Project Title: Random Quote Generator

Name: Sheetal

Course Code: CSM 216

Class: K23CH

Registration Number: 12310172

Roll Number: 59

Submission Date: 25-11-2024

LOVELY PROFESSIONAL UNIVERSITY

Acknowledgment

I would like to express my heartfelt gratitude to everyone who supported and guided me throughout the development of this project, Random Quote Generator.

First and foremost, I would like to thank my mentor, Mr. Aman Kumar, for their invaluable advice and guidance, which provided the foundation and direction necessary to successfully complete this project. Their insights and constructive feedback were instrumental in refining the application's functionality and ensuring that it met the project objectives.

I would also like to extend my gratitude to my colleagues and peers who shared their expertise and provided thoughtful suggestions for enhancing the application's features. Their contributions played an essential role in helping me identify potential improvements and explore effective solutions to common challenges in To-Do List Application.

Additionally, I am grateful to Lovely Professional University for providing a supportive environment and resources that enabled me to focus on and accomplish the project goals. Access to tools and platforms was critical in designing, implementing, and testing this application effectively.

Lastly, I would like to acknowledge the encouragement and patience of my family and friends, who supported me during this project's development. Their unwavering belief in my capabilities motivated me to remain dedicated to delivering a high-quality, user-friendly application.

Thank you all for your invaluable support and contributions.

Table of Contents

1. Introduction

The **Random Quote Generator** is a Python-based desktop application designed to inspire, entertain, and motivate users by providing access to a curated collection of meaningful quotes. Built using Python's Tkinter library for the graphical user interface (GUI) and SQLite for persistent data storage, this project showcases the seamless integration of modern programming tools to create an engaging user experience. The primary function of this application is to display random quotes sourced from a database, allowing users to encounter diverse perspectives, philosophies, and ideas with every click. The app goes beyond mere quote display, empowering users to personalize their experience by adding their favourite quotes to the collection, ensuring the content evolves to suit individual tastes. With an intuitive design and a user-friendly interface, the Random Quote Generator caters to a wide audience, including students seeking inspiration, professionals looking for quick motivation, or anyone in need of a momentary mental uplift. The randomized selection of quotes ensures a fresh experience each time, sparking curiosity and preventing monotony. Additionally, the app demonstrates how Python's simplicity and versatility can be harnessed to build practical, real-world applications. By combining creativity, technology, and user interaction, the Random Quote Generator not only serves as a source of positivity but also highlights the potential of programming to create meaningful digital solutions that enhance daily life.

2. Objectives and Scope of the Project

Objectives

The primary objectives of this project are:

1. Generate Random Quotes: Provide a seamless and engaging way for users to interact with a diverse set of quotes randomly chosen from a database.
2. Add Custom Quotes: Enable users to personalize the app by adding their own quotes, ensuring a dynamic and evolving experience.
3. Save and Retrieve Quotes: Maintain all quotes in a reliable SQLite database for easy access, even after restarting the application.
4. Enhance User Interaction: Create a visually appealing and intuitive GUI to ensure a positive user experience.

Scope

1. User Interaction:
    - Users can generate quotes with a click of a button.
    - Users can add new quotes along with the author's name, which are stored for future reference.
2. Database Integration:
    - Quotes are stored in an SQLite database, ensuring persistence and scalability.
    - The database structure supports efficient retrieval and addition of data.
3. Future Extensions:
    - Categorization of quotes (e.g., Inspirational, Funny, Success).
    - A search bar to filter quotes based on keywords or authors.
    - Sharing options for social media platforms.
    - Cloud storage for syncing across devices.

3. Application Tools

Programming Language:

- **Python:** Python is chosen for its ease of use, vast library support, and suitability for small-scale GUI-based applications.

Libraries and Modules:

1. **Tkinter:** Python's built-in GUI library for creating windows, labels, buttons, and other interactive elements.
    - **Advantages:** Lightweight, beginner-friendly, and sufficient for small desktop applications.
2. **SQLite3:** A lightweight, self-contained SQL database engine.
    - **Why SQLite?** It provides robust storage without requiring a separate server, making it perfect for small projects.
3. **Random Module:** A standard Python library used to randomly select quotes from the database.
    - **Why Random?** Ensures variety in quote generation, enhancing the user experience.

