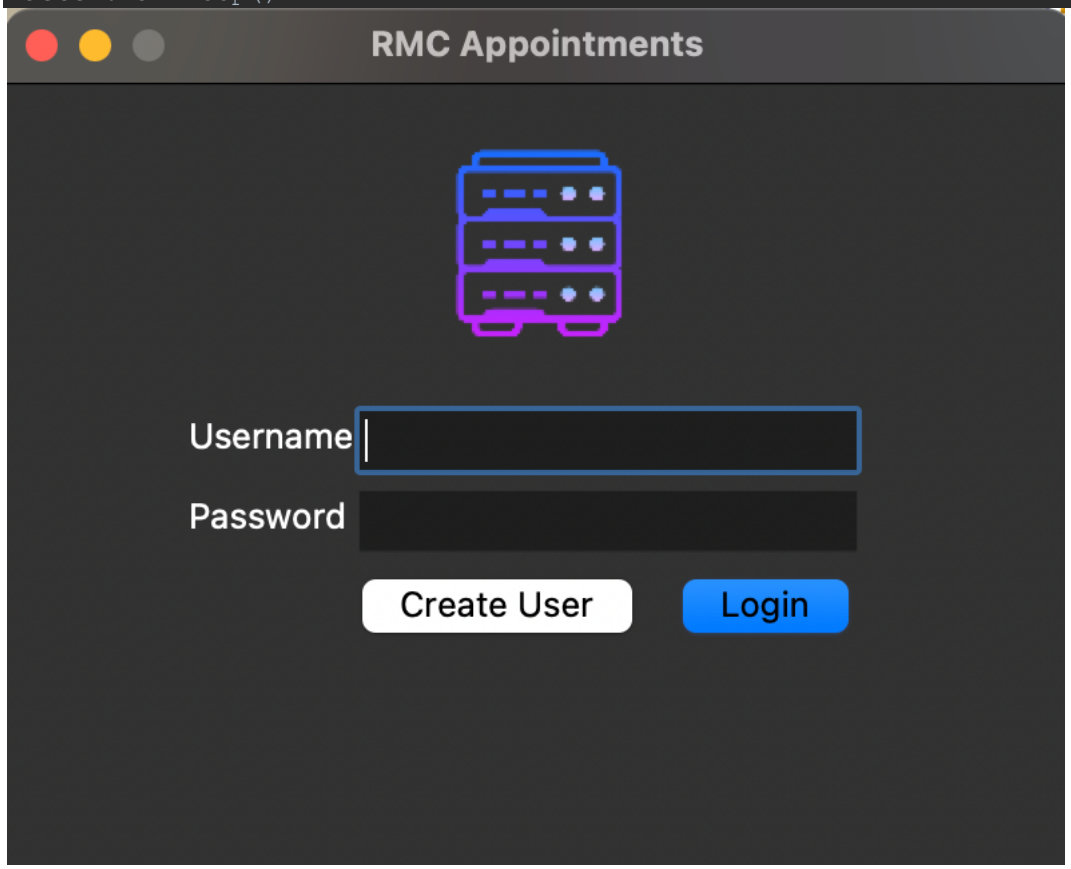
```
    tuesday_stop.grid(column=6, row=4, pady=5, padx=1)
    locale4.grid(column=7, row=4)
    wednesday_start.grid(column=3, row=5, pady=5)
    locale5.grid(column=4, row=5)
    wednesday_stop.grid(column=6, row=5, pady=5, padx=1)
    locale6.grid(column=7, row=5)
    thursday_start.grid(column=3, row=6, pady=5)
    locale7.grid(column=4, row=6)
    thursday_stop.grid(column=6, row=6, pady=5, padx=1)
    locale8.grid(column=7, row=6)
    friday_start.grid(column=3, row=7, pady=5)
    locale9.grid(column=4, row=7)
    friday_stop.grid(column=6, row=7, pady=5, padx=1)
    locale10.grid(column=7, row=7)
    saturday_start.grid(column=3, row=8, pady=5)
    locale11.grid(column=4, row=8)
    saturday_stop.grid(column=6, row=8, pady=5, padx=1)
    locale12.grid(column=7, row=8)
    sunday_start.grid(column=3, row=9, pady=5)
    locale13.grid(column=4, row=9)
    sunday_stop.grid(column=6, row=9, pady=5, padx=1)
    locale14.grid(column=7, row=9)

    doctor.grid(column=8, row=1)
    start.grid(column=3, row=2)
    end.grid(column=6, row=2)
    workdays.grid(column=1, row=2)


def create_appointments_btn(event=True):
    from datetime import date
    date = date.today()
    if cal.selection_get() < date:
        messagebox.showerror('Past Date', "Cannot create an appointment for
a day that has past!", parent=master)
        return 1

    getdate = cal.selection_get()
    create = Toplevel(master)
    create.title(f'Create Appointment for {getdate}')
    try:
        for i in range(9):
            Grid.columnconfigure(create, i, weight=1)
            Grid.rowconfigure(create, i + 1, weight=1)
    except:
        print("Index Error")
    fnamelabel = Label(create, text="First Name", anchor='e',
justify='right', width=12)
    fnameEntry = Entry(create)
    lnameLabel = Label(create, text="Last Name")
    lnameEntry = Entry(create)
    rolelist = ['hkdfbka']
    doctorlist = []
    timelist = []
    if server.is_connected():
        print("Server Is Connected")
        cursor = server.cursor()
        # cursor.execute(f'SELECT DISTINCT Specialisation FROM
rmc.availability')
        # get_specials = cursor.fetchall()
        # for specials in get_specials:
```

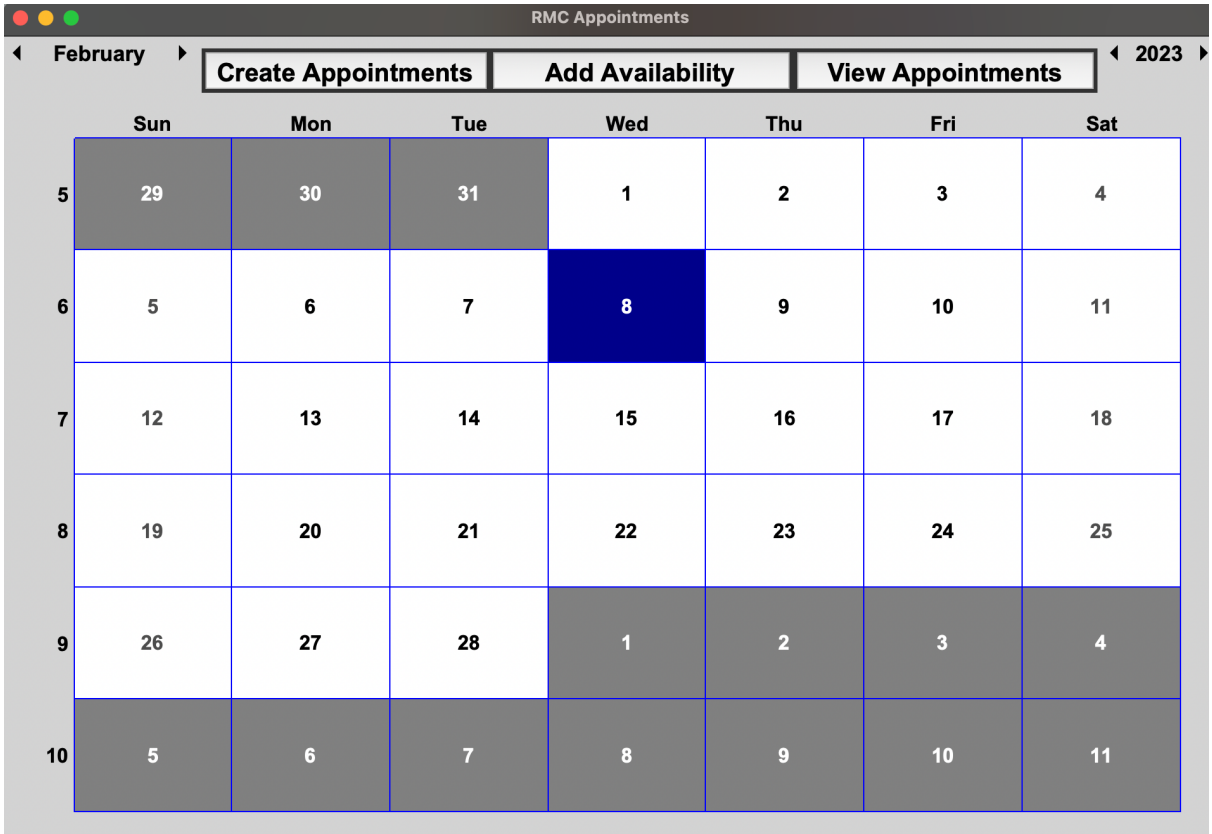
```
        #     result = specials[0]
        #     rolelist.append(str(result))
        cursor.execute(f'SELECT DISTINCT Name FROM rmc.availability')
