```
import tkinter as tk
from tkinter import messagebox, ttk
import qrcode
from PIL import Image, ImageTk
from twilio.rest import Client  # Import Twilio Client for SMS functionality

class TaxPaymentApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Online Tax Payment and Insurance")
        self.root.option_add("*Font", "Helvetica 14")

        # Colors
        self.primary_color = "#007acc"  # Blue
        self.secondary_color = "#f0a500"  # Orange
        self.background_color = "#f0f0f0"  # Light Gray
        self.frame_color = "#ffffff"  # White
        self.text_color = "#333333"  # Dark Gray

        # Initialize UI elements
        self.qr_code_label = None  # Initialize QR code label attribute
        self.logo_img = Image.open("government_logo.png").resize((100, 100))  # Load and resize
government logo image
        self.logo_img_tk = ImageTk.PhotoImage(self.logo_img)  # Convert Image object to Tkinter
PhotoImage

        # Initialize goals list
        self.goals = []

        # Initialize sections
        self.initialize_login_ui()
        self.initialize_register_ui()
        self.initialize_forgot_password_ui()
        self.initialize_tax_payment_ui()
        self.initialize_insurance_ui()
        self.initialize_interest_rates_ui()
        self.initialize_goal_tracking_ui()

        # Backward and Forward buttons
        self.back_button = tk.Button(root, text="Back", command=self.show_previous_section,
state=tk.DISABLED, bg=self.secondary_color, fg="white", padx=10)
        self.back_button.grid(row=4, column=0, pady=10)

        self.forward_button = tk.Button(root, text="Next", command=self.show_next_section,
bg=self.secondary_color, fg="white", padx=10)
        self.forward_button.grid(row=4, column=1, pady=10)

        # Track current section
        self.current_section = 0  # 0: Login, 1: Register, 2: Forgot Password, 3: Tax Payment, 4: Insurance,
5: Interest Rates, 6: Goal Tracking

        # Twilio credentials
        self.account_sid = "YOUR_ACCOUNT_SID"
        self.auth_token = "YOUR_AUTH_TOKEN"
        self.twilio_phone_number = "YOUR_TWILIO_PHONE_NUMBER"
```

```
    def initialize_login_ui(self):
        # Login Section
        self.login_frame = tk.LabelFrame(self.root, text="Login", bg=self.frame_color, fg=self.text_color,
padx=20, pady=20)
        self.login_frame.grid(row=0, column=0, padx=20, pady=20, columnspan=2)
        self.login_frame.grid_remove()

        tk.Label(self.login_frame, text="Username:", bg=self.frame_color, fg=self.text_color).grid(row=0,
column=0, padx=10, pady=5)
        self.login_username_entry = tk.Entry(self.login_frame, width=30)
        self.login_username_entry.grid(row=0, column=1, padx=10, pady=5)

        tk.Label(self.login_frame, text="Password:", bg=self.frame_color, fg=self.text_color).grid(row=1,
column=0, padx=10, pady=5)
        self.login_password_entry = tk.Entry(self.login_frame, show="*", width=30)
        self.login_password_entry.grid(row=1, column=1, padx=10, pady=5)

        tk.Button(self.login_frame, text="Login", command=self.login_action, bg=self.primary_color,
fg="white").grid(row=2, column=0, columnspan=2, pady=10)

        # Government logo on the canvas
        self.login_canvas = tk.Canvas(self.login_frame, width=100, height=100, bg=self.frame_color,
highlightthickness=0)
        self.login_canvas.create_image(50, 50, image=self.logo_img_tk)
        self.login_canvas.grid(row=0, column=2, rowspan=3, padx=10)

        def login_action(self):
            username = self.login_username_entry.get()
            password = self.login_password_entry.get()
            # Implement your login logic here (authentication against database or hardcoded credentials)

            # Dummy check for demonstration
            if username == "admin" and password == "admin":
                messagebox.showinfo("Login Successful", f"Welcome, {username}!")
                self.login_frame.grid_remove()
                self.tax_frame.grid()
                self.current_section = 3
                self.show_navigation_buttons()
            else:
                messagebox.showerror("Login Failed", "Invalid username or password. Please try again.")

