**FLASK ASSIGNMENT**

**PW SKILLS**

Q1. Flask is a lightweight web application framework for Python. It is designed to be simple and easy to use, yet powerful enough to build complex web applications. Flask provides tools and libraries for web development, including routing, templating, and form handling, while allowing developers the flexibility to choose their preferred libraries and tools for other components like databases.

Advantages of Flask Framework:

- Lightweight and easy to use: Flask has a simple and intuitive syntax, making it easy for beginners to get started with web development.

- Flexible: Flask allows developers to choose the components they need for their applications, rather than imposing a rigid structure.

- Extensible: Flask is highly extensible, with a large ecosystem of extensions and libraries available to add additional functionality as needed.

- Built-in development server: Flask comes with a built-in development server, making it easy to test and debug applications during development.

- Integrated unit testing support: Flask provides support for unit testing out of the box, making it easy to write and run tests for applications.

- RESTful request dispatching: Flask supports RESTful request dispatching, making it easy to build RESTful APIs.

Q2. Here's a simple Flask application to display "Hello World!!":

```python

from flask import Flask

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

@app.route('/')

def hello():

return 'Hello World!!'

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

```

To run this application, save it in a file named `app.py` and execute it in your terminal or command prompt using the command `python app.py`. You'll see the "Hello World!!" message displayed in your browser.

Q3. App routing in Flask refers to the mechanism of mapping URLs to view functions in Flask applications. We use app routes to define the URLs and their corresponding view functions, which handle requests made to those URLs. App routing allows us to create different routes for different parts of our application, making it easy to organize and structure our code.

Q4. Here's a Flask application with two routes:

```python

from flask import Flask

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

@app.route('/welcome')

def welcome():

return 'Welcome to ABC Corporation'

@app.route('/')

def details():

return '''

<h1>Company Name: ABC Corporation</h1>

<p>Location: India</p>

<p>Contact Detail: 999-999-9999</p>

'''

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

```

To access the `/welcome` route, you can visit `http://localhost:5000/welcome`, and to access the `/` route, you can visit `http://localhost:5000/`.

Q5. In Flask, the `url\_for()` function is used for URL building. It generates a URL to the given endpoint with the method provided. Here's a Python code to demonstrate the working of the `url\_for()` function:

```python

from flask import Flask, url\_for

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

@app.route('/')

def index():

return 'Index Page'

@app.route('/hello')

def hello():

return 'Hello, World'

@app.route('/user/<username>')

def profile(username):

return f'{username}\'s profile'

with app.test\_request\_context():

print(url\_for('index'))

print(url\_for('hello'))

print(url\_for('hello', next='/'))

print(url\_for('profile', username='John Doe'))

```

When you run this code, it will print out the URLs for the specified endpoints, demonstrating how `url\_for()` generates URLs based on the provided endpoint and any additional parameters.