| ASSIGNMENT NO. 7 | |
|---|---|
| | |
| Assignment No. 7 | To develop **Microservices** framework based distributed application. |
| Objective(s): | By the end of this assignment, the student will be able to create, deploy and test Web-service application. |
| Tools | Python 3, Flask |

# Microservices

- The way developer worked to build application is changing. In the past, software was built as large monolithic application where a team developers would take months to construct a large application built on a common code base.
- Define an architecture that structures the application as a set of loosely coupled, collaborating services. Each service implements a set of narrowly, related functions.
- For example, an application might consist of services such as the order management service, the customer management service etc.
- Each service has its own database in order to be decoupled from other services. Data consistency between services is maintained .

Microservices - also known as the microservice architecture - is an architectural style that structures an application as a collection of services that are:

- Highly maintainable and testable
- Loosely coupled
- Independently deployable
- Organized around business capabilities.

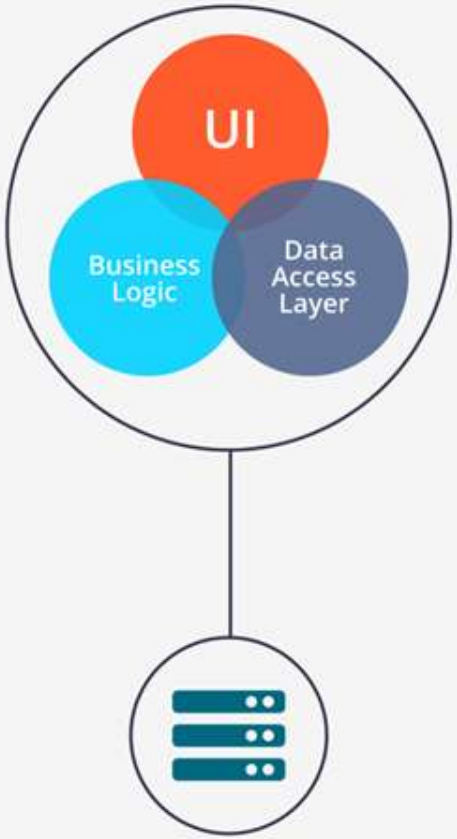
# Monolithic Design vs Microservice

- Traditional application design is often called “monolithic” because the whole thing is developed in one piece.
- Even if the logic of the application is modular it’s deployed as one group, like a Java application as a JAR file for example.
- This monolith eventually becomes so difficult to manage as the larger applications require longer and longer deployment timeframes.

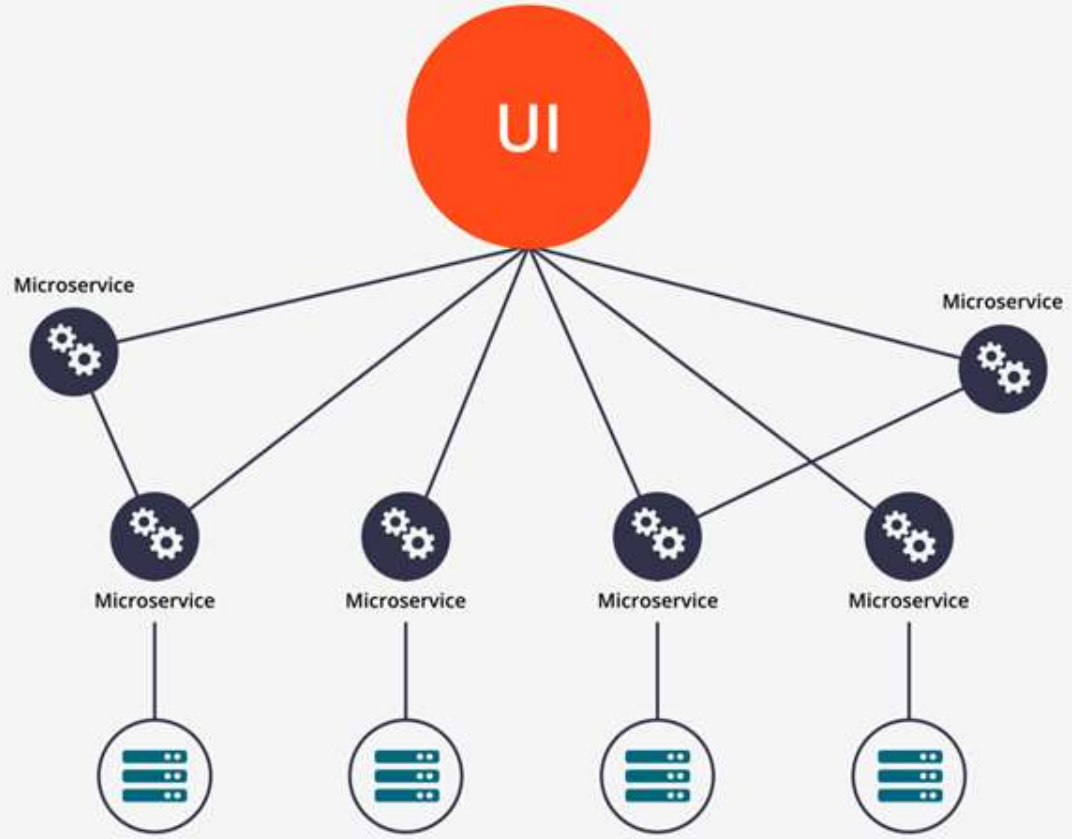
- In contrast, a team designing a microservices architecture for their application will split all of the major functions of an application into independent services.

