ASSESSMENT -3

1. What is Flask, and how does it differ from other web frameworks?
2. Describe the basic structure of a Flask application.
3. How do you install Flask and set up a Flask project?
4. Explain the concept of routing in Flask and how it maps URLs to Python functions.
5. What is a template in Flask, and how is it used to generate dynamic HTML content?
6. Describe how to pass variables from Flask routes to templates for rendering.
7. How do you retrieve form data submitted by users in a Flask application?
8. What are Jinja templates, and what advantages do they offer over traditional HTML?
9. Explain the process of fetching values from templates in Flask and performing arithmetic
1. calculations.
10. Discuss some best practices for organizing and structuring a Flask project to maintain
2. scalability and readability.

ANSWERS:

1. Flask is a lightweight web framework for Python that is designed to be simple and easy to use. It differs from other web frameworks by its minimalist design, which allows developers to have more flexibility and control over the structure of their applications. Flask is also modular, allowing developers to add or remove features as needed.
2. A basic Flask application consists of a Python script that defines the application instance, routes, and other configurations. It typically includes importing Flask, creating an instance of the Flask class, defining routes using decorators, and running the application using the **run** method.
3. To install Flask, you can use pip, the Python package installer. Simply run pip and install Flask in your terminal. To set up a Flask project, create a directory for your project, create a virtual environment, install Flask, and then create a Python script for your application.
4. Routing in Flask refers to the mechanism of mapping URLs to Python functions that handle the requests. This is done using the @app.route() decorator, where the argument is the URL path. When a request is made to a URL, Flask matches the URL to the corresponding route and executes the associated Python function.
5. A template in Flask is an HTML file that contains placeholders for dynamic content. These placeholders are typically variables or control structures provided by the Jinja templating

engine. Templates are used to generate dynamic HTML content that is sent to the client based on the data provided by the application.

6. To pass variables from Flask routes to templates, you can use the render template function provided by Flask. Simply pass the variables as keyword arguments to the function, and they will be available in the template for rendering.
7. Form data submitted by users in a Flask application can be retrieved using the request object provided by Flask. The request. form attribute contains a dictionary-like object that contains the submitted form data.
8. Jinja templates are a powerful feature of Flask that allows you to create dynamic HTML content using a syntax similar to Python. Jinja templates offer advantages over traditional HTML by allowing you to use control structures, variables, and filters to generate HTML content dynamically.
9. To fetch values from templates in Flask and perform arithmetic calculations, you can use Jinja's syntax for expressions and control structures. For example, you can use the '**{{ }}**' syntax to output variables in the template and use the **{% %}** syntax for control structures like loops and conditionals.
10. Some best practices for organizing and structuring a Flask project include separating concerns by using a modular structure, using blueprints for managing application components, using configuration files for environment-specific settings, and following the PEP 8 style guide for Python code.