Development Tools:

1. **IDE:** Visual Studio Code and PyCharm were used for writing, debugging, and maintaining the code.
2. **Operating System:** Python and Tkinter's cross-platform compatibility ensures the application works on Windows, macOS, and Linux.

4. Project Design

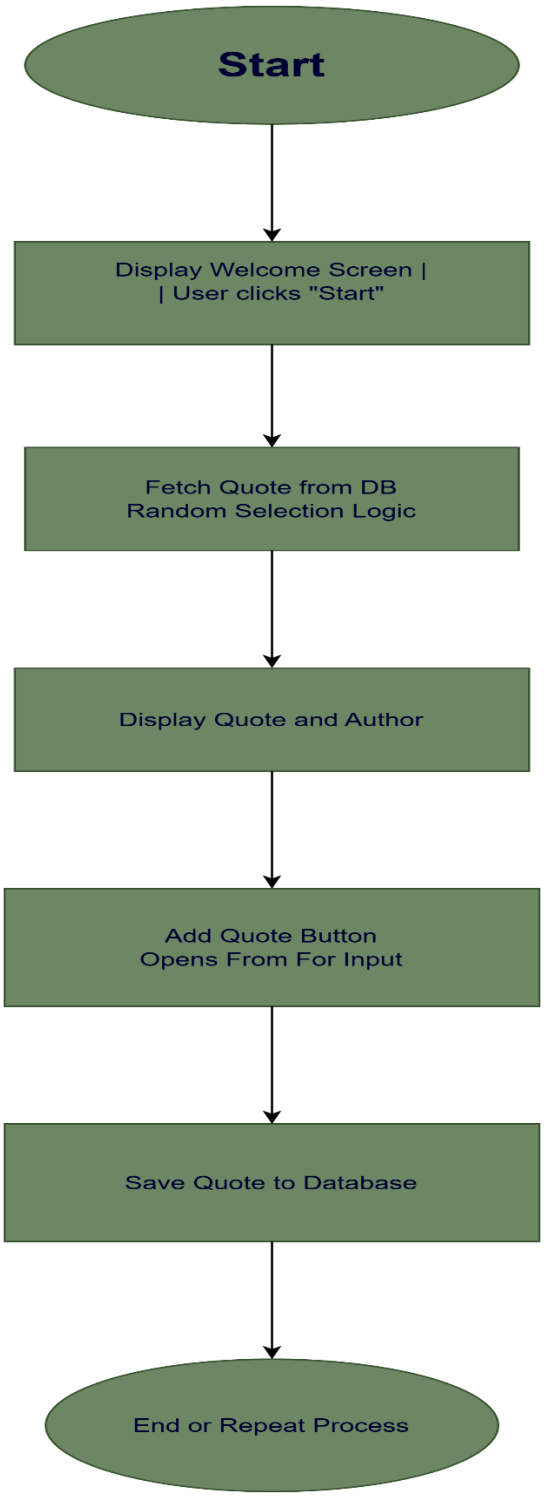
The design of the Random Quote Generator is modular, focusing on maintainability, scalability, and ease of use.

Components:

1. **Database Module:**
    - Functionality: Creates and manages the database, storing quotes and authors in a structured table.
    - Preloaded with motivational quotes for first-time users.
2. **Logic Module:**
    - Implements random selection of quotes.
    - Handles CRUD operations (Create, Read, Update, Delete) for the database.
    - Validates input from the user (e.g., no empty quotes or authors).
3. **User Interface Module:**
    - **Components:**
        1. A **welcome screen** with a "Start" button.
        2. A **main screen** displaying quotes and offering interactive buttons for generating new quotes or adding custom quotes.
        3. A **popup window** for adding new quotes.
    - **Design Considerations:**
        1. Responsive layout for readability.
        2. Consistent styling for buttons and labels.
4. **Add-Quote Module:**
    - Opens a form where users can input quotes and authors.
    - Validates inputs before saving data to the database.