    def initialize_register_ui(self):
        # Register Section
        self.register_frame = tk.LabelFrame(self.root, text="Register", bg=self.frame_color,
fg=self.text_color, padx=20, pady=20)
        self.register_frame.grid(row=0, column=0, padx=20, pady=20, columnspan=2)
        self.register_frame.grid_remove()

        tk.Label(self.register_frame, text="Username:", bg=self.frame_color, fg=self.text_color).grid(row=0,
column=0, padx=10, pady=5)
        self.register_username_entry = tk.Entry(self.register_frame, width=30)
        self.register_username_entry.grid(row=0, column=1, padx=10, pady=5)

        tk.Label(self.register_frame, text="Password:", bg=self.frame_color, fg=self.text_color).grid(row=1,
```

```
column=0, padx=10, pady=5)
        self.register_password_entry = tk.Entry(self.register_frame, show="*", width=30)
        self.register_password_entry.grid(row=1, column=1, padx=10, pady=5)

        tk.Label(self.register_frame, text="Confirm Password:", bg=self.frame_color,
fg=self.text_color).grid(row=2, column=0, padx=10, pady=5)
        self.register_confirm_password_entry = tk.Entry(self.register_frame, show="*", width=30)
        self.register_confirm_password_entry.grid(row=2, column=1, padx=10, pady=5)

        tk.Button(self.register_frame, text="Register", command=self.register_action, bg=self.primary_color,
fg="white").grid(row=3, column=0, columnspan=2, pady=10)

        # Government logo on the canvas
        self.register_canvas = tk.Canvas(self.register_frame, width=100, height=100, bg=self.frame_color,
highlightthickness=0)
        self.register_canvas.create_image(50, 50, image=self.logo_img_tk)
        self.register_canvas.grid(row=0, column=2, rowspan=4, padx=10)

    def initialize_forgot_password_ui(self):
        # Forgot Password Section
        self.forgot_password_frame = tk.LabelFrame(self.root, text="Forgot Password", bg=self.frame_color,
fg=self.text_color, padx=20, pady=20)
        self.forgot_password_frame.grid(row=0, column=0, padx=20, pady=20, columnspan=2)
        self.forgot_password_frame.grid_remove()

        tk.Label(self.forgot_password_frame, text="Username:", bg=self.frame_color,
fg=self.text_color).grid(row=0, column=0, padx=10, pady=5)
        self.forgot_password_username_entry = tk.Entry(self.forgot_password_frame, width=30)
        self.forgot_password_username_entry.grid(row=0, column=1, padx=10, pady=5)

        tk.Button(self.forgot_password_frame, text="Reset Password",
command=self.reset_password_action, bg=self.primary_color, fg="white").grid(row=1, column=0,
columnspan=2, pady=10)

        # Government logo on the canvas
        self.forgot_password_canvas = tk.Canvas(self.forgot_password_frame, width=100, height=100,
bg=self.frame_color, highlightthickness=0)
        self.forgot_password_canvas.create_image(50, 50, image=self.logo_img_tk)
        self.forgot_password_canvas.grid(row=0, column=2, rowspan=2, padx=10)

    def initialize_tax_payment_ui(self):
        # Tax Payment Section
        self.tax_frame = tk.LabelFrame(self.root, text="Tax Payment", bg=self.frame_color,
fg=self.text_color, padx=20, pady=20)
        self.tax_frame.grid(row=0, column=0, padx=20, pady=20, columnspan=2)

        self.name_label = tk.Label(self.tax_frame, text="Name:", bg=self.frame_color, fg=self.text_color)
        self.name_label.grid(row=0, column=0, sticky="e", pady=5)
        self.name_entry = tk.Entry(self.tax_frame, width=30)
        self.name_entry.grid(row=0, column=1, pady=5)

        self.tax_type_label = tk.Label(self.tax_frame, text="Tax Type:", bg=self.frame_color,
fg=self.text_color)
        self.tax_type_label.grid(row=1, column=0, sticky="e", pady=5)
        self.tax_type_var = tk.StringVar()
```

```
        self.tax_type_var.set("GST")  # default value
        self.tax_type_menu = tk.OptionMenu(self.tax_frame, self.tax_type_var, "GST", "Income Tax",
"House Tax", "Water Tax", command=self.update_gst_visibility)
        self.tax_type_menu.config(width=25)
        self.tax_type_menu.grid(row=1, column=1, pady=5)

        self.gst_number_label = tk.Label(self.tax_frame, text="GST Number:", bg=self.frame_color,
fg=self.text_color)
        self.gst_number_label.grid(row=2, column=0, sticky="e", pady=5)
        self.gst_number_entry = tk.Entry(self.tax_frame, width=30)
        self.gst_number_entry.grid(row=2, column=1, pady=5)

        self.amount_label = tk.Label(self.tax_frame, text="Amount:", bg=self.frame_color, fg=self.text_color)
        self.amount_label.grid(row=3, column=0, sticky="e", pady=5)
        self.amount_entry = tk.Entry(self.tax_frame, width=30)
        self.amount_entry.grid(row=3, column=1, pady=5)

        self.logo_label = tk.Label(self.tax_frame, image=self.logo_img_tk, bg=self.frame_color)
        self.logo_label.grid(row=0, column=2, rowspan=4, padx=10)

        # Tax statements
        self.tax_statements = {
            "GST": "GST is a consumption tax levied on goods and services.",
            "Income Tax": "Income tax is a tax imposed on individuals or entities that varies with respective
income or profits.",
            "House Tax": "House tax is a tax levied on the ownership of real estate.",
            "Water Tax": "Water tax is a tax imposed on water usage."