- Each independent service is usually packaged as an API so it can interact with the rest of the application elements.

# Monolithic Design vs Microservice




Monolithic Architecture




Microservice Architecture

# Web frameworks

- Web frameworks encapsulate what developers have learned over the past twenty years while programming sites and applications for the web.
- Frameworks make it easier to reuse code for common HTTP operations and to structure projects so other developers with knowledge of the framework can quickly build and maintain the application.
- Common web framework functionality: Frameworks provide functionality in their code or through extensions to perform common operations required to run web applications.
    1. URL routing
    2. Input form handling and validation
    3. HTML, XML, JSON, and other output formats with a templating engine
    4. Database connection configuration and persistent data manipulation through an object-relational mapper (ORM) Web security against Cross-site request forgery (CSRF), SQL Injection, Cross-site Scripting (XSS) and other common malicious attacks.
    5. Session storage and retrieval.

# Web frameworks

- **Flask**

  Flask (source code) is a Python web framework built with a small core and easy-to-extend philosophy. Flask is based on the Werkzeug WSGI toolkit and Jinja2 template engine.

- **WSGI**

Web Server Gateway Interface (WSGI) has been adopted as a standard for Python web application development. WSGI is a specification for a universal interface between the web server and the web applications.

- **Werkzeug**

It is a WSGI toolkit, which implements requests, response objects, and other utility functions. This enables building a web framework on top of it. The Flask framework uses Werkzeug as one of its bases.

# Virtual Environment

- In Python, by default, every project on the system will use the same directories to store and retrieve site packages (third party libraries) and system packages (packages that are part of the standard Python library).
- Consider the a scenario where there are two projects: ProjectA and ProjectB, both have a **dependency on the same library**, ProjectC. The problem becomes apparent when we start requiring **different versions** of ProjectC.

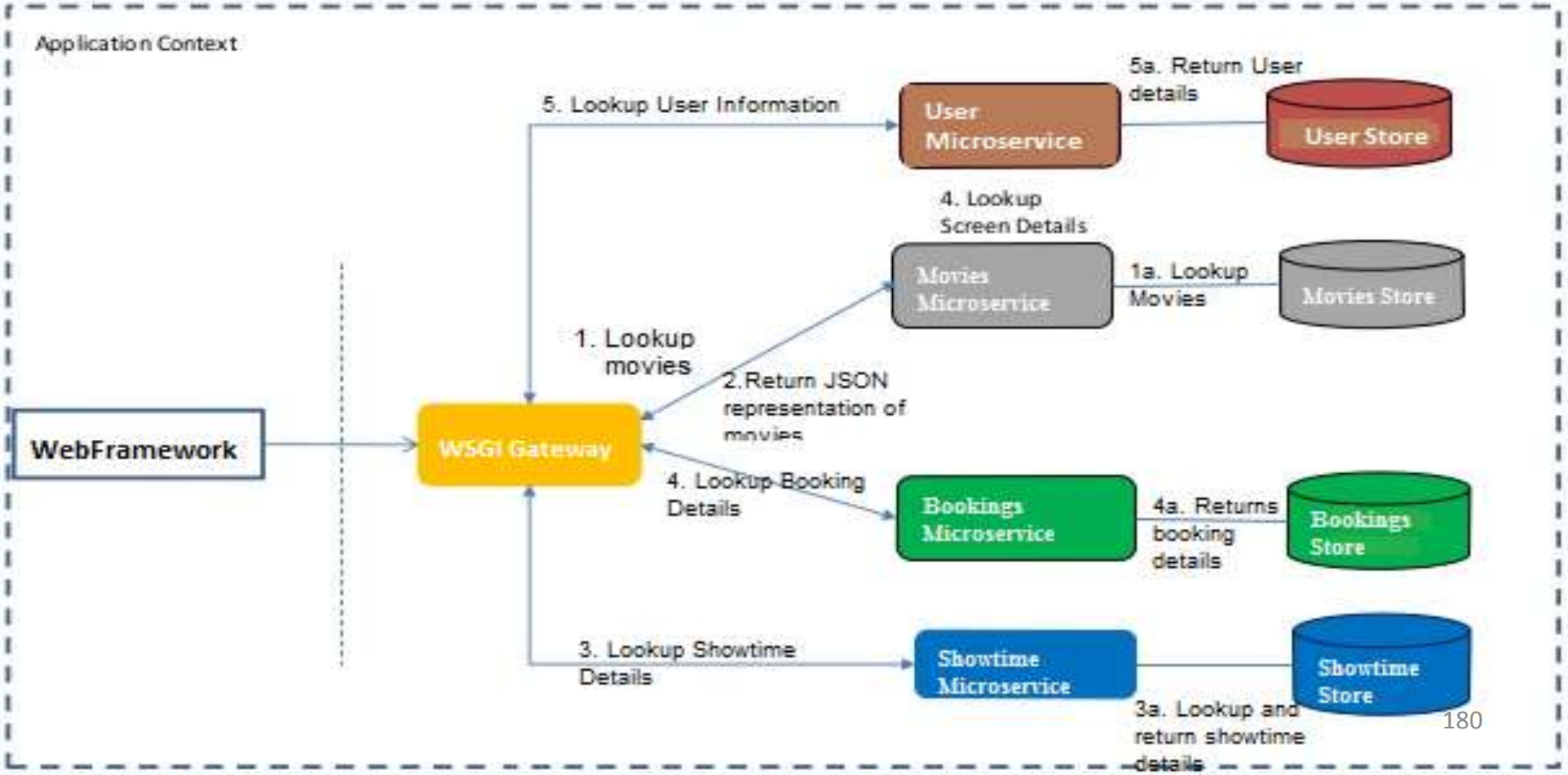
  Maybe ProjectA needs v1.0.0, while ProjectB requires the newer v2.0.0, for example.