5. Flowchart

Workflow of the Random Quote Generator:



Expanded Workflow:

1. *Start:*

   The application initializes, displaying the welcome screen.

2. *Database Check:*

   - Ensures the database and the quotes table exist. If not, creates them.

3. *Welcome Screen:*

   - Greets the user and provides a "Start" button to proceed to the main interface.

4. *Main Screen:*

   - Displays a randomly fetched quote and its author.
   - Offers a "Generate Quote" button to fetch a new random quote.
   - Provides an "Add Quote" button to open a form for input.

5. *Add Quote:*

   - Validates the user’s input.
   - Stores the new quote-author pair in the database.

6. *Repeat or Exit:*

   - The user can continue generating quotes or close the application.

6.Project Implementation

Implementation Steps:

1. **Database Creation:**
    - The database is initialized with a quotes table containing preloaded quotes.
    - Example SQL query to insert a quote:

      INSERT INTO quotes (quote, author) VALUES

      ('Success is not final; failure is not fatal: It is the courage to continue that counts.', 'Winston Churchill');
2. **Random Quote Generation:**
    - A SELECT query fetches all quotes from the database.
    - The random.choice function selects one quote from the fetched list.
3. **Adding Custom Quotes:**
    - Users can input new quotes and authors through a popup form.
    - Input is validated to prevent empty or duplicate entries.
4. **GUI Design:**
    - The Tkinter-based interface uses frames, labels, and buttons to create an engaging user experience.
5. **Error Handling:**
    - Handles cases like empty databases or invalid inputs.
    - Displays meaningful messages for errors.

Screenshots of the Execution:

import sq.py

C: > Users > ndahi > Downloads > import sq.py > ...

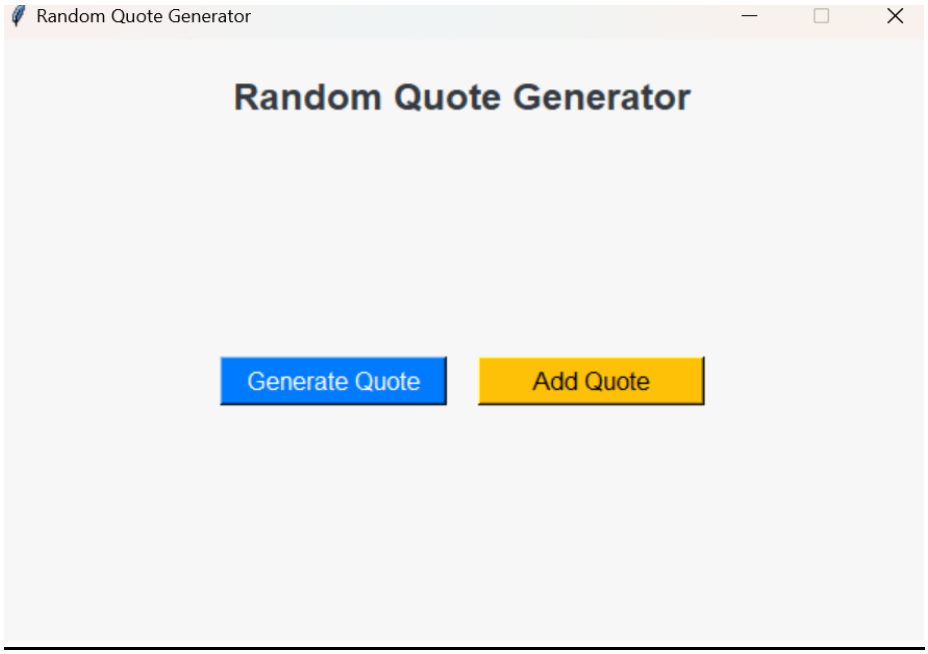
```python
import sqlite3
import random
from tkinter import *
from tkinter import messagebox

# Connect to SQLite Database
def connect_database():
    conn = sqlite3.connect("quotes.db")
    cursor = conn.cursor()
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS quotes (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            quote TEXT NOT NULL,
            author TEXT NOT NULL
        )
    """)
    # Add sample quotes (only if table is empty)
    cursor.execute("SELECT COUNT(*) FROM quotes")
    if cursor.fetchone()[0] == 0:
        sample_quotes = [
            ('The only limit to our realization of tomorrow is our doubts of today.', 'Franklin D. Roosevelt'),
            ('Life is what happens when you're busy making other plans.', 'John Lennon'),
            ('Do what you can, with what you have, where you are.', 'Theodore Roosevelt'),
            ('Success is not the key to happiness. Happiness is the key to success.', 'Albert Schweitzer'),
            ('The future belongs to those who believe in the beauty of their dreams.', 'Eleanor Roosevelt'),
            ('You miss 100% of the shots you don't take.', 'Wayne Gretzky'),
            ('In the middle of every difficulty lies opportunity.', 'Albert Einstein'),
            ('Happiness is not something ready-made. It comes from your own actions.', 'Dalai Lama'),
            ('Your time is limited, so don't waste it living someone else's life.', 'Steve Jobs'),
            ('Believe you can and you're halfway there.', 'Theodore Roosevelt'),
            ('The best way to predict the future is to invent it.', 'Alan Kay'),
            ('Don't watch the clock; do what it does. Keep going.', 'Sam Levenson'),
            ('The way to get started is to quit talking and begin doing.', 'Walt Disney'),
            ('Act as if what you do makes a difference. It does.', 'William James'),
            ('It always seems impossible until it's done.', 'Nelson Mandela')
        ]
        cursor.executemany("INSERT INTO quotes (quote, author) VALUES (?, ?)", sample_quotes)
    conn.commit()
    conn.close()

# Fetch Random Quote
def get_random_quote():
    conn = sqlite3.connect("quotes.db")
```

