

ASSIGNMENT NO. 7

AIM/OBJECTIVE:

To develop microservices framework based distributed application.

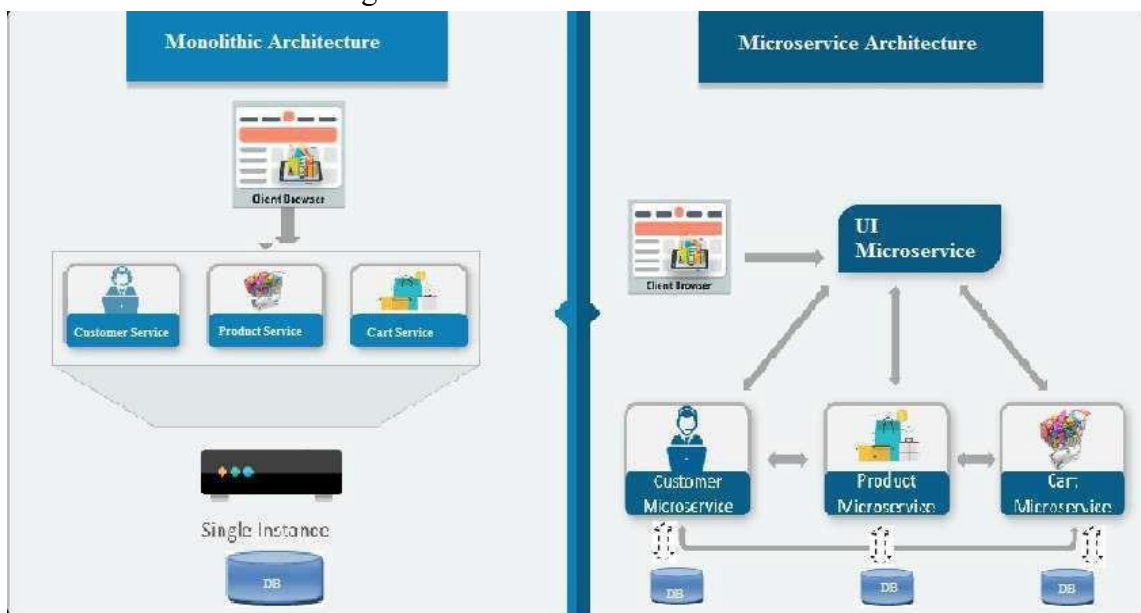
Tools / Environment:

Python 3.6.0 using Flask framework.

Related Theory:

1. Microservices:

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 with the monolith type application, here’s what an app developed with a microservices focus might look like:



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.

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.

The microservice architecture enables the continuous delivery/deployment of large, complex applications. It also enables an organization to evolve its technology stack.

2. 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. These common operations include:

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)
5. Web security against Cross-site request forgery (CSRF), SQL Injection, Cross-site Scripting (XSS) and other common malicious attacks
6. Session storage and retrieval.

3. 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.

4. 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.

5. 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.

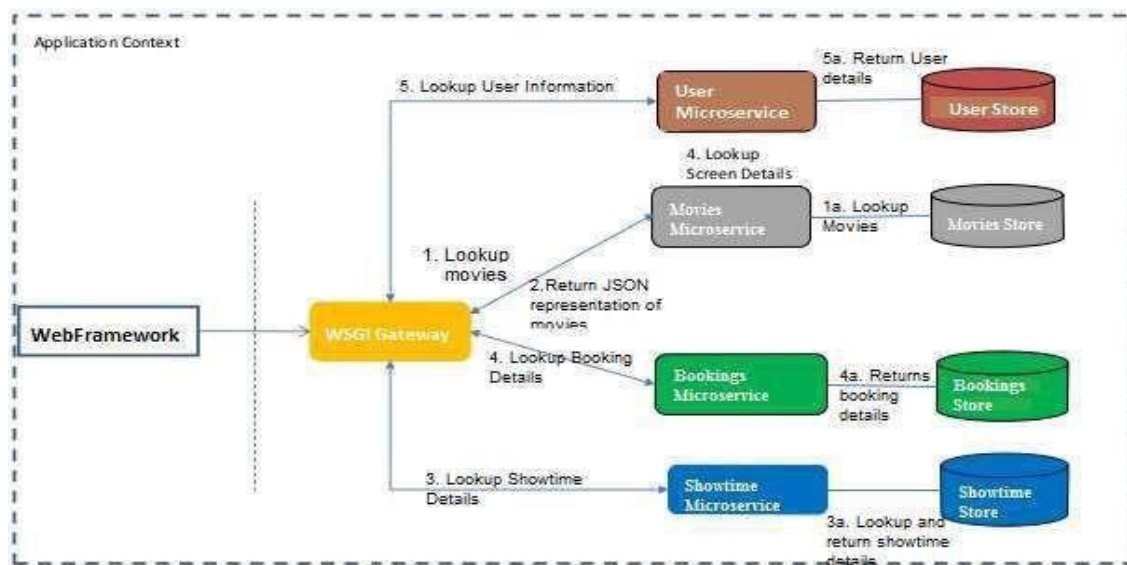
6. Virtual Environment:

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 of which 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 just 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. There are no limits to the number of environments you can have since they're just directories containing a few scripts. Plus, they're easily created using the `virtualenv` or `pyenv` command line tools.

Designing the solution:



Here, we are attempting to develop an 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:

- 1. 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
```

- 2. 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.

Route decorator:

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

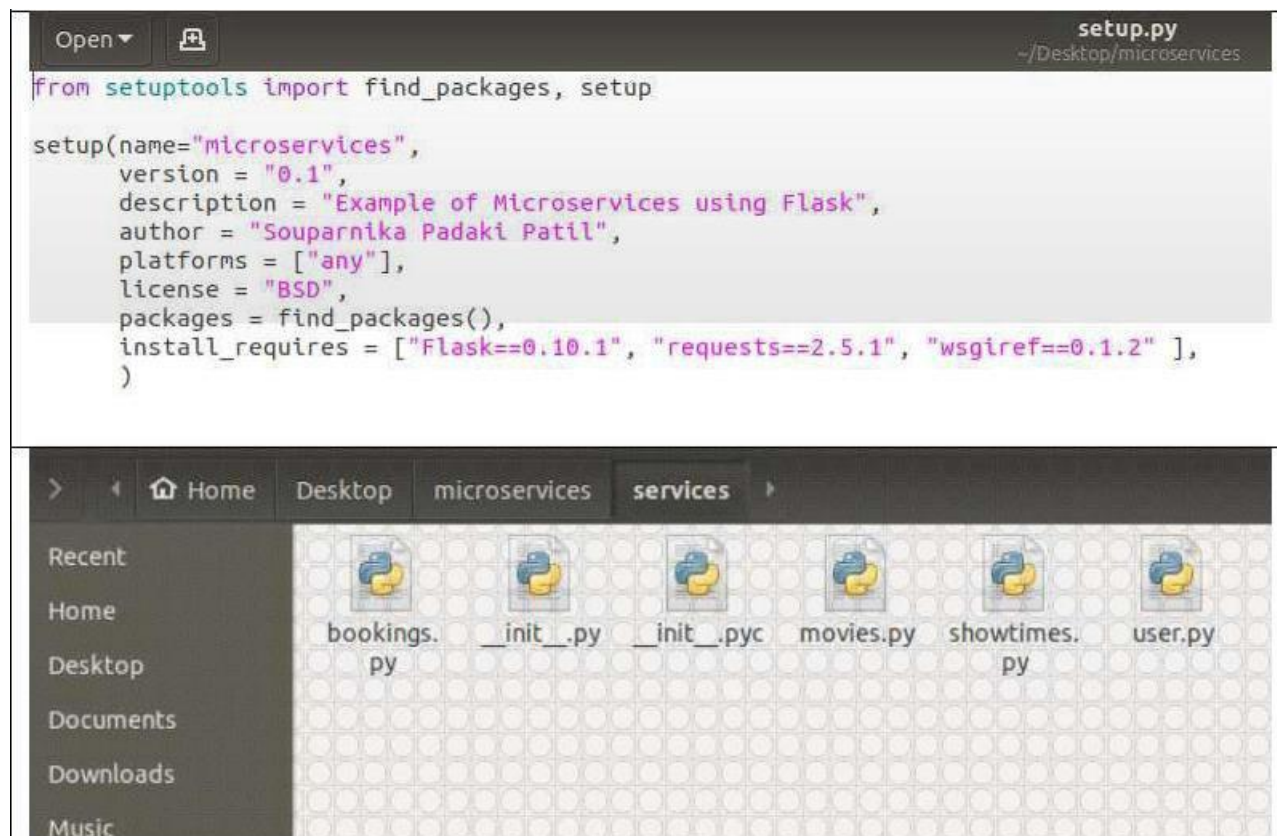
