Overview

In this exercise, we created a simple web application using Dash by Plotly. The app functions as a message bank, allowing users to submit messages and view a sample of the messages currently stored in the bank. The app is designed to be visually appealing and user-friendly, incorporating various Dash fundamentals, including callbacks, as well as database management skills.

Submission Functionality

To enable message submissions, the app includes three primary user interface elements: a text box for submitting the message, a text box for submitting the user's name, and a "submit" button. The core of the database management is handled by two Python functions: get_message_db() and insert_message(handle, message).

The get_message_db() function is responsible for creating or connecting to the database of messages. It first checks if a global message_db variable exists. If not, it connects to the SQLite database and creates a table called messages with columns for handle and message if it does not already exist. This function ensures that the database is ready for storing messages.

The insert_message(handle, message) function handles inserting new messages into the database. It uses a cursor to execute an SQL command that inserts the provided handle and message into the messages table. After the insertion, it commits the transaction to ensure the data is saved and closes the database connection.

The submit() callback function is triggered when the user clicks the "submit" button. It extracts the handle and message from the input components and calls insert_message(handle, message) to store the data in the database. If both fields are filled, a thank you message is displayed; otherwise, an error message prompts the user to fill in both fields.

Viewing Random Submissions

To allow users to view random submissions, we implemented the random_messages(n) function, which returns a collection of n random messages from the database. This function connects to the database, executes a query to fetch random messages, and then closes the database connection before returning the results.

The app includes a component to display these messages, and the view() callback function is responsible for updating this component. When the "view" button is clicked, the view() function calls random_messages(5) to fetch up to five random messages and displays them in a list. If no messages are found, a message indicating this is shown instead.

Customization and Styling

To make the app visually appealing, we used Bootstrap for a clean and responsive layout and Google Fonts for a modern font style. The app's main container has a subtle background image, and elements have transparent backgrounds with shadow effects to create a floating effect. Buttons are styled with vibrant colors and smooth transitions to enhance user interaction.

Conclusion

The final app meets the requirements of the assignment, providing a user-friendly interface for submitting and viewing messages, robust database management, and a visually appealing design. The app's code is organized and well-commented, making it easy to understand and maintain.

Key Python Functions

get_message_db(): Creates or connects to the message database.

insert_message(handle, message): Inserts a new message into the database.

random_messages(n): Fetches n random messages from the database.

submit(): Handles the submission of new messages.

view(): Displays random messages from the database.

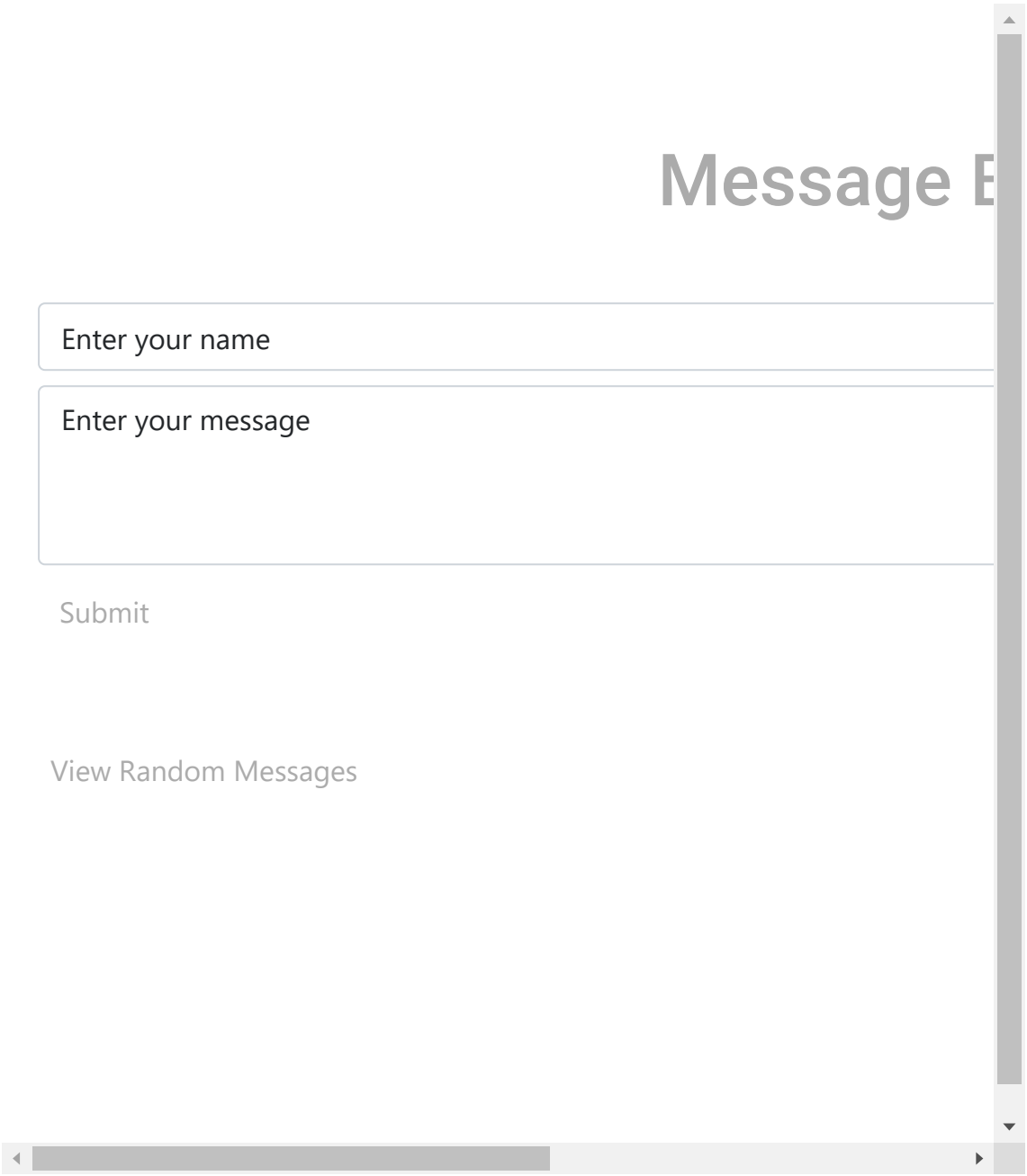
The app runs within a Jupyter Notebook, which also includes the necessary screencaps demonstrating the submission and viewing functionalities.

