1.What is Flask? Why is it called a microframework?

Flask is a **lightweight Python web framework** used to build web applications and APIs. Flask is a **microframework** because it provides the essential tools to build web applications without including extra features like authentication, database abstraction, or form validation by default keeping it **lightweight**.

2. Compare Flask, Django, and FastAPI.

 **Django**: Full-featured, comes with ORM, admin panel, and built-in tools → best for large, complex projects.

 **Flask**: Minimal, lightweight, flexible → best for small/medium apps where you pick your own components.

 **FastAPI**: Modern, fast, async-ready, with automatic validation → best for APIs and high-performance apps.

3. What is @app.route() in Flask and why do we use it?

 **@app.route()** is a **decorator** that maps a **URL path** to a specific **function** (called a view).

 It tells Flask: *“When a user visits this URL, run this function and return its result.”*

 We use it to define the **routes/endpoints** of our web application.

* 4. What is the difference between path parameters and query parameters in Flask?  
  **Path Parameter**
  + Part of the **URL path** itself.
  + Used to identify a specific resource.
  + Example: /users/101 → here 101 is the path parameter (user ID).
  + URL pattern: /users/<id>
* **Query Parameter**
  + Added **after ?** in the URL.
  + Used for filtering, sorting, or providing extra info.
  + Example: /users?id=101&active=true → here id=101 and active=true are query parameters.
  + URL pattern: /users?key=value

👉 In short:

* **Path parameter** = *“Which resource?”*
* **Query parameter** = *“How to filter/modify the resource?”*

5. How do you access query parameters inside a Flask route?

In Flask, you access **query parameters** using request.args.

6. What does debug=True do in app.run()?

In Flask, setting **debug=True** in app.run() does two main things:

* **Enables Debugger** → If your code throws an error, Flask shows an interactive debugger in the browser with the full error traceback.
* **Enables Auto-Reload** → Any time you change your code and save, the server restarts automatically so you don’t have to stop and start it manually.

👉 Example:

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

⚠️ Important: Use debug=True **only in development**, never in production (it can expose sensitive information).

7. What is the purpose of the request object in Flask?  
In Flask, the **request** object represents the **incoming HTTP request** from the client.

**Purpose:**

* To **access data sent by the client**, such as:
  + **Form data** → request.form
  + **Query parameters** → request.args
  + **JSON data** → request.json
  + **Headers** → request.headers
  + **Cookies** → request.cookies
* To get **request metadata**, like HTTP method (request.method) or URL (request.url)

8. How would you explain the Request → Response Lifecycle in Flask?

 **Client Sends Request** → A user’s browser or API client makes an HTTP request to your Flask app (e.g., GET /home).

 **Flask Receives Request** → Flask server catches the request and creates a **request object** containing all the data (URL, headers, form data, JSON, etc.).

 **Routing** → Flask looks at @app.route() decorators to **match the request URL** to the correct view function.

 **View Function Executes** → The matched function runs, processes data, interacts with databases or services if needed.

 **Response Created** → The view function **returns a response** (string, HTML, JSON, or a Response object).

 **Flask Sends Response** → Flask sends the HTTP response back to the client with **status code, headers, and body**.

Client sends request → Flask matches route → View function runs → Response returned → Client receives response.

9. What is type hinting in Python? Show an example we used in class.  
**Type hinting** in Python is a way to **indicate the expected data type** of a variable, function parameter, or return value. It **doesn’t enforce types** at runtime but helps with **code readability, debugging, and IDE suggestions**.

10. Write a Python function to check if a number is prime.  
def is\_prime(n: int) -> bool:

"""Check if a number is prime."""

if n <= 1:

return False

if n == 2:

return True

if n % 2 == 0:

return False

for i in range(3, int(n\*\*0.5) + 1, 2):

if n % i == 0:

return False

return True

# Example usage

print(is\_prime(7)) # True

print(is\_prime(10)) # False