**21/03/2025 - TEST**

1. **Explain the key features of Streamlit that make it suitable for data science and machine learning applications.**

Streamlit is designed for quick and easy development of web applications, especially for data science and machine learning. Its key features include:

* **Rapid prototyping**: You can turn data scripts into interactive web apps with just a few lines of code.
* **Simplicity**: It allows for a high level of abstraction and minimal boilerplate code. You write Python code, and Streamlit automatically handles the front-end.
* **Real-time interactivity**: Streamlit allows users to interact with models or datasets in real-time through widgets like sliders, buttons, and text inputs.
* **Built-in support for visualizations**: It integrates seamlessly with popular libraries like Matplotlib, Plotly, and Altair, making it easier to display data insights.
* **Dynamic updates**: When data changes, Streamlit automatically re-runs parts of the script to reflect updated results.

1. **How does Streamlit handle state management, and what are some ways to persist data across interactions?**

Streamlit uses a simple model for managing state. It offers:

* **Session state**: This allows you to store and manage variables across multiple interactions within the same session (e.g., storing a user’s inputs or the state of a model).
* **Ways to persist data**: You can use st.session\_state to hold variables and make them persist throughout the app's runtime. Additionally, you can persist data externally by saving it to a database or a file system.
* **Stateful widgets**: Streamlit widgets maintain their state across reruns unless explicitly reset by the user. This is useful for complex apps where interaction persists.

1. **Compare Streamlit with Flask and Django. In what scenarios would you prefer Streamlit over these traditional web frameworks?**

* **Streamlit** is a specialized framework for creating data-driven web applications quickly. It is great for building interactive dashboards and visualizations with minimal setup.
* **Flask** and **Django** are full-fledged web frameworks that are more flexible but also more complex. They are designed for general web app development and allow for deeper customization and integration.
* **When to prefer Streamlit**: Choose Streamlit over Flask or Django when you want to create a data-driven app quickly with less need for customization, and when the app focuses heavily on interactive visualizations, machine learning models, or data analysis.
* **When to prefer Flask/Django**: Use Flask or Django if you need more control over the application’s structure, user management, complex routing, or want to integrate complex business logic.

1. **Describe the role of caching (@st.cache\_data and @st.cache\_resource) in Streamlit. How does it improve performance?**

Caching is used in Streamlit to store results of expensive computations or resource-intensive processes, so they don't need to be recomputed each time the app is rerun.

* **@st.cache\_data**: This is used for caching data such as a DataFrame, results from a machine learning model, or API calls. It stores the results to avoid recalculating them repeatedly.
* **@st.cache\_resource**: This caches resources like models or large files, which are expensive to load or initialize.
* **Performance Improvement**: By caching expensive operations, the app becomes faster and more responsive because the results can be retrieved from memory rather than recalculated.

1. **How can you integrate a database with a Streamlit app? Provide an example using SQLite or PostgreSQL.**

Streamlit allows you to integrate a database easily by using Python libraries like sqlite3 or psycopg2 (for PostgreSQL).

**Example with SQLite**:  
import sqlite3

import streamlit as st

# Connect to SQLite database

conn = sqlite3.connect('example.db')

c = conn.cursor()

# Fetch data from the database

c.execute("SELECT \* FROM my\_table")

data = c.fetchall()

# Display data in Streamlit

st.write(data)

1. **Discuss how you can deploy a Streamlit application. Mention at least two deployment platforms.**

You can deploy Streamlit apps on several platforms:

* **Streamlit Sharing**: This is a free, cloud-based platform from Streamlit itself that allows easy deployment directly from GitHub.
* **Heroku**: A popular platform-as-a-service (PaaS) provider where you can deploy Streamlit apps by pushing the app to a Git repository connected to Heroku.
* **AWS, GCP, and Azure**: These cloud services offer more scalable deployment options for production environments.

1. **What are some limitations of Streamlit, and how can you overcome them when building production-grade applications?**

* **Limited flexibility**: Streamlit is great for rapid prototyping, but it may not be ideal for complex web applications that require advanced user management or custom routing.
  + **Solution**: Combine Streamlit with other frameworks like Flask or Django for more control over the app’s logic while maintaining the interactive capabilities of Streamlit.
* **Performance**: In large applications with heavy traffic or very large datasets, performance could be a concern.
  + **Solution**: Use caching, optimize data processing, or scale the app through cloud services for better performance.

1. **Explain the process of creating an interactive dashboard in Streamlit. What components would you use?**

* Use widgets like st.slider, st.selectbox, st.button, and st.text\_input to create interactivity.
* Visual components like charts (st.line\_chart, st.bar\_chart, or custom visualizations using st.pyplot or Plotly) can be used to represent the data.
* Layouts can be managed using st.columns for a responsive design.

1. **How would you implement user authentication in a Streamlit app? Provide possible solutions.**

Streamlit doesn't have built-in authentication, but you can integrate it with external services or use Python libraries:

* **Using streamlit-authenticator**: A simple library for adding authentication to a Streamlit app.
* **Custom solutions**: Use Flask or Django for backend authentication and integrate with Streamlit for the front-end interface.
* **OAuth**: You can integrate OAuth for authentication using Google, GitHub, or other third-party login services.

1. **Describe a real-world use case where you have implemented or would implement a Streamlit application.**

A real-world example would be creating a Streamlit app for predicting customer churn based on historical data. The app could visualize key metrics, allow users to interact with model inputs like customer age, usage patterns, etc., and show predictions in real-time. It could be used by business analysts or customer service teams to make data-driven decisions.