        names = cursor.fetchall()
        for name in names:
            doctorlist.append(name)
    else:
        print("Server Error")
    rolevalue = StringVar(create)
    rolevalue.set("Choose a department")
    department = OptionMenu(create, rolevalue, *rolelist)
    doctorvalue = StringVar(create)
    doctorlist.clear()
    timevar = StringVar(create)

    fnamelabel.grid(column=0, row=0)
    fnameEntry.grid(column=1, row=0)
    lnameLabel.grid(column=2, row=0, padx=5)
    lnameEntry.grid(column=3, row=0, padx=5)
    department.grid(column=0, row=1, pady=5)

    # rolevalue.trace("w", validate)
    # set_time = (list_of_times)


def view_appointments_btn():
    getdate = cal.selection_get()
    view = Toplevel(master)
    view.title(f'Appointments for {getdate}')
    view.geometry('300x300')
    calendar_events = cal.get_calevents(getdate)
    i = -1

    # def callback(event=True):
    #     if server.is_connected():
    #         cursor = server.cursor()
    #         cursor.execute(f'SELECT * FROM rmc.appointments WHERE
Department = \'Orthodontist\'')
    #         results = cursor.fetchall()
    #         labeled = Toplevel(view)
    #         labeled.title('Patient History')
    #         for result in results:
    #             n1 = Label(labeled, text=f'{result}')
    #             n1.grid(column=0, row=0)

    for events in calendar_events:
        get_cal_events = cal.calevent_cget(events, option="text")
        label = str(get_cal_events)
        labelname = Label(view, text=f'⏱ {label}', borderwidth=1,
relief="ridge", justify='left', anchor='w', width=30)
        labelname.grid(column=0, row=i + 1)
        # labelname.bind('<Button-1>', callback)
        i += 1


try:
    server = mysql.connector.connect(
        host='localhost',
        user='root',
        password='$TR8up95'
    )
```

```
except Error as e:
    messagebox.showerror('Connection Error', 'Connection to database
failed.\nPlease check your internet connection',
                         parent=master)


def get_appointments(event=True):
    if not event:
        restart()
    if server.is_connected():
        cursor = server.cursor()
        # Get Past calendar events and assign them the past_appointment
tag, so they have a different color
        cursor.execute(f'SELECT * FROM rmc.appointments WHERE Date <
\'{curr_date.year}-{curr_date.month}'
                       f'-{curr_date.day}\' ORDER BY Date ASC, Time ASC')
