In [ ]:
```
Name - Shivraj Pandurang Mane.
Class - BE Artificial Intelligence and Data Science.
Roll No. - 37
Practical No. 04 - Write code to simulate requests coming from clients and distribute th
using the load balancing algorithms.
```

In [2]:
```
import random
import time

class Server:
    def __init__(self, id):
        self.id = id
        self.current_connections = 0

    def handle_request(self):
        """Simulate handling a request by a server."""
        self.current_connections += 1
        print(f"Server {self.id} is handling the request. Total connections: {self.curre

    def release_request(self):
        """Release a request from the server."""
        if self.current_connections > 0:
            self.current_connections -= 1
            print(f"Server {self.id} has released a request. Total connections: {self.cu

class LoadBalancer:
    def __init__(self, servers):
        self.servers = servers
        self.server_count = len(servers)
        self.round_robin_index = 0

    def round_robin(self):
        """Round Robin load balancing algorithm."""
        server = self.servers[self.round_robin_index]
        self.round_robin_index = (self.round_robin_index + 1) % self.server_count
        return server

    def least_connections(self):
        """Least Connections load balancing algorithm."""
        server = min(self.servers, key=lambda s: s.current_connections)
        return server

    def random_selection(self):
        """Random load balancing algorithm."""
        return random.choice(self.servers)

    def distribute_request(self, algorithm="round_robin"):
        """Distribute requests among servers based on the selected algorithm."""
        if algorithm == "round_robin":
            server = self.round_robin()
        elif algorithm == "least_connections":
            server = self.least_connections()
        elif algorithm == "random":
            server = self.random_selection()
        else:
            raise ValueError("Unknown algorithm")
```

```python
        server.handle_request()

# Simulate the scenario
def simulate_requests():
    servers = [Server(i) for i in range(1, 4)]  # Three servers
    load_balancer = LoadBalancer(servers)

    # Simulate 10 client requests
    for _ in range(10):
        algorithm = random.choice(["round_robin", "least_connections", "random"])  # Ran
        print(f"\nDistributing request using {algorithm} algorithm.")
        load_balancer.distribute_request(algorithm)
        time.sleep(0.5)  # Simulate time delay between requests

    # Simulate releasing requests
    for server in servers:
        server.release_request()

# Run the simulation
simulate_requests()
```

```
Distributing request using least_connections algorithm.
Server 1 is handling the request. Total connections: 1

Distributing request using random algorithm.
Server 1 is handling the request. Total connections: 2

Distributing request using least_connections algorithm.
Server 2 is handling the request. Total connections: 1

Distributing request using least_connections algorithm.
Server 3 is handling the request. Total connections: 1

Distributing request using random algorithm.
Server 2 is handling the request. Total connections: 2

Distributing request using least_connections algorithm.
Server 3 is handling the request. Total connections: 2

Distributing request using round_robin algorithm.
Server 1 is handling the request. Total connections: 3

Distributing request using random algorithm.
Server 3 is handling the request. Total connections: 3

Distributing request using least_connections algorithm.
Server 2 is handling the request. Total connections: 3

Distributing request using random algorithm.
Server 1 is handling the request. Total connections: 4
Server 1 has released a request. Total connections: 3
Server 2 has released a request. Total connections: 2
Server 3 has released a request. Total connections: 2
```

```
In [ ]:
```