Pros of Streamlit:

1. **Quick Development:**
    - It allows rapid prototyping of web applications in just a few lines of Python code.
2. **Simple and Easy:**
    - Accessible to beginners in web development, no need HTML, CSS, or JavaScript. Everything is written in Python.
3. **Real-time Interactivity:**
    - It automatically updates the app whenever the user interacts with it, without refreshing the page.
    - It has sliders, graphs, and input making it great for data exploration and dashboards.
4. **Supports DS and ML:**
    - Libraries like Pandas, Numpy, Matplotlib, etc are supportable to visualize and display data-driven models.
5. **Built-in Components:**
    - Like charts, tables, maps, file upload/download, etc., which means we don't need to build these from scratch.
6. **Deployment is Easy:**
    - It provides simple deployment options. We can deploy our app with just a few commands, or use Streamlit sharing platform to host the app.

Cons of Streamlit:

1. **Limited Customization:**
    - It lacks the flexibility and customization options of traditional web frameworks like Flask or Django.
    - If we need highly customized UI components, Streamlit may not be the best choice.

2. **Not Ideal for Large Applications:**
    - It is designed for small to medium-sized applications. If our application needs complex routing, user authentication, or large-scale functionality, go to other frameworks like Flask, Django
3. **Performance Issues for Complex Applications:**
    - It's automatic re-execution on every user interaction can lead to performance issues for complex applications or large datasets.
    - The app may also reload unnecessarily, leading to a poor user experience.
4. **Limited Control Over the UI:**
    - Customizing the look and feel of your application beyond Streamlit's built-in components can be difficult. It's not as flexible in terms of UI/UX as other web frameworks.
    - We can inject custom CSS and JavaScript, but it's not as seamless as in other frontend frameworks.
5. **Limited Backend Support:**
    - Streamlit is mainly focused on the frontend and user interface and does not offer much in terms of handling backend logic, authentication, or database management.
    - For more complex apps, we'll need to integrate Streamlit with other backend technologies like Flask or Django.
6. **Single-Page Nature:**
    - Streamlit applications are designed as single-page apps, so if our app requires multiple pages with complex navigation, it is not recommend to implement
    - Multi-page apps are possible with some tricks(like query params), but it's not as straightforward as in other web frameworks.
7. **Limited Community and Ecosystem:**
    - While Streamlit's community is growing, it is still smaller compared to more established web frameworks like Flask or Django.
    - There are fewer tutorials, plugins, and third-party resources available for Streamlit compared to larger web development ecosystems.

When to Use Streamlit:

- **Prototyping**: It's perfect for quickly building and sharing interactive prototypes, especially for data visualization and ML applications.
- **Data Exploration**: Ideal for creating interactive dashboards to explore datasets and share insights with others.
- **Small Applications**: It works well for small, standalone applications or demos, particularly those focusing on data science, machine learning, or visualizations.

When Not to Use Streamlit:

- **Large-scale Applications**: If we need complex backend logic, user authentication, or multi-page applications with extensive routing, Streamlit might not