**Section 1: What is Flask? Why use Flask over Django? (10 Tasks)**

1. Write a short note (in your own words) on what Flask is.

**Flask** is a lightweight and flexible web framework for building web applications in Python. It’s known for its simplicity and ease of use, making it a popular choice for beginners and small projects.

| **Feature** | **Flask** | **Django** |
| --- | --- | --- |
| **Framework Type** | Micro-framework | Full-stack framework |
| **Built-in Features** | Minimal; requires third-party extensions | Includes ORM, admin panel, authentication, etc. |
| **Flexibility** | Highly flexible and customizable | More structured and follows set conventions |
| **Learning Curve** | Easier for beginners due to simplicity | Steeper learning curve due to many built-in features |
| **Project Size** | Best for small to medium projects or APIs | Ideal for large, complex applications |

2. List 5 differences between Flask and Django in a table format.

### 3. Research and write 3 pros and 3 cons of using Flask.

### Pros of Using Flask:

1. **Lightweight and Flexible**  
   Flask provides only the essential components to build a web app, giving developers full control over how they want to structure and extend their application.
2. **Easy to Learn and Use**  
   Flask’s simple syntax and minimal setup make it beginner-friendly, especially for developers new to web development or Python.
3. **Extensible with Many Plugins**  
   Flask has a large ecosystem of extensions (e.g., for databases, authentication, or form validation) that allow you to add features as needed without extra overhead.

### Cons of Using Flask:

1. **Lacks Built-in Tools**  
   Unlike Django, Flask does not include tools like an ORM, admin interface, or form handling by default—developers must add these manually.
2. **Less Suitable for Large Projects**  
   Because it lacks a fixed project structure, managing large and complex applications in Flask can become difficult without careful organization.
3. **More Responsibility on the Developer**  
   With its flexibility comes more responsibility—developers must make more architectural decisions and handle integrations that Django handles automatically.
4. Explain with examples which types of projects are best suited for Flask.  
   \

* **RESTful APIs**
* **Prototypes and MVPs (Minimum Viable Products)**
* **Single-Page Applications (SPAs) with JavaScript Frontend**
* Internal Tools or Dashboards
* Educational Projects and Tutorials

1. Compare Flask and Django based on project setup complexity.

| **Aspect** | **Flask** | **Django** |
| --- | --- | --- |
| **Initial Setup** | Very simple; you can start with a single .py file | More complex; comes with a default project structure |
| **Files Generated at Start** | Minimal – often just app.py and requirements.txt | Generates multiple files and directories (settings.py, urls.py, etc.) |
| **Configuration** | Manual; you add only what you need | Pre-configured with many default settings and tools |
| **Learning Curve (Setup)** | Easier for beginners to get started | Requires understanding the project structure and conventions |
| **Customization** | Fully customizable from the ground up | Customization requires overriding built-in components |

6. Describe the role of WSGI in Flask applications.  
**WSGI (Web Server Gateway Interface)** is a specification that defines how web servers communicate with Python web applications. It acts as a **bridge between the web server and the Flask application**.

7. List and explain 5 companies or platforms that use Flask in production.

* Netflix
* Airbnb
* Reddit
* Lyft
* Pinterest

8. Create a mind map comparing Flask and Django's architecture.  
Flask vs Django

│

├── 🔧 Framework Type

│ ├── Flask: Micro-framework

│ └── Django: Full-stack framework

│

├── 🏗️ Project Structure

│ ├── Flask:

│ │ ├── Minimal and customizable

│ │ └── Developer defines structure

│ └── Django:

│ ├── Predefined project layout

│ └── Built-in apps (admin, auth, etc.)

│

├── 📦 Built-in Components

│ ├── Flask:

│ │ ├── Routing

│ │ ├── Templates (Jinja2)

│ │ └── Minimal defaults

│ └── Django:

│ ├── ORM

│ ├── Admin panel

│ ├── Authentication

│ ├── Middleware

│ └── Template system

│

├── 🔗 Extensibility

│ ├── Flask:

│ │ ├── Use third-party extensions

│ │ └── Highly flexible

│ └── Django:

│ ├── Rich plugin ecosystem

│ └── Customization through settings and apps

│

├── ⚙️ Request Handling

│ ├── Flask:

│ │ └── WSGI-based with Werkzeug

│ └── Django:

│ └── WSGI-based with its own handler stack

│

├── 👨‍💻 Use Cases

│ ├── Flask:

│ │ ├── Microservices

│ │ ├── APIs

│ │ └── MVPs, small apps

│ └── Django:

│ ├── Large applications

│ ├── Admin-heavy sites

│ └── Complex data-driven platforms

9. List reasons why Flask is preferred for beginners.

* Simple and Minimalistic
* Easy to Setup
* Pythonic and Readable
* Great for Learning Web Fundamentals
* Highly Flexible and Unopinionated
* Excellent Documentation and Community Support
* Easy Transition to Larger Frameworks

10. Choose a sample use-case (like a TODO app) and explain why Flask would be a better choice than Django.

* + Add tasks
  + View a list of tasks
  + Mark tasks as complete
  + Delete tasks