        }
        self.tax_statement_label = tk.Label(self.tax_frame, text=self.tax_statements["GST"],
wraplength=300, justify="left", bg=self.frame_color, fg=self.text_color)
        self.tax_statement_label.grid(row=4, column=0, columnspan=2, pady=10)

        self.generate_tax_button = tk.Button(self.tax_frame, text="Generate QR Code",
command=self.generate_tax_qr_code, bg=self.primary_color, fg="white", padx=10)
        self.generate_tax_button.grid(row=5, column=0, columnspan=2, pady=10)

    def initialize_insurance_ui(self):
        # Insurance Section
        self.insurance_frame = tk.LabelFrame(self.root, text="Insurance Information", padx=20, pady=20,
bg=self.frame_color, fg=self.text_color)
        self.insurance_frame.grid(row=0, column=0, padx=20, pady=20, columnspan=2)
        self.insurance_frame.grid_remove()

        self.insurance_type_label = tk.Label(self.insurance_frame, text="Insurance Type:",
bg=self.frame_color, fg=self.text_color)
        self.insurance_type_label.grid(row=0, column=0, sticky="e", pady=5)
        self.insurance_type_var = tk.StringVar()
        self.insurance_type_var.set("Life Insurance")  # default value
        self.insurance_type_menu = tk.OptionMenu(self.insurance_frame, self.insurance_type_var, "Life
Insurance", "Health Insurance", "Vehicle Insurance")
        self.insurance_type_menu.config(width=25)
        self.insurance_type_menu.grid(row=0, column=1, pady=5)

        self.insurance_amount_label = tk.Label(self.insurance_frame, text="Insurance Amount:",
bg=self.frame_color, fg=self.text_color)
```

```
        self.insurance_amount_label.grid(row=1, column=0, sticky="e", pady=5)
        self.insurance_amount_entry = tk.Entry(self.insurance_frame, width=30)
        self.insurance_amount_entry.grid(row=1, column=1, pady=5)

        self.generate_insurance_button = tk.Button(self.insurance_frame, text="Generate QR Code",
command=self.generate_insurance_qr_code, bg=self.primary_color, fg="white", padx=10)
        self.generate_insurance_button.grid(row=2, column=0, columnspan=2, pady=10)

    def initialize_interest_rates_ui(self):
        # Interest Rates Section (Dummy data)
        self.interest_frame = tk.LabelFrame(self.root, text="Interest Rates", padx=20, pady=20,
bg=self.frame_color, fg=self.text_color)
        self.interest_frame.grid(row=0, column=0, padx=20, pady=20, columnspan=2)
        self.interest_frame.grid_remove()

        self.interest_label = tk.Label(self.interest_frame, text="Real-Time Interest Rate Comparison",
bg=self.frame_color, fg=self.text_color)
        self.interest_label.grid(row=0, column=0, columnspan=2, pady=10)

        banks = ["Bank A", "Bank B", "Bank C", "Bank D"]
        rates = [3.5, 3.6, 3.4, 3.7]

        for i, (bank, rate) in enumerate(zip(banks, rates), start=1):
            tk.Label(self.interest_frame, text=f"{bank}: {rate}%", bg=self.frame_color,
fg=self.text_color).grid(row=i, column=0, columnspan=2, pady=5)

    def initialize_goal_tracking_ui(self):
        # Goal Tracking Section
        self.goal_frame = tk.LabelFrame(self.root, text="Goal Tracking", padx=20, pady=20,
bg=self.frame_color, fg=self.text_color)
        self.goal_frame.grid(row=0, column=0, padx=20, pady=20, columnspan=2)
        self.goal_frame.grid_remove()

        # Label and entry for goal setting
        ttk.Label(self.goal_frame, text="Financial Goals", font=('Helvetica', 16, 'bold')).grid(row=0, column=0,
padx=10, pady=10)
        self.goal_entry = ttk.Entry(self.goal_frame, width=40, font=('Helvetica', 14))
        self.goal_entry.grid(row=1, column=0, padx=10, pady=10)

        # Button to add goals
        ttk.Button(self.goal_frame, text="Add Goal", command=self.add_goal).grid(row=1, column=1,
padx=10, pady=10)

        # Treeview to display goals
        self.goal_tree = ttk.Treeview(self.goal_frame, columns=("Goal", "Progress"), show="headings",
height=10)
        self.goal_tree.heading("Goal", text="Goal")
        self.goal_tree.heading("Progress", text="Progress")
        self.goal_tree.column("Goal", width=300)
        self.goal_tree.column("Progress", width=150)
        self.goal_tree.grid(row=2, column=0, columnspan=2, padx=10, pady=10)

        # Button to track progress
        ttk.Button(self.goal_frame, text="Track Progress", command=self.track_progress).grid(row=3,
column=0, columnspan=2, padx=10, pady=10)
```

```
    def add_goal(self):
        goal = self.goal_entry.get()
        if goal:
            self.goals.append(goal)
            self.goal_tree.insert("", "end", values=(goal, "0%"))
            self.goal_entry.delete(0, tk.END)
        else:
            messagebox.showwarning("Empty Field", "Please enter a goal.")