In [ ]:
```python
# hw6.py
import dash
from dash import dcc, html
from dash.dependencies import Input, Output, State
import sqlite3
import random

# Initialize the Dash app
app = dash.Dash(__name__)
server = app.server

# Global variable to hold the database connection
message_db = None

def get_message_db():
    """
    Create or connect to the message database.

    This function checks whether there is a global `message_db` connection.
    If not, it connects to the SQLite database "messages_db.sqlite" and
    creates the `messages` table if it does not exist.
    """
```

```
    global message_db
    if message_db:
        return message_db
    else:
        # Connect to the SQLite database
        message_db = sqlite3.connect(
            "messages_db.sqlite", check_same_thread=False)
        # Create the messages table if it does not exist
        cursor = message_db.cursor()
        cursor.execute('''
            CREATE TABLE IF NOT EXISTS messages (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                handle TEXT,
                message TEXT
            )
        ''')
        message_db.commit()
        return message_db

def insert_message(handle, message):
    """
    Insert a new message into the database.

    This function takes the user's handle and message as inputs and inserts
    them into the `messages` table in the database. It then commits the
    transaction.
    """
    db = get_message_db()
    cursor = db.cursor()
    cursor.execute('''
        INSERT INTO messages (handle, message) VALUES (?, ?)
    ''', (handle, message))
    db.commit()

def random_messages(n):
    """
    Fetch n random messages from the database.

    This function retrieves `n` random messages from the `messages` table.
    It connects to the database, executes the query to fetch random messages,
    and returns the messages.
    """
    db = get_message_db()
    cursor = db.cursor()
    cursor.execute('''
        SELECT handle, message FROM messages ORDER BY RANDOM() LIMIT ?
    ''', (n,))
    messages = cursor.fetchall()
    return messages

# Define the layout of the Dash app
app.layout = html.Div([
    # Link to Bootstrap CSS and Google Fonts
    html.Link(
        rel='stylesheet',
        href='https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/'
             'bootstrap.min.css'
    ),
    html.Link(
        href='https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;'
```

