ASSIGNMENT - 3

1. Flask and its Differences from Other Web Frameworks:
- Flask is a micro-framework, meaning it provides the essentials for web development without imposing too much structure or dependencies.
- Other web frameworks like Django come with a lot of built-in features and follow the "batteries-included" philosophy, providing a more comprehensive solution out of the box.
- Flask, on the other hand, allows developers to choose and integrate components as needed, making it lightweight and flexible.
- Flask is more suitable for small to medium-sized projects or when developers prefer more control over the components used in their application.

2. Basic Structure of a Flask Application:
- A Flask application typically consists of a main Python script (usually named `app.py`) where you define your Flask application instance.
- You can have optional folders like `static` for static files (e.g., CSS, JavaScript, images) and `templates` for HTML templates.
- Flask applications are built around routes, which are URL patterns mapped to Python functions (view functions) responsible for handling requests and generating responses.

3. Installing Flask and Setting Up a Flask Project:
- You can install Flask using pip, the Python package manager, by running `pip install Flask`.
- After installing Flask, you can create a new Flask project by creating a directory for your project, navigating to that directory in your terminal, and then creating a Python script (usually `app.py`) to start building your application.

4. Routing in Flask:
- Routing in Flask involves mapping URLs to specific Python functions (view functions) using the `@app.route()` decorator.
- The route decorator specifies the URL pattern for which the decorated function should be invoked.
- For example, `@app.route('/hello')` maps requests to the URL `/hello` to the decorated view function.

5. Templates in Flask:
- Templates in Flask are HTML files with placeholders for dynamic content.

- Flask uses the Jinja2 templating engine to render templates and insert dynamic data into HTML pages.
- Templates allow developers to generate HTML content dynamically by passing data from the Flask application to the template.

6. Passing Variables from Flask Routes to Templates:
- Variables can be passed from Flask routes to templates for rendering by including them as arguments when rendering the template using the `render_template()` function.
- For example, `render_template('index.html', name=name)` would pass a variable named `name` to the `index.html` template.

7. Retrieving Form Data in Flask:
- Form data submitted by users in a Flask application can be retrieved using the `request` object, which is available within Flask routes.
- Form data can be accessed using `request.form['key']` for POST requests or `request.args['key']` for GET requests.

8. Jinja Templates and Their Advantages:
- Jinja templates are HTML files with placeholders for dynamic content that are processed by the Jinja2 template engine.
- Jinja templates offer advantages over traditional HTML by allowing for the insertion of dynamic data, including conditional statements, loops, and other programming constructs directly within HTML files.

9. Fetching Values from Templates and Performing Arithmetic Calculations:
- Jinja syntax is used to fetch values from templates in Flask.
- Variables are inserted into templates using double curly braces `{{ }}`, and control flow statements are enclosed in `{% %}`.
- Arithmetic calculations can be performed directly within Jinja templates using standard Python syntax.

10. Best Practices for Organizing and Structuring a Flask Project:
- Use blueprints to modularize and organize routes, especially for larger projects.
- Separate concerns by using separate files for views, models, forms, and other components.
- Use configuration files for environment-specific settings and sensitive information.
- Follow a consistent naming convention for files and folders to maintain readability.
- Utilize version control (e.g., Git) to track changes and collaborate with other developers.

- Adhere to PEP 8 style guidelines for Python code to ensure consistency and maintainability.