    def track_progress(self):
        selected_item = self.goal_tree.selection()
        if not selected_item:
            messagebox.showwarning("No Goal Selected", "Please select a goal to track progress.")
            return

        goal_index = int(selected_item[0][1:]) - 1
        goal = self.goals[goal_index]

        # Simulating progress update (assuming random percentage)
        import random
        progress_percentage = random.randint(10, 100)

        # Update progress in the treeview
        self.goal_tree.item(selected_item, values=(goal, f"{progress_percentage}%"))

        # Show progress update message
        messagebox.showinfo("Progress Update", f"Progress for '{goal}' updated to
{progress_percentage}%.")

    def update_gst_visibility(self, tax_type):
        if tax_type == "GST":
            self.gst_number_label.grid()
            self.gst_number_entry.grid()
            self.tax_statement_label.config(text=self.tax_statements["GST"])
        elif tax_type in self.tax_statements:
            self.gst_number_label.grid_remove()
            self.gst_number_entry.grid_remove()
            self.tax_statement_label.config(text=self.tax_statements[tax_type])

        # Hide QR code when changing tax type
        if self.qr_code_label:
            self.qr_code_label.grid_remove()

    def generate_tax_qr_code(self):
        name = self.name_entry.get()
        tax_type = self.tax_type_var.get()
        gst_number = self.gst_number_entry.get() if tax_type == "GST" else "N/A"
        amount = self.amount_entry.get()

        if name and tax_type and amount:
            qr_data = f"Name: {name}\nTax Type: {tax_type}\nGST Number: {gst_number}\nAmount:
{amount}"

            qr = qrcode.QRCode()
```

```
            qr.add_data(qr_data)
            qr.make(fit=True)
            img = qr.make_image(fill_color="black", back_color="white")

            img_tk = ImageTk.PhotoImage(img)
            if self.qr_code_label:
                self.qr_code_label.grid_remove()
            self.qr_code_label = tk.Label(self.root, image=img_tk, bg=self.background_color)
            self.qr_code_label.image = img_tk
            self.qr_code_label.grid(row=3, column=0, columnspan=2, pady=20)  # Show QR code label

            # Assuming payment is successful, send SMS notification
            self.send_sms_notification("ENTER_PHONE_NUMBER_HERE")

        else:
            messagebox.showwarning("Missing Information", "Please fill in all mandatory fields (Name, Tax Type, Amount) before generating QR code.")

        self.show_navigation_buttons()

    def generate_insurance_qr_code(self):
        insurance_type = self.insurance_type_var.get()
        insurance_amount = self.insurance_amount_entry.get()

        if insurance_amount:
            qr_data = f"Insurance Type: {insurance_type}\nInsurance Amount: {insurance_amount}"

            qr = qrcode.QRCode()
            qr.add_data(qr_data)
            qr.make(fit=True)
            img = qr.make_image(fill_color="black", back_color="white")

            img_tk = ImageTk.PhotoImage(img)
            if self.qr_code_label:
                self.qr_code_label.grid_remove()
            self.qr_code_label = tk.Label(self.root, image=img_tk, bg=self.background_color)
            self.qr_code_label.image = img_tk
            self.qr_code_label.grid(row=3, column=0, columnspan=2, pady=20)  # Show QR code label

            # Assuming payment is successful, send SMS notification
            self.send_sms_notification("ENTER_PHONE_NUMBER_HERE")

        else:
            messagebox.showwarning("Missing Information", "Please fill in all mandatory fields (Insurance Amount) before generating QR code.")