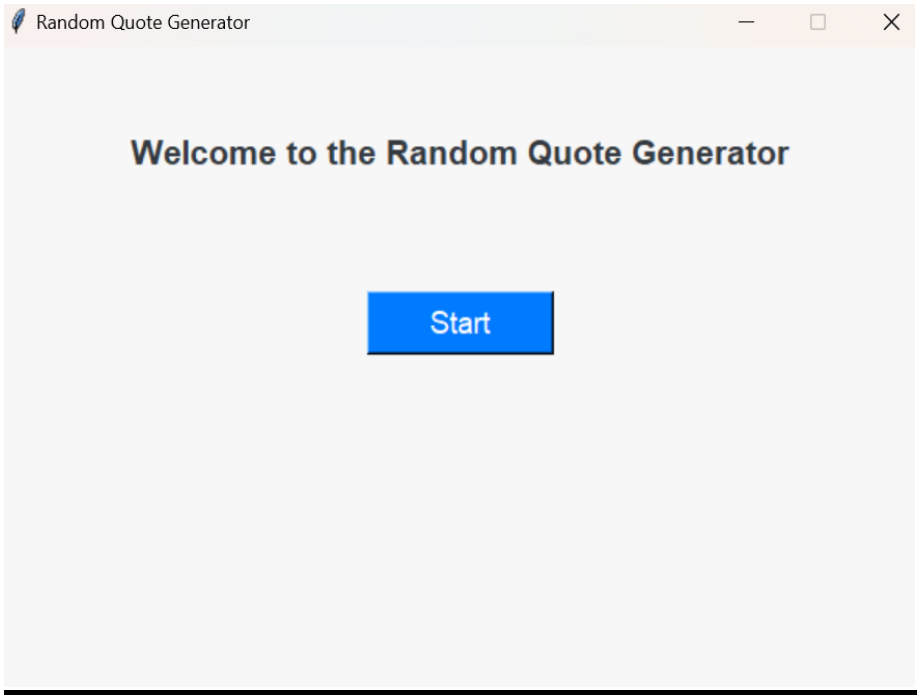
```
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM quotes")
        rows = cursor.fetchall()
        conn.close()
        if rows:
            quote, author = random.choice(rows)[1:]  # Randomly select a quote
            return quote, author
        else:
            return "No quotes available", "Unknown"

    # Display Random Quote
    def display_quote():
        quote, author = get_random_quote()
        quote_label.config(text=f'"{quote}"')
        author_label.config(text=f'- {author}')

    # Start Main App
    def start_app():
        start_frame.pack_forget()  # Hide the welcome screen
        main_frame.pack(fill=BOTH, expand=True)  # Show the main quote interface

    # Add Custom Quote
    def add_quote():
        def save_quote():
            new_quote = new_quote_entry.get("1.0", "end").strip()
            new_author = new_author_entry.get().strip()
            if new_quote and new_author:
                conn = sqlite3.connect("quotes.db")
                cursor = conn.cursor()
                cursor.execute("INSERT INTO quotes (quote, author) VALUES (?, ?)", (new_quote, new_author))
                conn.commit()
                conn.close()
                messagebox.showinfo("Success", "Quote added successfully!")
                add_window.destroy()
            else:
                messagebox.showwarning("Validation Error", "Both quote and author are required!")

        add_window = Toplevel(root)
        add_window.title("Add New Quote")
        add_window.geometry("400x300")
        add_window.resizable(False, False)

        Label(add_window, text="New Quote:", font=("Helvetica", 12)).pack(pady=10)
        new_quote_entry = Text(add_window, height=5, width=40, wrap=WORD)
        new_quote_entry.pack(pady=5)

        Label(add_window, text="Author:", font=("Helvetica", 12)).pack(pady=10)
        new_author_entry = Entry(add_window, width=30)
        new_author_entry.pack(pady=5)

        Button(add_window, text="Save", command=save_quote, font=("Helvetica", 12), bg="#28a745", fg="white").pack(pady=10)

    # Tkinter GUI Setup
    root = Tk()
    root.title("Random Quote Generator")
    root.geometry("600x400")
    root.resizable(False, False)
    root.configure(bg="#f7f7f7")

    # Welcome Screen
    start_frame = Frame(root, bg="#f7f7f7")
    start_frame.pack(fill=BOTH, expand=True)

    welcome_label = Label(start_frame, text="Welcome to the Random Quote Generator", font=("Helvetica", 16, "bold"), bg="#f7f7f7", fg="#343a40"
    welcome_label.pack(pady=50)

    start_button = Button(start_frame, text="Start", command=start_app, font=("Helvetica", 14), bg="#007bff", fg="white", width=10)
    start_button.pack(pady=20)

    # Main Quote Display
```

