**SMART INTERNZ - APSCHE**

**AI / ML Training**

**Assessment 3.**

**1. What is Flask, and how does it differ from other web frameworks?**

**Ans.** Flask is a lightweight and flexible web framework for Python designed to make it easy to build web applications quickly and efficiently. It is classified as a microframework because it does not require particular tools or libraries and only provides the core components needed for web development.

In contrast to Flask, other web frameworks like Django provide a more comprehensive set of tools and conventions, making them suitable for larger projects with complex requirements.

**2. Describe the basic structure of a Flask application.**

**Ans.** The basic structure of a Flask application includes setting up the application instance, defining routes and views, managing templates and static files, configuring the application, optionally defining models and using extensions, organizing code with blueprints, and testing and deploying the application as needed.

**3. How do you install Flask and set up a Flask project?**

**Ans.**

**1. Install Flask:**

**🡪** Open your terminal or command prompt.

🡪If you haven't already installed Python, download and install it from the official Python website: [https://www.python.org/downloads/](https://www.python.org/downloads/" \t "_new)

🡪Once Python is installed, you can install Flask using pip, which is Python's package installer. Run the following command:

**Pip install Flask**

**2. Set up a Flask project:**

**🡪**Create a new directory for your Flask project. You can name it anything you like.

🡪Navigate into the newly created directory using the terminal or command prompt.

🡪Inside this directory, create a Python file. This will be your Flask application file. You can name it **app.py** or any other suitable name.

**4. Explain the concept of routing in Flask and how it maps URLs to Python functions.**

**Ans.** In Flask, routes are defined using the ‘**@app.route()’** decorator, where **app** is the Flask application instance. The ‘**@app.route()’** decorator takes the URL pattern as an argument and associates it with a Python function that handles requests to that URL.

**from flask import Flask**

**app = Flask(\_\_name\_\_)**

**@app.route('/')**

**def index():**

**return 'Hello, World!'**

Flask provides a ‘**url\_for()’** function that helps generate URLs dynamically based on route names and parameters. This is particularly useful for creating links within templates or when redirecting to specific routes.

**from flask import url\_for**

**@app.route('/profile')**

**def profile():**

**return 'User Profile'**

**with app.test\_request\_context():**

**print(url\_for('profile'))**

**5. What is a template in Flask, and how is it used to generate dynamic HTML content?**

**Ans.** Templates are typically stored in a directory named "templates" within the Flask application directory. You can create a template file using any text editor or an integrated development environment (IDE). Flask uses the Jinja2 templating engine, which provides features for template inheritance, variable interpolation, control structures, and more.

By using templates in Flask, developers can create dynamic HTML content with ease, separate presentation from logic, reuse code, and build maintainable and scalable web applications.

**6. Describe how to pass variables from Flask routes to templates for rendering.**

**Ans.** In Flask, passing variables from routes to templates for rendering involves using the **‘render\_template()’** function along with passing the variables as arguments.

**7. How do you retrieve form data submitted by users in a Flask application?**

**Ans.** In Flask, you can retrieve form data submitted by users using the **request** object, which provides access to incoming request data such as form data, query parameters, headers, and more. To retrieve form data, you typically handle the POST request sent by the form submission and extract the form data from the request object.

**8. What are Jinja templates, and what advantages do they offer over traditional HTML?**

**Ans.** Jinja templates are a powerful feature of the Flask web framework that allows developers to generate dynamic content in HTML files using a syntax similar to Python. Jinja is a templating engine used by Flask, inspired by Django's templating system, which helps separate the presentation layer from the application logic.

Jinja templates offer a more dynamic and flexible approach to generating HTML content compared to traditional static HTML files. They provide powerful features for template inheritance, variable interpolation, control structures, error handling, and integration with Flask, making them an essential tool for building modern web applications with Flask.

**9. Explain the process of fetching values from templates in Flask and performing arithmetic**

**calculations.**

**Ans.** In Flask, you can fetch values from templates and perform arithmetic calculations by passing the necessary data from your routes to the templates and using Jinja templates' syntax for variable interpolation and expressions.

Steps:

**🡪 Pass Data from Flask Routes to Templates**

**🡪 Access Variables in the Template**

**🡪Perform Arithmetic Calculations**

**10. Discuss some best practices for organizing and structuring a Flask project to maintain**

**scalability and readability.**

**Ans.** Some best practices for organizing and structuring a Flask project to maintain

scalability and readability.

🡪 **Modularize with Blueprints**

**🡪Organize Static Files and Templates**

**🡪** **Use Configuration Files**

**🡪** **Implement Error Handling**

**🡪** **Testing and Quality Assurance**