Generative AI

Trainer: Nilesh Ghule

dotenv: Managing Environment Variables Securely

- Problem: Hardcoding API keys, database passwords, and secrets in code is insecure and inflexible
- Solutions:
  - Define secrets in OS environment variables and load in your application
  - Treat properties in .env file as OS env variables and use in your application
- Key Benefits:
  - Security: Keep secrets out of version control *(.env is gitignored)*
  - Flexibility: Different configurations for development, testing, production
  - Portability: Easily share code without exposing credentials

```
# Installation
# pip install dotenv
```

```
# .env file (add to .gitignore!)
# OPENWEATHER_API_KEY=sk-abc123...
```

```
# Using dotenv -- myapp.py
from dotenv import load_dotenv
import os

# Load environment variables from .env file
load_dotenv()  # Loads from .env

# Access environment variables
api_key = os.getenv("OPENWEATHER_API_KEY")

flag = 'Yes' if api_key else 'No'
print(f"API Key loaded: {flag}")

# Use api_key to get the weather
```

SQL on Pandas: Why and How

- Problem: Pandas has powerful DataFrame operations, but SQL is often more intuitive for data querying.
- Solution: Use SQL syntax directly on Pandas DataFrames
- Key Libraries:
  - pandasql: SQLite backend for querying DataFrames
    - **> pip install pandasql**
  - duckdb: High-performance SQL engine with Pandas integration

```
import pandas as pd
import pandasql as ps

sales_data = pd.read_csv("sales.csv")
query = """SELECT customer, SUM(amount) as total,
COUNT(*) as cnt FROM sales
GROUP BY customer HAVING total > 100
ORDER BY cnt DESC
"""

result = ps.sqldf(query, {"sales": sales_data})
print(result)
```

Introduction to Streamlit

- What is Streamlit?
  - **An open-source Python framework for building web apps**
- Perfect for AI Apps
  - Quickly prototype applications without web development knowledge
- Our focus:
  - Chatbot Like UI ✓
  - File uploads ✓
  - User authentication ✓
  - Multi-page apps ✓

Full Stack Web Dev

① Backend
- Database
- Python, Node, Spring, .Net Core, ...

② Frontend
- HTML
- JS
- CSS - Bootstrap, ...
- React or Angular

Streamlit Installation and Basic Setup

- Installation: pip install streamlit
- Basic app structure: Single Python file
- Running apps: streamlit run app.py
- Hot reload: Automatically updates when you save code

```
# Installation
pip install streamlit

# Create your first app
# File: app.py
import streamlit as st

st.title("My First Streamlit App")
st.write("Hello, World!")

# Run it
# streamlit run app.py
```

Core Streamlit Widgets for GenAI Apps

- Input widgets:
  - st.text_input(): Text input
  - st.text_area(): Multi-line text
  - st.file_uploader(): File upload
  - st.selectbox(): Dropdown
  - st.button(): Button
  - st.slider(): Slider control
- Chat widgets
  - st.chat_input(): Text input message
  - st.chat_message(role): Message container with icon for human/ai.

- Display widgets:
  - st.write(): General output
  - st.markdown(): Rich text
  - st.dataframe(): Display tables
  - st.json(): Display JSON
  - st.toast(): Display alert (temp)
  - st.title(): Page title
  - st.header(): Text in header format
  - st.subheader(): Sub-heading
  - st.write_stream(): write generator/stream
  - st.columns(): divide screen in n cols

Core Streamlit Widgets for GenAI Apps

```
# Input widgets
name = st.text_input("Enter your name:")
message = st.text_area("Enter your message:", height=100)
uploaded_file = st.file_uploader("Choose a file", type=['txt', 'pdf', 'csv'])
model = st.selectbox("Choose AI model:", ["GPT-4", "Llama 3", "Gemini", "Claude"])

# Display widgets
if name:
    st.write(f"Hello, {name}!")

st.markdown("**This is bold text** and *this is italic*")

# Display dataframe
import pandas as pd
df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})
st.dataframe(df)

# Display JSON
config = {"model": "gpt-4", "temperature": 0.7, "max_tokens": 500}
st.json(config)
```

Session State: Managing User Data

- Problem: Streamlit reruns entire script on interaction & all previous data is lost
- Solution: st.session_state persists data across reruns.
- Streamlit session state acts like a dictionary that holds information for a single user's session, & its contents are preserved between script executions
- Critical for: Chat history, user preferences, uploaded files
- Usage patterns:
  - Initialize:

```
if 'key' not in st.session_state:
    st.session_state.key = value
```