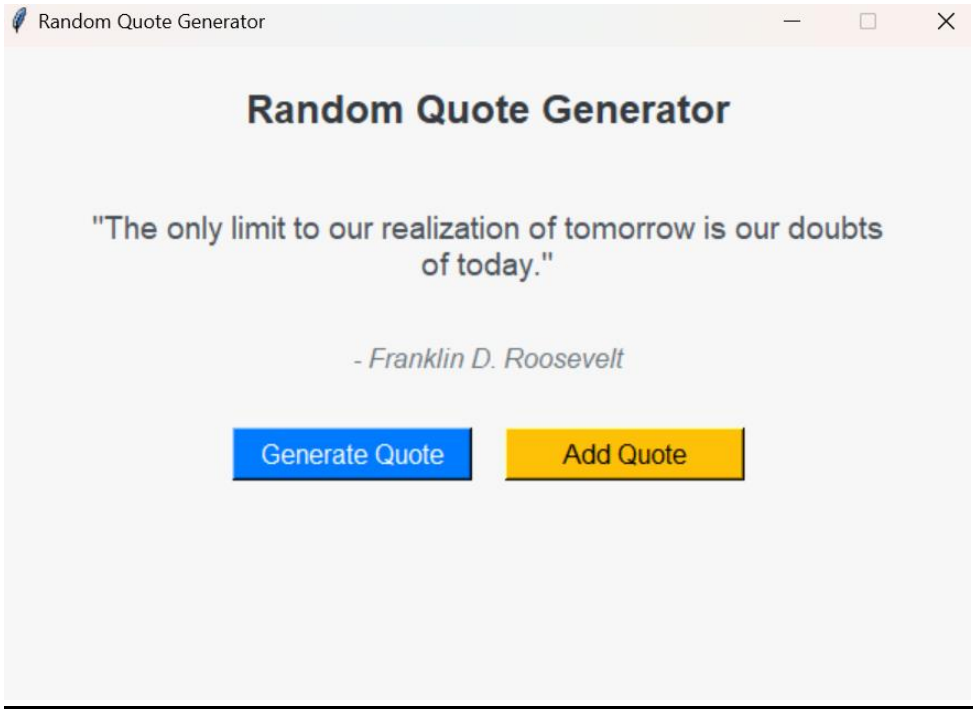
```
main_frame = Frame(root, bg="#f7f7f7")

title_label = Label(main_frame, text="Random Quote Generator", font=("Helvetica", 18, "bold"), bg="#f7f7f7", fg="#343a40")
title_label.pack(pady=20)

quote_label = Label(main_frame, text="", wraplength=500, font=("Helvetica", 14), justify="center", bg="#f7f7f7", fg="#495057")
quote_label.pack(pady=20)

author_label = Label(main_frame, text="", font=("Helvetica", 12, "italic"), justify="center", bg="#f7f7f7", fg="#6c757d")
author_label.pack(pady=10)

button_frame = Frame(main_frame, bg="#f7f7f7")
button_frame.pack(pady=20)

generate_button = Button(button_frame, text="Generate Quote", command=display_quote, font=("Helvetica", 12), bg="#007bff", fg="white", widt
```

