1. **What is Streamlit and what are its main features?**

* Streamlit is an open-source Python library for creating interactive, data-driven web applications. Its main features include:
  + Fast and easy development of web apps using Python scripts.
  + Real-time updates and interactions.
  + Built-in widgets for inputs like sliders, buttons, and text boxes.
  + Support for various data visualization libraries (e.g., Matplotlib, Plotly).
  + Simple deployment process.

1. **How does Streamlit differ from other web application frameworks like Flask or Django?**

* Streamlit focuses on simplicity and rapid prototyping for data apps, while Flask and Django are general-purpose web frameworks with broader functionality for building complex, full-featured web applications. Streamlit requires minimal code to create a functional app, whereas Flask and Django involve more setup and configuration.

1. **What are some typical use cases for Streamlit?**

* Data visualization dashboards.
* Machine learning model deployment.
* Prototyping data analysis tools.
* Interactive reports.
* Real-time data monitoring.

1. **How do you create a simple Streamlit app?**

* Install Streamlit: pip install streamlit.
* Create a Python script (e.g., app.py).
* Import Streamlit: import streamlit as st.
* Use Streamlit functions to build the app:
* st.title("Simple Streamlit App")
* st.write("Hello, Streamlit!")
* Run the app: streamlit run app.py.

1. **Can you explain the basic structure of a Streamlit script?**

* A Streamlit script typically starts with importing the Streamlit library (import streamlit as st), followed by defining the layout and components of the app using Streamlit functions (st.title, st.write, st.slider, etc.). The script runs top to bottom, and updates are handled automatically.

1. **How do you add widgets like sliders, buttons, and text inputs to a Streamlit app?**

Streamlit provides built-in functions to add widgets to your app. Here are examples for sliders, buttons, and text inputs:

import streamlit as st

# Slider

value = st.slider("Select a value", min\_value=0, max\_value=100)

st.write(f"Slider value: {value}")

# Button

if st.button("Click me"):

st.write("Button clicked!")

# Text input

text = st.text\_input("Enter text")

st.write(f"Text input: {text}")

These functions create interactive widgets that users can interact with, and the values can be used within the app for various purposes.

1. **How does Streamlit handle user interaction and state management?**

Streamlit reruns the script from top to bottom with each user interaction. To manage state, you can use Streamlit's session state (st.session\_state), which allows you to store and retrieve values across reruns.

1. **What are some best practices for organizing and structuring a Streamlit project?**

* Modularize code by separating logic into functions and modules.
* Use a clear and descriptive file naming convention.
* Keep the main script simple and focused on layout and interactions.
* Use version control (e.g., Git) for managing changes.

1. **How would you deploy a Streamlit app locally?**

* Ensure all dependencies are installed.
* Run the app with streamlit run app.py.
* Access the app through the local server URL provided by Streamlit

1. **Can you describe the steps to deploy a Streamlit app?**

* Prepare the app and ensure all dependencies are listed in requirements.txt.
* Choose a deployment platform (e.g., Streamlit Sharing, Heroku, AWS).
* For Streamlit Sharing:
  + Push your code to a GitHub repository.
  + Go to Streamlit Sharing, connect your GitHub account, and deploy the repository.

1. **What is the purpose of the requirements.txt file in the context of Streamlit deployment?**

The requirements.txt file lists all the Python dependencies required to run the Streamlit app. During deployment, this file is used to install the necessary packages to ensure the app runs correctly in the deployment environment.