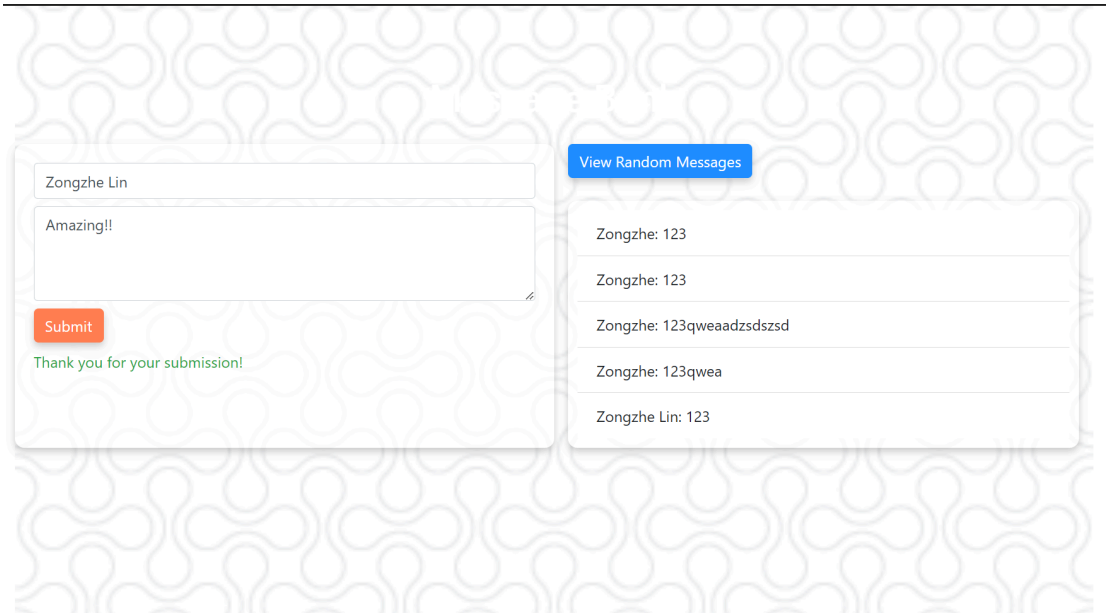
```
            '500&display=swap',
        rel='stylesheet'
    ),
    # Main container
    html.Div([
        html.H1("Message Bank", className='text-center my-4', style={
            'fontFamily': 'Roboto, sans-serif', 'color': '#ffffff'}),
        html.Div([
            # Submission form
            html.Div([
                dcc.Input(id='handle-input', type='text',
                          placeholder='Enter your name',
                          className='form-control mb-2'),
                dcc.Textarea(id='message-input',
                             placeholder='Enter your message',
                             className='form-control mb-2',
                             style={'height': '100px'}),
                html.Button('Submit', id='submit-button', n_clicks=0,
                            className='btn btn-primary mb-2',
                            style={'backgroundColor': '#ff7f50', 'border':
                                   'none', 'borderRadius': '5px',
                                   'boxShadow': '0 4px 8px 0 rgba(0, 0, 0, '
                                                '0.2)', 'transition': '0.3s'}),
                html.Div(id='submit-message', className='text-success')
            ], className='col-md-6 mb-4',
                style={'backgroundColor': 'rgba(255, 255, 255, 0.9)',
                       'padding': '20px', 'borderRadius': '10px',
                       'boxShadow': '0 4px 8px 0 rgba(0, 0, 0, 0.2)',
                       'transition': '0.3s'}),
            # View messages section
            html.Div([
                html.Button('View Random Messages', id='view-button',
                            n_clicks=0, className='btn btn-primary mb-2',
                            style={'backgroundColor': '#1e90ff', 'border':
                                   'none', 'borderRadius': '5px',
                                   'boxShadow': '0 4px 8px 0 rgba(0, 0, 0, '
                                                '0.2)', 'transition': '0.3s'}),
                html.Div(id='message-display', className='mt-3',
                         style={'backgroundColor': 'rgba(255, 255, 255, 0.9)',
                                'padding': '10px', 'borderRadius': '10px',
                                'boxShadow': '0 4px 8px 0 rgba(0, 0, 0, 0.2)',
                                'transition': '0.3s'})
            ], className='col-md-6 mb-4'),
        ], className='row justify-content-center')
    ], className='container', style={
        'backgroundImage': 'url("https://www.toptal.com/designers/'
                           'subtlepatterns/uploads/double-bubble-outline.'
                           'png")', 'backgroundSize': 'cover', 'height':
        '100vh', 'paddingTop': '50px'})
])

@app.callback(
    Output('submit-message', 'children'),
    Input('submit-button', 'n_clicks'),
    State('handle-input', 'value'),
    State('message-input', 'value')
)
def submit(n_clicks, handle, message):
    """
    Handle the submission of a new message.
```

```
    This callback function is triggered when the "submit" button is clicked.
    It retrieves the handle and message from the input fields and inserts the
    message into the database. It then updates the `submit-message` div with
    a thank you message or an error message if fields are empty.
    """
    if n_clicks > 0:
        if handle and message:
            insert_message(handle, message)
            return "Thank you for your submission!"
        else:
            return "Please fill in both fields."
    return ""

@app.callback(
    Output('message-display', 'children'),
    Input('view-button', 'n_clicks')
)
def view(n_clicks):
    """
    Display random messages from the database.

    This callback function is triggered when the "view" button is clicked.
    It fetches up to 5 random messages from the database and displays them
    in a list. If no messages are found, it displays a message indicating
    this.
    """
    if n_clicks > 0:
        messages = random_messages(5)
        if messages:
            return html.Ul([
                html.Li(f"{handle}: {message}", className='list-group-item')
                for handle, message in messages
            ], className='list-group list-group-flush')
        else:
            return "No messages found."
    return ""

# Run the app
if __name__ == '__main__':
    app.run_server(debug=True)
```



Example

Below is the screenshot showing the user submitting a message with the handle "Zongzhe Lin" and the message "Amazing!!". The thank you message confirms the submission.


Zongzhe Lin
Amazing!!
Submit
Thank you for your submission!
View Random Messages
Zongzhe: 123
Zongzhe: 123
Zongzhe: 123qweaadzsdszsd
Zongzhe: 123qwea
Zongzhe Lin: 123