  Since projects are stored in site-packages directory according to their name and can't differentiate between versions, both projects, ProjectA and ProjectB, would be required to use the same version which is unacceptable in many cases and hence the virtual environment.

**The main purpose of Python virtual environments is to create an isolated environment for Python projects**. This means that each project can have its own dependencies, regardless of what **dependencies** every other project has.

# Virtual Environment

Here, we are attempting to develop a microservice based architecture for Movie ticket Booking web application. The services are being implemented using python and JSON is used as for Data Store.



# Implementing the Solution

- **Using Virtual Environments:** Install `virtualenv` for development environment. `virtualenv` is a virtual Python environment builder. It helps a user to create multiple Python environments side-by-side. Thereby, it can avoid compatibility issues between the different versions of the libraries.

  The following command installs `virtualenv`:

  ```
  sudo apt-get install virtualenv
  ```

- **Flask Module**: Importing flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of current module (__name__) as argument. The route() function of the Flask class is a decorator, which tells the application which URL should call the associated function.

# Implementing the Solution

- **Route Decoder:**

  The `route()` decorator in Flask is used to bind URL to a function.

  For example –

```
@app.route('/hello')
def hello_world():
return 'hello world'
```

  Here, URL '/hello' rule is bound to the hello_world() function. As a result, if a user visits http://localhost:5000/hello URL, the output of the hello_world() function will be rendered in the browser.

- **Writing the subroutine for the four microservices:** There are four microservices viz., user, Showtimes, Bookings and Movies for which microservices are to be implemented.

# Expected Output

- To install the necessary files and create a virtual environment run:

```
sudo ./setup.sh
```

- To start the 4 microservices run :

```
./startup.sh
```

- To start the command line UI:

```
python cmdline.py
```

# Expected Output

## Running startup.sh

```
dos@dospc ~/Desktop/Junaid/microservices-20181226T110346Z-001/microservices ./startup.sh
dos@dospc ~/Desktop/Junaid/microservices-20181226T110346Z-001/microservices  * Running on http://127.0.0.1:
5003/ (Press CTRL+C to quit)
 * Running on http://127.0.0.1:5001/ (Press CTRL+C to quit)
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Restarting with stat
 * Restarting with stat
 * Running on http://127.0.0.1:5002/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger is active!
 * Debugger is active!
 * Debugger PIN: 229-444-055
 * Debugger PIN: 229-444-055
 * Debugger PIN: 229-444-055
 * Debugger is active!
 * Debugger PIN: 229-444-055
127.0.0.1 - - [26/Dec/2018 16:44:36] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Dec/2018 16:44:36] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Dec/2018 16:44:36] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Dec/2018 16:44:36] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [26/Dec/2018 16:44:41] "GET /movies HTTP/1.1" 200 -
127.0.0.1 - - [26/Dec/2018 16:44:44] "GET /showtimes HTTP/1.1" 200 -
127.0.0.1 - - [26/Dec/2018 16:44:44] "GET /movies HTTP/1.1" 200 -
127.0.0.1 - - [26/Dec/2018 16:44:53] "GET /bookings/Shreyas HTTP/1.1" 404 -
```