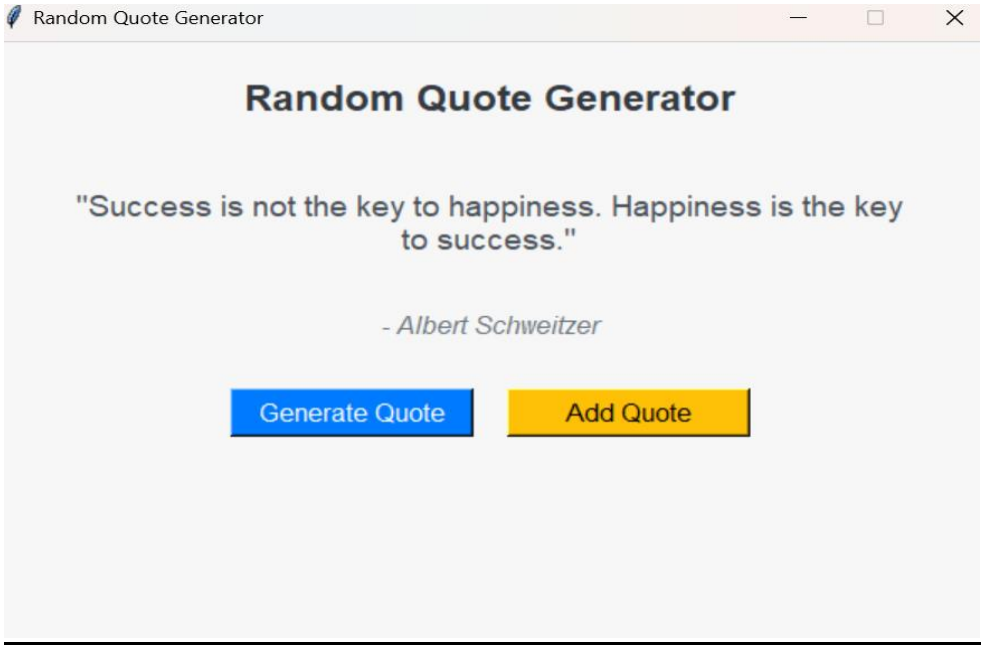
```
generate_button.grid(row=0, column=0, padx=10)

add_button = Button(button_frame, text="Add Quote", command=add_quote, font=("Helvetica", 12), bg="#ffc107", fg="black", width=15)
add_button.grid(row=0, column=1, padx=10)

# Initialize Database
connect_database()

# Run the Tkinter Event Loop
root.mainloop()
```