  - Access:

```
st.session_state.key
```

  - Update:

```
st.session_state.key = new_value
```

Nilesh Ghule

Session State: Managing User Data

```
import streamlit as st

if 'counter' not in st.session_state:
    st.session_state.counter = 0

col1, col2 = st.columns(2)
with col1:
    if st.button("Increment"):
        st.session_state.counter += 1
with col2:
    if st.button("Reset"):
        st.session_state.counter = 0

st.write(f"{st.session_state.counter}")
```

```
import streamlit as st

# Initialize session state
if 'messages' not in st.session_state:
    st.session_state.messages = []

# Display chat history
if st.session_state.messages:
    st.subheader("Chat History")
    for msg in st.session_state.messages:
        st.write(msg)

# input the message and append to history
message = st.chat_input("Say something")
if message:
    st.session_state.messages.append(message)
```

Advanced Controls & Features

```python
with st.sidebar:
    st.header("Settings")
    options = ["Upper", "Lower", "Toggle"]
    case = st.selectbox("Select Case", options)
    count = st.slider("Max Messages", 3, 10, 3, 1)

    st.subheader("Current Conf")
    st.json({"mode": case, "count": count})

    if st.button("Clear History"):
        st.session_state.messages = []
```

```python
data_file = st.file_uploader("Choose
a CSV file", type=["csv"])
if data_file:
    data = pd.read_csv(data_file)
    st.dataframe(data.dtypes)
```

```python
with st.form(key="reg_form"):
    st.header("Your details")
    # form elements inside the form block
    first_name = st.text_input("First Name")
    last_name = st.text_input("Last Name")
    age = st.slider("Age", 18, 100, value=25)
    # form must have a submit button
    submit_btn=st.form_submit_button("Submit")

# form submission outside the 'with' block
if submit_btn:
    st.success(f"Name:{first_name}, Age:{age}")
```

- **streamlit widget key:**
  - Distinguishing Identical Widgets: two widget with same props must have unique key.
  - State mgmt & access: widget added in session state & accessed using st.session_state["key"]

Multi-page application

```
if 'page' not in st.session_state:
    st.session_state.page = "Home"
with st.sidebar:
    if st.button("Home", use_container_width=True):
        st.session_state.page = "Home"
    if st.button("Courses", use_container_width=True):
        st.session_state.page = "Courses"
    if st.button("Internship", use_container_width=True):
        st.session_state.page = "Internship"
    if st.button("About Us", use_container_width=True):
        st.session_state.page = "About Us"
```

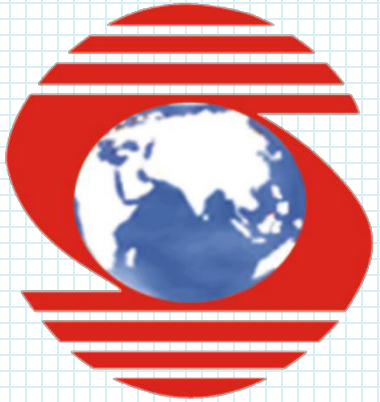
```
def show_home_page():
    st.title("Home")
    # ...

def show_courses_page():
    st.title("Courses")
    # ...
```

```
if st.session_state.page == "Home":
    show_home_page()
elif st.session_state.page == "Courses":
    show_courses_page()
elif st.session_state.page == "Internship":
    show_internship_page()
elif st.session_state.page == "About Us":
    show_aboutus_page()
```

Practice Assignments

1. Upload a CSV file. Input a SQL query from user and execute it on the CSV data (as dataframe). Display result on the page.
2. Show Login Form. If login is successful (fake auth -- if username & passwd is same, consider valid user), show weather page. There input a city name from text box and display current weather information. Provide a logout button and on its click, display thanks message.
3. Make a chat bot like UI. Input a message from user and reply it back, but display the reply using st.write_stream(). Use delay to show chatlike effect.
4. For unauthenticated users, show menu (in sidebar) as Home, Login, Register. Keep login details in users.csv. For authenticated users, show menu explore CSV, See history, Logout. Maintain CSV upload history (userid, csv file name, date-time of upload) into userfiles.csv. Use pandas for reading/writing data CSVs.



Thank You!

Nilesh Ghule

<nilesh@sunbeaminfo.com>