For example –

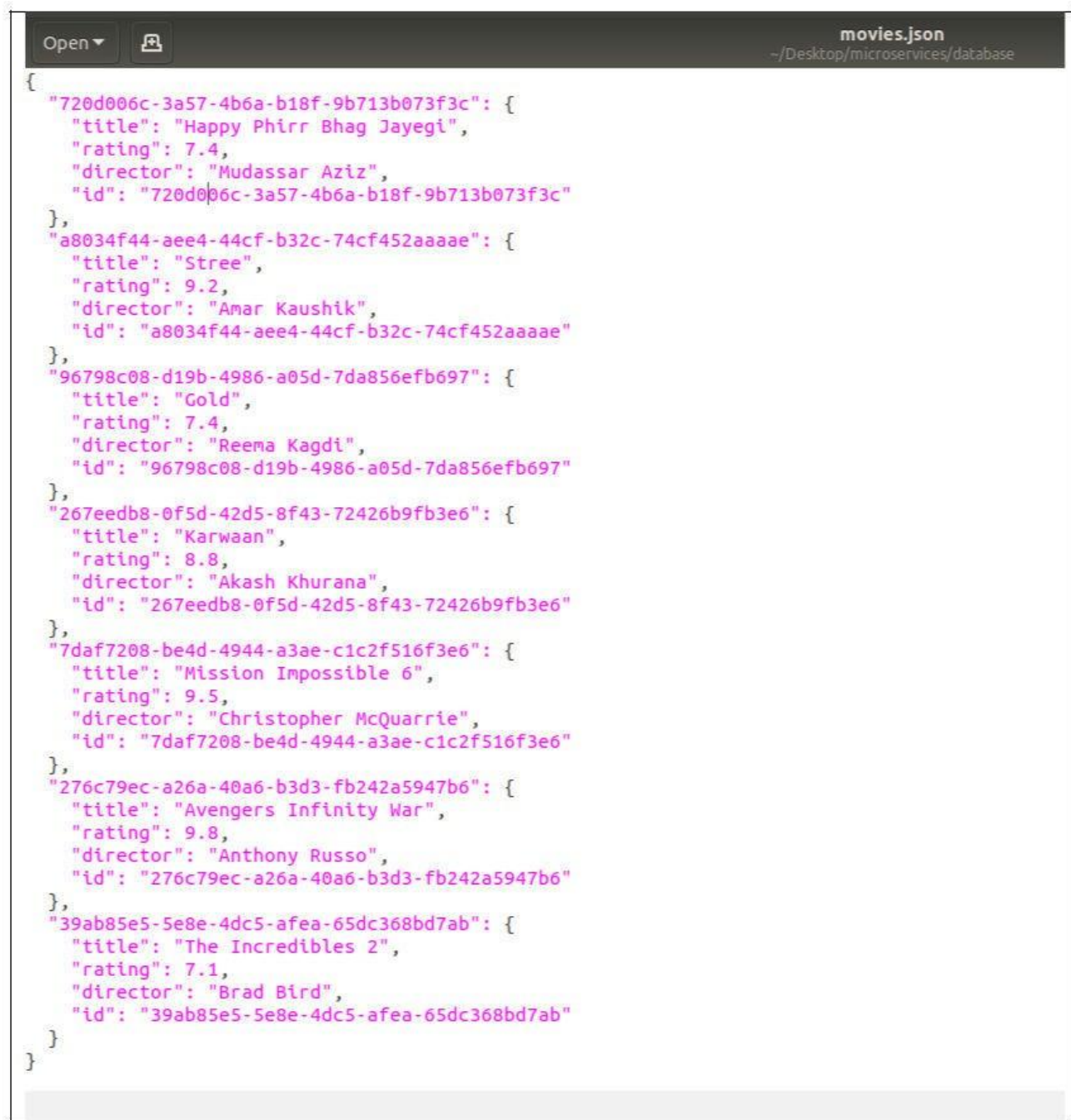
```
@app.route('/hello ')\n\ndef hello_world():\n\return 'hello world'
```

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.

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

Writing the source code:





The screenshot shows a code editor window with a dark theme. The title bar at the top right says "movies.json" and the path below it is "~/Desktop/microservices/database". The editor contains a JSON array of movie objects. Each object has fields for "title", "rating", "director", and "id". The movies listed are "Happy Phirr Bhag Jayegi", "Stree", "Gold", "Karwaan", "Mission Impossible 6", "Avengers Infinity War", and "The Incredibles 2".