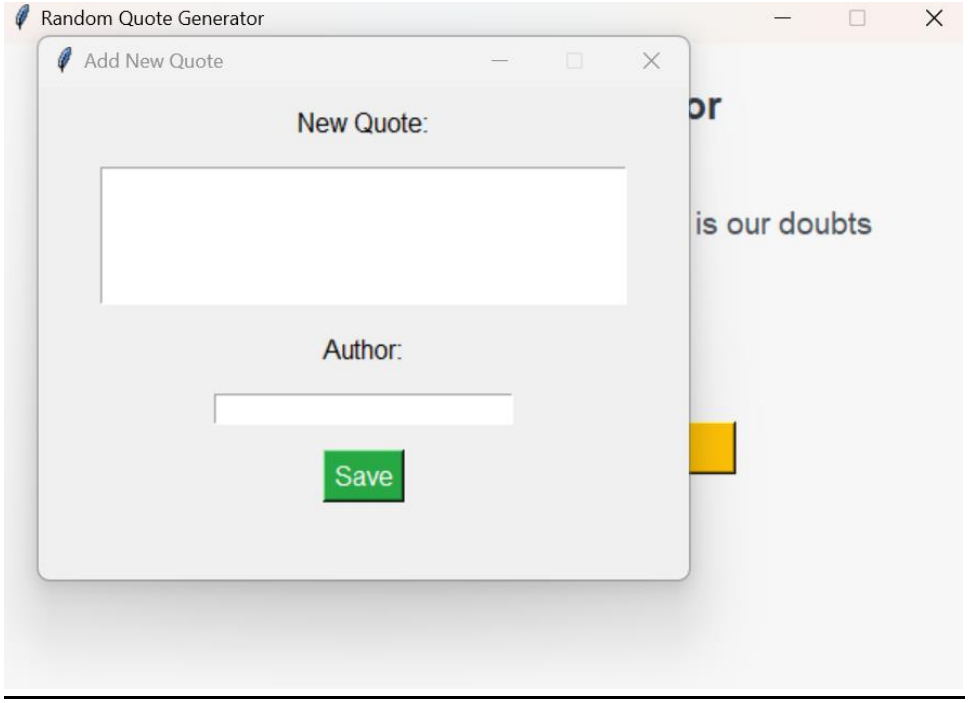
Screenshots of the implementation:




Random Quote Generator
Welcome to the Random Quote Generator
Start


Random Quote Generator
Random Quote Generator
"The only limit to our realization of tomorrow is our doubts of today."
- Franklin D. Roosevelt
Generate Quote
Add Quote


Random Quote Generator
Random Quote Generator
"Success is not the key to happiness. Happiness is the key to success."
- Albert Schweitzer
Generate Quote
Add Quote


Random Quote Generator
Add New Quote
New Quote:
Author:
Save
is our doubts

7. Testing and Validation

Unit Testing Test Cases

Test Case ID: UT_01

• Test Designed by: Sheetal

• Test Designed date: 25/11/24

• Test Title: Start the Application

• Test Execution date: 25/11/24

Description: To start the application successfully.

Step	Test Steps	Test Data	Expected Result	Actual Result	Status	Notes
1	Launch the application.	N/A	The welcome screen is displayed with a "Start" button.	Welcome screen appears with "Start" button visible.	Pass	N/A
2	Click the "Start" button	N/A	The welcome screen is hidden, and the main quote display screen appears.	Welcome screen disappears, and main frame appears.	Pass	Verify that navigation from the welcome screen works correctly.

Test Case ID: UT_02

• Test Designed by: Sheetal

• Test Designed date: 25/11/24

• Test Title: Quote Functionality

• Test Execution date: 25/11/24

Description: To test all the Quote Functionalities

Step	Test Steps	Test Data	Expected Result	Actual Result	Status	Notes
1	Generate a random quote	Click the "Generate Quote" button	A random quote and its author are displayed in the main frame	A random quote and its author appear correctly.	Pass	Ensure quotes are displayed properly, including long quotes with proper wrapping.
2	Add a new quote	New Quote: "Stay positive." Author: "Unknown"	The quote is added successfully to the database. A success message is shown.	Success message appears, quote saved in database.	Pass	Check that valid quotes and authors are added to the database.
3	Attempt to add a quote with empty fields	New Quote: "" Author: ""	Data is added to the database.	A validation warning message is displayed, and no data is added to the database.	Fail	N/A

