Test

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

Streamlit is an open source framework used by data scientist or Machine learning engineers to build apps. Streamlit allows you to create stunning-looking application with only a few lines of code.

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

Streamlit differs in various ways from other web application which makes it suitable for data application

Ease of use: It is designed to be to be easy of use, even who does not have prior experience in web development can use Streamlit. It provides an api which allows user to create web applications with few lines of code.

Focus on data science: Streamlit is specifically designed for data science and machine learning applications. It has number of built in features that make it easy to develop data science applications.

Speed of development. Streamlit enables users to develop and deploy interactive data applications very quickly. This is because Streamlit automatically handles all of the underlying web development tasks, such as generating HTML, CSS, and JavaScript code

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

Typical use cases for streamlit

1.Text element

2.Data display elements

3. Chart elements

4. Media and layout elements.

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

1. Import Streamlit: Begin by importing the Streamlit library at the top of your Python script.

2. Create the App Title: Set the title of your app using st.title()

3. Add Text: You can include text using st.write().

4. Display User Input: Let’s make it interactive by adding a text input widget that allows users to customize the displayed message.

5. Display the Customized Message: Display the customized message using st.write().

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

A few key files are required for your Streamlit app to successfully build and deploy on Dash Enterprise.

First we threw a few streamlit commands into python script then you run it with streamlit run.

As soon as we run the script a local streamlit server will open and your app will be open in a new tab in your browser. The app is your canvas where we will draw charts, , widgets, tables, and more.

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

1. To add a button:

st.button(label, key=None, help=None, on\_click=None, args=None, kwargs=None, \*, type="secondary", disabled=False, use\_container\_width=False)

2. To add a slider:

st.slider(label, min\_value=None, max\_value=None, value=None, step=None, format=None, key=None, help=None, on\_change=None, args=None, kwargs=None, \*, disabled=False, label\_visibility="visible")

3. To add text inputs:

st.text\_input(label, value="", max\_chars=None, key=None, type="default", help=None, autocomplete=None, on\_change=None, args=None, kwargs=None, \*, placeholder=None, disabled=False, label\_visibility="visible")

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

We define access to a Streamlit app in a browser tab as a session. For each browser tab that connects to the Streamlit server, a new session is created. Streamlit reruns your script from top to bottom every time you interact with your app. Each reruns takes place in a blank slate: no variables are shared between runs.

Session State is a way to share variables between reruns, for each user session. In addition to the ability to store and persist state, Streamlit also exposes the ability to manipulate state using Callbacks. Session state also persists across pages inside a multipage app.

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

best practices for organizing and structuring a Streamlit project

1.Separating Input and Output Data Locations

2. Adding Multiple Pages to a Streamlit App

3. Creating Components For Reusability

4. Splitting Functional Code from UI Code

5. Cookiecutter

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

1.Install streamlit:

pip install streamlit

2.Streamlit simple app:

create a file with your app app.py

3. Running the app locally:

streamlit run streamlit\_app.py

4. 6. requirements.txt for Streamlit app¶

streamlit==1.34.0

5. run Streamlit app in production¶

Create a file run.sh We'll need this file to pass some arguments to the streamlit command when deploying the app

6. 8. Push your Streamlit app to GitHub:

Create initialize git and push to GitHub Now we need to initialize git repository, commit it and push to GitHub. First let's create a .gitignore file and put env there, to avoid accidentally committing our virtual environment:

git init

git add .

git commit -m'Initial commit'

git remote add origin git@github.com:appliku/demostreamlit.git

git push -u origin master

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

The steps to deploy a Streamlit app:

Step 1. Create a simple Streamlit app

Step 2. Set up an account on Community Cloud

Step 3. Connect your account to GitHub

Step 4. Create a GitHub repo of your app

Step 5. Deploy your app in a few clicks

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

requirements.txt’ file which specifies the packages to be installed. The versions of the packages must be specified to avoid the ‘it works on my machine’ problem. The version of the packages must match with the ones in our Conda environment.