        self.show_navigation_buttons()

    def show_navigation_buttons(self):
        if self.current_section == 0:
            self.back_button.config(state=tk.DISABLED)
            self.forward_button.config(state=tk.NORMAL)
        elif self.current_section == 6:
            self.back_button.config(state=tk.NORMAL)
            self.forward_button.config(state=tk.DISABLED)
```

```
    else:
        self.back_button.config(state=tk.NORMAL)
        self.forward_button.config(state=tk.NORMAL)

def show_next_section(self):
    if self.current_section == 0:
        self.login_frame.grid_remove()
        self.register_frame.grid()

    elif self.current_section == 1:
        self.register_frame.grid_remove()
        self.forgot_password_frame.grid()

    elif self.current_section == 2:
        self.forgot_password_frame.grid_remove()
        self.tax_frame.grid()

    elif self.current_section == 3:
        self.tax_frame.grid_remove()
        self.insurance_frame.grid()

    elif self.current_section == 4:
        self.insurance_frame.grid_remove()
        self.interest_frame.grid()

    elif self.current_section == 5:
        self.interest_frame.grid_remove()
        self.goal_frame.grid()

    elif self.current_section == 6:
        self.goal_frame.grid_remove()
        self.login_frame.grid()

    self.current_section = (self.current_section + 1) % 7
    self.show_navigation_buttons()

def show_previous_section(self):
    if self.current_section == 0:
        self.login_frame.grid_remove()
        self.goal_frame.grid()

    elif self.current_section == 1:
        self.register_frame.grid_remove()
        self.login_frame.grid()

    elif self.current_section == 2:
        self.forgot_password_frame.grid_remove()
        self.register_frame.grid()

    elif self.current_section == 3:
        self.tax_frame.grid_remove()
        self.forgot_password_frame.grid()

    elif self.current_section == 4:
        self.insurance_frame.grid_remove()
```

```
            self.tax_frame.grid()

        elif self.current_section == 5:
            self.interest_frame.grid_remove()
            self.insurance_frame.grid()

        elif self.current_section == 6:
            self.goal_frame.grid_remove()
            self.interest_frame.grid()

        self.current_section = (self.current_section - 1) % 7
        self.show_navigation_buttons()

    def login_action(self):
        username = self.login_username_entry.get()
        password = self.login_password_entry.get()
        # Implement your login logic here (authentication against database or hardcoded credentials)

        # Dummy check for demonstration
        if username == "admin" and password == "admin":
            messagebox.showinfo("Login Successful", f"Welcome, {username}!")
            self.show_next_section()
        else:
            messagebox.showerror("Login Failed", "Invalid username or password. Please try again.")

    def register_action(self):
        username = self.register_username_entry.get()
        password = self.register_password_entry.get()
        confirm_password = self.register_confirm_password_entry.get()
        # Implement your registration logic here (e.g., save to database)

        # Dummy check for demonstration
        if password == confirm_password:
            messagebox.showinfo("Registration Successful", f"Account created for {username}. Please
login.")
            self.show_next_section()
        else:
            messagebox.showerror("Registration Failed", "Passwords do not match. Please try again.")

    def reset_password_action(self):
        username = self.forgot_password_username_entry.get()
        # Implement your reset password logic here (e.g., send reset link or SMS)

        # Dummy check for demonstration
        if username:
            messagebox.showinfo("Password Reset", f"Password reset instructions sent to {username}.")
            self.show_next_section()
        else:
            messagebox.showerror("Password Reset Failed", "Username not found. Please try again.")

    def send_sms_notification(self, phone_number):
        # Initialize Twilio client
        client = Client(self.account_sid, self.auth_token)

        try:
```

```
            message = client.messages.create(
                body="Your payment was successful. Thank you!",
                from_=self.twilio_phone_number,
                to=phone_number
            )
            print(f"SMS notification sent to {phone_number}. SID: {message.sid}")

        except Exception as e:
            print(f"Failed to send SMS notification: {str(e)}")
            messagebox.showerror("SMS Notification", "Failed to send SMS notification. Please check your phone number.")

if __name__ == "__main__":
    root = tk.Tk()
    app = TaxPaymentApp(root)
    root.mainloop()
```