System Testing Test Cases Test Case ID: ST_01

- Test Designed by: Sheetal
- Test Designed date: 25/11/24
- Test Executed by: Sheetal

- Test Title: System Test for Random Quote Generator Application
- Test Execution date: 25/11/24
- **Description: This test case validates the entire functionality of the Random Quote Generator application, including database initialization, quote generation, adding new quotes, handling invalid inputs, and exiting.**

Step	Test Steps	Test Data	Expected Result	Actual Result	Status	Notes
1	Launch the application and navigate through all functionalities.	**Step 1:** Open the application. **Step 2:** Click the "Start" button. **Step 3:** Click "Generate Quote". **Step 4:** Add a new quote ("Stay positive." by "Unknown"). **Step 5:** Generate another quote. **Step 6:** Test with invalid input (empty fields in "Add New Quote"). **Step 7:** Close the "Add New Quote" window. **Step 8:** Exit the application.	**Step 1:** The welcome screen is displayed. **Step 2:** Main quote display is shown. **Step 3:** A random quote with its author is displayed. **Step 4:** The new quote is added successfully, and a success message appears. **Step 5:** The newly added quote appears randomly. **Step 6:** A warning message appears, and no data is added. **Step 7:** The "Add New Quote" window closes without issues. **Step 8:** The application exits gracefully.	**Step 1:** The welcome screen loaded with a "Start" button. **Step 2:** Main quote display appeared with options. **Step 3:** Random quote "Life is what happens..." by John Lennon displayed. **Step 4:** New quote "Stay positive." added; success message displayed. **Step 5:** Newly added quote displayed randomly. **Step 6:** Warning message "Both quote and author are required" displayed; no changes to the database. **Step 7:** Add Quote window closed successfully. **Step 8:** Application closed without errors.	Pass	Tested all integrated functionalities in one flow. Application is functional as expected.

8. Conclusion

The Random Quote Generator successfully integrates Python, Tkinter, and SQLite to provide a practical and engaging tool for accessing motivational content. The modular design ensures ease of maintenance and scalability for future enhancements.

Key Outcomes:

1. A user-friendly application for accessing and adding quotes.
2. Robust data management using SQLite, ensuring persistence.
3. Modular design to support future extensions, such as categorization or social media sharing.

The Random Quote Generator application was thoroughly tested to validate its functionality, reliability, and usability. The following key aspects were covered during testing:

1. **Database Initialization**:
   The SQLite database (quotes.db) is successfully created on first use, and a sample set of quotes is added automatically if the table is empty. This ensures the application is ready for use upon installation without requiring manual setup.
2. **Quote Display Functionality**:
   The application accurately retrieves and displays random quotes from the database. The displayed quotes are formatted correctly, with proper wrapping for long text.
3. **Add New Quote**:
   Users can add new quotes to the database through a user-friendly interface. The application validates user input, ensuring that both the quote and author fields are filled before submission. Error handling is implemented to notify users of invalid inputs.
4. **Navigation and User Interface**:
   The application transitions smoothly between the welcome screen and the main functionality. The interface is intuitive, visually appealing, and free of major usability issues. Features such as the "Add New Quote" and "Generate Quote" buttons work seamlessly.

5. **Edge Case Handling**:
    - The application handles scenarios such as an empty database by displaying a default message ("No quotes available").
    - Duplicate quotes are allowed, ensuring flexibility for users while storing multiple identical entries.
6. **Performance and Stability**:
   The application remains stable under normal usage and gracefully handles errors, such as invalid input in the "Add New Quote" functionality. The application closes without leaving residual processes or errors.
7. **System Integration**:
   All components, including the database, user interface, and logic for random quote generation and new quote addition, are well-integrated and function cohesively.

9. References

1. Python Documentation: https://docs.python.org/3/
2. SQLite Documentation: https://sqlite.org/docs.html
3. GeeksforGeeks, Tkinter Tutorial: https://www.geeksforgeeks.org/python-gui-tkinter/
4. Real Python, Random Module: https://realpython.com/python-random/
5. Example Discussion on Tkinter Window Management: https://stackoverflow.com/questions/tagged/tkinter