        past_events = cursor.fetchall()
        for event in past_events:
            x = event
            name = str(x[0] + ' ' + x[1])
            date = x[4]
            department = str(x[2])
            doctor = str(x[3])
            time = str(x[5])
            time = time[:5]
            if time[4] == ':':
                time = time[:4]
                time = '0' + time
            cal.calevent_create(date, text=f'{time_convert[time]}-
{name}\n{department}, {doctor}',
                                tags='past_appointment')
        # Get current and upcoming events and assign them the tag
appointment, so they have a different color
        cursor.execute(f'SELECT * FROM rmc.appointments WHERE Date >=
\'{curr_date.year}-{curr_date.month}'
                       f'-{curr_date.day}\' ORDER BY Date ASC, Time ASC')
        curr_upcoming_events = cursor.fetchall()
        for result in curr_upcoming_events:
            x = result
            name = str(x[0] + ' ' + x[1])
            date = x[4]
            department = str(x[2])
            doctor = str(x[3])
            time = str(x[5])
            time = time[:5]
            if time[4] == ':':
                time = time[:4]
                time = '0' + time
            cal.calevent_create(date, text=f'{time_convert[time]}-
{name}\n{department}, {doctor}', tags='appointment')


time_convert = {
    "13:00": "1:00 PM", "13:30": "1:30 PM",
    "14:00": "2:00 PM", "14:30": "2:30 PM",
    "15:00": "3:00 PM", "15:30": "3:30 PM",
    "16:00": "4:00 PM", "16:30": "4:30 PM",
    "17:00": "5:00 PM", "17:30": "5:30 PM",
    "18:00": "6:00 PM", "18:30": "6:30 PM",
    "19:00": "7:00 PM", "19:30": "7:30 PM",
    "20:00": "8:00 PM", "20:30": "8:30 PM",
```

```
    "21:00": "9:00 PM", "21:30": "9:30 PM",
    "22:00": "10:00 PM", "22:30": "10:30 PM",
    "23:00": "11:00 PM", "23:30": "11:30 PM",
    "00:00": "12:00 AM", "0:30": "12:30 AM",
    "01:00": "1:00 AM", "01:30": "1:30 AM",
    "02:00": "2:00 AM", "02:30": "2:30 AM",
    "03:00": "3:00 AM", "03:30": "3:30 AM",
    "04:00": "4:00 AM", "04:30": "4:30 AM",
    "05:00": "5:00 AM", "05:30": "5:30 AM",
    "06:00": "6:00 AM", "06:30": "6:30 AM",
    "07:00": "7:00 AM", "07:30": "7:30 AM",
    "08:00": "8:00 AM", "08:30": "8:30 AM",
    "09:00": "9:00 AM", "09:30": "9:30 AM",
    "10:00": "10:00 AM", "10:30": "10:30 AM",
    "11:00": "11:00 AM", "11:30": "11:30 AM",
