7.08 FLASK

What is Flask?

- Flask is Python’s micro-framework for web app development.
- It was developed by Armin Ronacher, who led an international team of Python enthusiasts called Pocco.
- Flask consists of Werkzeug WSGI toolkit and Jinja2 template engine. Both were also developed by Pocco and was initially released in April 2010

What is Flask?

- Flask is a lightweight and extensible Python web framework

- Flask allows web developers to be flexible and to comfortably accommodate the frequently released changes in the software development community.

What Is Flask Used For

- Flask framework is used for developing Web Applications in Python programming language.
- It integrates with other third-party services and APIs to bring richness and meaning to the application under development.
- Flask’s core concepts are simple, and it has a tiny footprint.

What is WSGI?

- Web Server Gateway Interface (WSGI) is the standard for Python web application development.
- WSGI is a specification for a universal interface between the web server and the web applications.
- Werkzeug is one of the most advanced WSGI modules that contain various tools and utilities that facilitate web application development. Flask implements Werkzeug.

What is Jinja 2?

- Jinja 2 is a template rendering engine.
- It renders the web pages for the server with any specified custom content given to it by the webserver.
- Flask renders its HTML based templates using Jinja 2.

Why Flask?

- Major advantages of Flask are:
  - Ease of setup and use.
  - Freedom to build the structure of the web application.

- With freedom comes responsibility, similarly, Flask needs the developers to carefully structure it, since Flask doesn’t have “flask rules” to follow as compared to other frameworks.
- As the web app increases in complexity, this structuring is what is going to be the foundation.

Flask – Sample Workflow

1. The first line of code `from flask import Flask` is basically importing Flask package in the file.
2. We next create an object of the flask class using `app = flask()`
3. We send the following argument to the default constructor `__name__`, this will help Flask look for templates and static files.
4. Next, we use the route decorator to help define which routes should be navigated to the following function. `@app.route(/)`
5. In this, when we use `/` it lets Flask direct all the default traffic to the following function.
6. We define the function and the action to be performed.
7. The default content type is HTML, so our string will be wrapped in HTML and displayed in the browser.
8. In the main function created, `app.run()` will initiate the server to run on the local development server.