Flask Interview Questions & Answers
Basic Level (For Beginners / 1–2 Years)

Q1. What is Flask?
A: Flask is a lightweight, open-source Python web framework used to build web applications and APIs. It is called a micro-framework because it does not require particular tools or libraries, but can be extended with plugins.

Q2. What are the key features of Flask?

- Lightweight and simple
- Built-in development server and debugger
- RESTful request dispatching
- Jinja2 templating engine
- WSGI compliant
- Extensible with third-party libraries

Q3. How do you install Flask?

pip install flask

Q4. Write a simple "Hello World" Flask app.

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello, World!"

if __name__ == "__main__":
    app.run(debug=True)
```

Q5. What is the difference between Django and Flask?

- Django: Full-stack framework with built-in features.
- Flask: Lightweight, flexible, micro-framework.

Q6. How do you handle routing in Flask?

```
@app.route("/about")
def about():
   return "About Page"
```

Q7. How do you return JSON response in Flask?

```
from flask import jsonify
@app.route("/api")
def api():
   return jsonify({"message": "success"})
```

Q8. How do you run Flask in debug mode?

```
export FLASK_ENV=development
flask run
```

(or app.run(debug=True)).

Q9. What is Jinja2 in Flask?
A: Jinja2 is the templating engine used to render HTML dynamically with placeholders ({{ }} for variables, {% %} for logic).

Q10. How do you manage sessions in Flask?
A: Using Flask's session object, which stores data on the server and sends a session ID cookie to the client.

Intermediate / Advanced (3+ Years Experience)

Q11. How do you connect Flask with a database?

- Flask-SQLAlchemy for SQL databases
- PyMongo for MongoDB

```
from flask_sqlalchemy import SQLAlchemy
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///test.db'
db = SQLAlchemy(app)
```

Q12. Explain Flask request and response objects.

- request: Holds HTTP request data (form, JSON, headers).
- response: Object returned by view functions (HTML, JSON, etc.).

Q13. What are Flask Blueprints? Why are they used?
Blueprints allow splitting large applications into modular components.

```
from flask import Blueprint

user_bp = Blueprint("user", __name__)

@user_bp.route("/profile")
def profile():
   return "User Profile"
```

Q14. How do you handle authentication in Flask?

- Flask-Login → session-based authentication
- Flask-JWT-Extended → token-based authentication

Q15. How do you deploy a Flask application in production?

- Use Gunicorn or uWSGI as WSGI server
- Behind a reverse proxy like Nginx
- Set debug=False
- Use virtual environments and proper logging

Q16. What is the difference between Flask and FastAPI?

- Flask: Older, synchronous by default, simple.
- FastAPI: Modern, async support, auto-docs with Swagger, higher performance.

Q17. How do you implement middlewares in Flask?

```
@app.before_request
def before():
   print("Before each request")

@app.after_request
def after(response):
   print("After each request")
   return response
```

Q18. How do you handle error handling in Flask?

```
@app.errorhandler(404)
def page_not_found(e):
   return "Page Not Found", 404
```

Q19. How do you scale Flask applications?

- Use Gunicorn with multiple workers
- Deploy with Docker/Kubernetes
- Add caching (Redis, Memcached)
- Enable DB connection pooling

Q20. What are Flask extensions you have used?

- Flask-SQLAlchemy (DB integration)
- Flask-Migrate (DB migrations)
- Flask-WTF (forms)
- Flask-JWT-Extended (JWT auth)
- Flask-Mail (emails)