```
{
  "720d006c-3a57-4b6a-b18f-9b713b073f3c": {
    "title": "Happy Phirr Bhag Jayegi",
    "rating": 7.4,
    "director": "Mudassar Aziz",
    "id": "720d006c-3a57-4b6a-b18f-9b713b073f3c"
  },
  "a8034f44-ae4-44cf-b32c-74cf452aaaae": {
    "title": "Stree",
    "rating": 9.2,
    "director": "Amar Kaushik",
    "id": "a8034f44-ae4-44cf-b32c-74cf452aaaae"
  },
  "96798c08-d19b-4986-a05d-7da856efb697": {
    "title": "Gold",
    "rating": 7.4,
    "director": "Reema Kagdi",
    "id": "96798c08-d19b-4986-a05d-7da856efb697"
  },
  "267eedb8-0f5d-42d5-8f43-72426b9fb3e6": {
    "title": "Karwaan",
    "rating": 8.8,
    "director": "Akash Khurana",
    "id": "267eedb8-0f5d-42d5-8f43-72426b9fb3e6"
  },
  "7daf7208-be4d-4944-a3ae-c1c2f516f3e6": {
    "title": "Mission Impossible 6",
    "rating": 9.5,
    "director": "Christopher McQuarrie",
    "id": "7daf7208-be4d-4944-a3ae-c1c2f516f3e6"
  },
  "276c79ec-a26a-40a6-b3d3-fb242a5947b6": {
    "title": "Avengers Infinity War",
    "rating": 9.8,
    "director": "Anthony Russo",
    "id": "276c79ec-a26a-40a6-b3d3-fb242a5947b6"
  },
  "39ab85e5-5e8e-4dc5-afea-65dc368bd7ab": {
    "title": "The Incredibles 2",
    "rating": 7.1,
    "director": "Brad Bird",
    "id": "39ab85e5-5e8e-4dc5-afea-65dc368bd7ab"
  }
}
```

Building and Executing the solution:

1. To install the necessary files and create a virtual environment

run: `sudo ./setup.sh`

2. To start the 4 microservices run :

`./startup.sh`

3. To start the command line UI:
python cmdline.py

Running startup.sh

```
dos@dospc ./startup.sh
dos@dospc * 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 -
```

Running cmdline.py

```
dos@dospc 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
```

```
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-ae4-44cf-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-ae4-44cf-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-ae4-44cf-b32c-74cf452aaaae MOVIE: Stree
ID: 7daf7208-be4d-4944-a3ae-c1c2f516f3e6 MOVIE: Mission Impossible 6
```

```
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
```

```
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-ae4-44cf-b32c-74cf452aaaae
Title: Stree
Director: Amar Kaushik
Rating: 9.2
```

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

```
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
```

NAME: SOUMYA MISHRA
DATE:1-05-2021

SEAT NO-B150088639
PRN-71843071K

Conclusion:

With microservices, modules within software can be independently deployable. In a microservices architecture, each service runs a unique process and usually manages its own database. This not only provides development teams with a more decentralized approach to building software, it also allows each service to be deployed, rebuilt, redeployed and managed independently. Netflix, eBay, Amazon, the UK Government Digital Service, Twitter, PayPal, The Guardian, and many other large-scale websites and applications have all evolved from monolithic to microservices architecture.