Streamlit - Deployment Module

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

   Streamlit is an open-source Python framework for creating interactive web applications. Its main features are:

   - Easy to use with pure Python.
   - Supports complex visualizations with minimal code.
   - Real-time interactivity.
   - Auto-refresh when code changes.
   - Integration with data science libraries like Pandas, NumPy, and Matplotlib.

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

   Flask and Django are more general-purpose web frameworks for building a wide range of web applications. Streamlit differs from Flask and Django as it is specifically designed for data apps with real-time interactivity.

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

   Typical use cases for streamlit are:

   - Creating machine learning model prototypes.
   - Developing interactive reports.
   - Constructing data exploration tools.
   - Sharing data science results with non-technical stakeholders.

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

   Steps to create a simple Streamlit app is:

   - Install Streamlit with *pip install streamlit.*
   - Create a Python script (e.g., my_app.py).
   - Add Streamlit code, e.g.,

     ```
     import streamlit as st
     st.write("Hello, Streamlit!")
     ```

   - Run the app using *streamlit run my_app.py*

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

   A basic Streamlit script structure includes:

   - Importing Streamlit: *import streamlit as st* (also any other library if needed).
   - Adding app elements like text, data, and widgets using Streamlit functions (st.write(), st.header(), st.slider(), etc.).
   - Organizing the script with functions to manage layout and adding user interaction facility.

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

   To add widgets in Streamlit:

   - Use built-in functions like st.slider(), st.button(), st.text_input(), etc.
   - For example,

```python
import streamlit as st

# Slider widget
slider_value = st.slider("Select a value", 0, 100)

# Button widget
if st.button("Click me"):
    st.write("Button clicked!")

# Text input widget
user_input = st.text_input("Enter text")


st.write(f"Slider value: {slider_value}")
st.write(f"Text entered: {user_input}")
```

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

   Streamlit handles user interaction by re-running the script from top to bottom whenever a widget's state changes.

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

   Best practices for Streamlit project organization are:

   - Modularizing code into functions or classes.
   - Using st.cache for expensive computations.
   - Structuring the project with clear directories for data, scripts, and assets.
   - Commenting in code.

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

   To deploy a Streamlit app locally:

   - Run *streamlit run my_app.py* in the terminal
   - Streamlit will launch a local server(http://localhost:3000/) and provide a URL to access the app in a web browser.

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

    Steps to deploy a Streamlit app:

    - Prepare streamlit app script.
    - Create a `requirements.txt` with necessary dependencies
    - Use a hosting service like Streamlit Cloud, Heroku, etc.
    - Configure deployment settings and deploy through the chosen platform.

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

    The `requirements.txt` file lists all the Python dependencies needed to run your Streamlit app. It ensures that all necessary packages are installed in the deployment environment, maintaining consistency across different setups.