    "12:00": "12:00 PM", "12:30": "12:30 PM",
}

inv_timeconvert = {v: k for k, v in time_convert.items()}

curr_date = datetime.datetime.now()
# Date we get for appointment from user
x = datetime.date(2022, 0o06, 1)
cal = Calendar(master, selectmode='day', year=curr_date.year,
month=curr_date.month, day=curr_date.day, borderwidth=30,
               cursor="arrow", background='lightgrey', foreground="black",
font="Arial 16 bold",
               headersbackground='lightgrey', bordercolor='blue',
normalbackground='white', tooltipdelay=500,
               tooltipforeground='yellow', othermonthbackground='grey',
weekendbackground='white',
               othermonthwebackground='grey', othermonthforeground='white',
othermonthweforeground='white',
               firstweekday="sunday", arrowcolor='red',
selectbackground='darkblue')
createAppointments = Button(master, text="Create Appointments", bg='blue',
width=16, height=1, font="Arial 20 bold",
                           command=create_appointments_btn)
viewAppointments = Button(master, text="View Appointments", bg='blue',
width=17, height=1, font="Arial 20 bold",
                          command=view_appointments_btn)
addAvailability = Button(master, text='Add Availability', bg='blue',
width=17, height=1, font="Arial 20 bold",
                         command=add_availability_btn)
s = tkinter.ttk.Style(master)
s.theme_use('clam')
cal.tag_config('appointment', background='#740f35', foreground='white')
cal.tag_config('past_appointment', background='#32CD32',
foreground='white')
from datetime import datetime

count = []


def restart():
    os.execv(sys.executable, ['python'] + sys.argv)


def check_time(event=True):
    now = datetime.now()
    start_time = now.strftime("%D")
```

```
    count.append(start_time)
    current_time = now.strftime("%D")
    if current_time > count[0]:
        restart()


master.bind('<Enter>', check_time)
# master.bind('<Double-Button-1>', create_appointments_btn)
cal.pack(fill="both", expand=True)
# viewAppointments.place(x=340, y=10)
createAppointments.place(relx=0.164, y=10)
addAvailability.place(relx=0.4, y=10)
viewAppointments.place(relx=0.65, y=10)
get_appointments()
master.mainloop()
```




RMC Appointments
February
Create Appointments
Add Availability
View Appointments
2023
Sun
Mon
Tue
Wed
Thu
Fri
Sat
5
29
30
31
1
2
3
4
6
5
6
7
8
9
10
11
7
12
13
14
15
16
17
18
8
19
20
21
22
23
24
25
9
26
27
28
1
2
3
4
10
5
6
7
8
9
10
11