# Expected Output

## Running cmdline.py

```
dos@dospc ~/Desktop/Junaid/microservices-20181226T103462-001/microservices python cmdline.py
Welcome to cinema app
1.Get Movie list
2.Get Show Times
3.Get Bookings Info
4.Get User list
5.Book a show
6.Clearscreen
7.Exit
Select an option
1
ID:  276c79ec-a26a-40a6-b3d3-fb242a5947b6
Title:  Avengers Infinity War
Director:  Anthony Russo
Rating:  9.8

ID:  a8034f44-aee4-44cf-b32c-74cf452aaaae
Title:  Stree
Director:  Amar Kaushik
Rating:  9.2

ID:  7daf7208-be4d-4944-a3ae-c1c2f516f3e6
Title:  Mission Impossible 6
Director:  Christopher McQuarrie
Rating:  9.5
```

# Expected Output

```
1.Get Movie list
2.Get Show Times
3.Get Bookings Info
4.Get User list
5.Book a show
6.Clearscreen
7.Exit
Select an option
2
On date: 20180801
ID:  267eedb8-0f5d-42d5-8f43-72426b9fb3e6  MOVIE:  Karwaan
ID:  7daf7208-be4d-4944-a3ae-c1c2f516f3e6  MOVIE:  Mission Impossible 6
ID:  39ab85e5-5e8e-4dc5-afea-65dc368bd7ab  MOVIE:  The Incredibles 2
ID:  a8034f44-aee4-44cf-b32c-74cf452aaaae  MOVIE:  Stree
On date: 20180803
ID:  720d006c-3a57-4b6a-b18f-9b713b073f3c  MOVIE:  Happy Phirr Bhag Jayegi
ID:  39ab85e5-5e8e-4dc5-afea-65dc368bd7ab  MOVIE:  The Incredibles 2
On date: 20180802
ID:  a8034f44-aee4-44cf-b32c-74cf452aaaae  MOVIE:  Stree
ID:  96798c08-d19b-4986-a05d-7da856efb697  MOVIE:  Gold
ID:  39ab85e5-5e8e-4dc5-afea-65dc368bd7ab  MOVIE:  The Incredibles 2
ID:  276c79ec-a26a-40a6-b3d3-fb242a5947b6  MOVIE:  Avengers Infinity War
On date: 20180805
ID:  96798c08-d19b-4986-a05d-7da856efb697  MOVIE:  Gold
ID:  a8034f44-aee4-44cf-b32c-74cf452aaaae  MOVIE:  Stree
ID:  7daf7208-be4d-4944-a3ae-c1c2f516f3e6  MOVIE:  Mission Impossible 6
```

# Expected Output

```
1.Get Movie list
2.Get Show Times
3.Get Bookings Info
4.Get User list
5.Book a show
6.Clearscreen
7.Exit
Select an option
4
Anuja Kharatmol
Souparnika Patil
Vasundhara Kurtakoti
Yojane Mane
Nachiket Ghorpade
Nayana Patil
Kamraj Ambalkar
```

# Expected Output

```
1.Get Movie list
2.Get Show Times
3.Get Bookings Info
4.Get User list
5.Book a show
6.Clearscreen
7.Exit
Select an option
5
>Please enter username for the booking : souparnika_patil
>Please enter the date for the booking : 20180805



ID:  96798c08-d19b-4986-a05d-7da856efb697
Title:  Gold
Director:  Reema Kagdi
Rating:  7.4

ID:  a8034f44-aee4-44cf-b32c-74cf452aaaae
Title:  Stree
Director:  Amar Kaushik
Rating:  9.2

ID:  7daf7208-be4d-4944-a3ae-c1c2f516f3e6
```

# Expected Output

```
ID:  39ab85e5-5e8e-4dc5-afea-65dc368bd7ab
Title:  The Incredibles 2
Director:  Brad Bird
Rating:  7.1

ID:  276c79ec-a26a-40a6-b3d3-fb242a5947b6
Title:  Avengers Infinity War
Director:  Anthony Russo
Rating:  9.8

>Enter the id for the booking : 276c79ec-a26a-40a6-b3d3-fb242a5947b6



Booking the show for the following movie on date 20180802
ID:  276c79ec-a26a-40a6-b3d3-fb242a5947b6
Title:  Avengers Infinity War
Director:  Anthony Russo
Rating:  9.8

Press enter to continue

BOOKING DONE!! Thank you for using Cinema app
```

# THANK YOU