```python
import tkinter as tk  # Import the tkinter library and alias it as 'tk'
from tkinter import filedialog, messagebox  # Import filedialog and messagebox
modules from tkinter
import qrcode  # Import the qrcode library for generating QR codes
from PIL import Image, ImageTk  # Import Image and ImageTk from the Pillow
library for image handling

def generate_qr():  # Define a function to generate the QR code
    input_data = data_entry.get()  # Get the input data from the entry widget
    if not input_data:  # Check if the input data is empty
        messagebox.showerror("Error", "Please enter data to encode in the QR
code.")  # Show an error message if no data is entered
        return  # Exit the function if no data is entered

    filename = filedialog.asksaveasfilename(defaultextension=".png",
filetypes=[("PNG files", "*.png"), ("All files", "*.*")])  # Open a file save
dialog and get the filename
    if not filename:  # Check if a filename was not provided
        return  # Exit the function if no filename is provided

    # Create QR code instance
    qr = qrcode.QRCode(version=1, box_size=10, border=5)  # Create a QRCode
object with specified version, box size, and border
    # Add data to the QR code
    qr.add_data(input_data)  # Add the input data to the QR code
    qr.make(fit=True)  # Generate the QR code
    # Create an image from the QR code
    img = qr.make_image(fill_color="black", back_color="white")  # Create an
image from the QR code with black fill and white background
    # Save the image
    img.save(filename)  # Save the image to the specified filename

    # Display the QR code in the GUI
    img = Image.open(filename)  # Open the saved image
    img = img.resize((250, 250), Image.ANTIALIAS)  # Resize the image to 250x250
pixels with anti-aliasing
    img = ImageTk.PhotoImage(img)  # Convert the image to a format suitable for
tkinter
    qr_label.config(image=img)  # Set the image to the label widget
    qr_label.image = img  # Keep a reference to the image to prevent garbage
collection
```

```python
    messagebox.showinfo("Success", f"QR code generated successfully as {filename}")  # Show a success message with the filename

# Create main application window
root = tk.Tk()  # Create the main application window
root.title("QR Code Generator")  # Set the title of the window

# Create and place widgets
tk.Label(root, text="Enter data to encode:").pack(pady=10)  # Create and place a label widget with some padding
data_entry = tk.Entry(root, width=50)  # Create an entry widget with a specified width
data_entry.pack(pady=5)  # Place the entry widget with some padding

generate_button = tk.Button(root, text="Generate QR Code", command=generate_qr)  # Create a button widget and link it to the generate_qr function
generate_button.pack(pady=20)  # Place the button widget with some padding

qr_label = tk.Label(root)  # Create a label widget for displaying the QR code image
qr_label.pack(pady=10)  # Place the label widget with some padding

# Run the application
root.mainloop()  # Start the tkinter event loop
```

1. **Imports:**
    - `tkinter as tk`: Imports tkinter with an alias for ease of use.
    - `filedialog, messagebox`: Imports specific functions from tkinter for file dialogs and message boxes.
    - `qrcode`: Imports the qrcode library to generate QR codes.
    - `Image, ImageTk`: Imports Image and ImageTk from Pillow for image handling and display in the GUI.
2. **Function `generate_qr()`:**
    - Retrieves the data from the entry widget.
    - Checks if the input data is empty and shows an error message if true.
    - Prompts the user to choose a file location and name for saving the QR code image.

- Creates a QR code with specified parameters, adds the input data, and optimizes the fit.
- Generates the QR code image and saves it to the chosen file.
- Opens the saved image, resizes it, and converts it to a format suitable for displaying in tkinter.
- Updates the label to display the QR code image.
- Shows a success message indicating the QR code has been generated.

3. **GUI Setup:**
    - Creates the main application window and sets the title.
    - Adds a label prompting the user to enter data.
    - Adds an entry widget for user input.
    - Adds a button to trigger the QR code generation.
    - Adds a label to display the generated QR code image.
    - Starts the tkinter event loop to run the application.