CLOUD COMPUTING-MINI PROJECT REPORT

BREAKDOWN MONOLITHS

APRIL 2023

SUBMITTED BY:

KANISHK A D – PES1UG20CS716

DARSHAN BAFNA – PES1UG20CS727

VENKATESH A – PES1UG21CS840

VIGNESH KANNAN – PES1UG21CS842

CLASS:6TH SEMESTER SECTION:"L"

PES UNIVERSITY

Short Description and Scope of the Project

- In order to deploy a Docker container that runs the Flask application, the first task of the project is to add the necessary modules to the requirements file. After that, write a Dockerfile.

- The following goal is to troubleshoot the following problems with a web-based calculator application: Python treats the inputs as strings, which results in the arithmetic operations failing to function as expected.
- Breaking the monolith, or separating each calculator function into a distinct flask application, is the next task. This will allow each function to continue operating as needed even if the main landing service is unavailable.
- And we splitted the services as per below:
    1. Addition
    2. Division
    3. Equal
    4. Exponent
    5. GCD [Greatest Commom Divisor]
    6. Greater-than
    7. Less-than
    8. LCM
    9. Modulus
    10. Multiplication
    11. Subtraction

Methodology

- Breaking a monolithic application into microservices can be accomplished using various methodologies, depending on the specific needs and circumstances of the project. Some common approaches are:

- Strangler Pattern: This approach involves gradually replacing parts of a monolithic application with microservices over time. The new microservices are gradually integrated into the system until the entire application has been broken down into smaller services.

- Domain-driven Design: This approach involves analyzing the business domain and breaking the monolithic application into smaller, autonomous services that are focused on specific business capabilities.

- Event-driven Architecture: This approach involves breaking the monolithic application into smaller services that communicate with each other through events. This allows for a loosely coupled system where each service can operate independently.

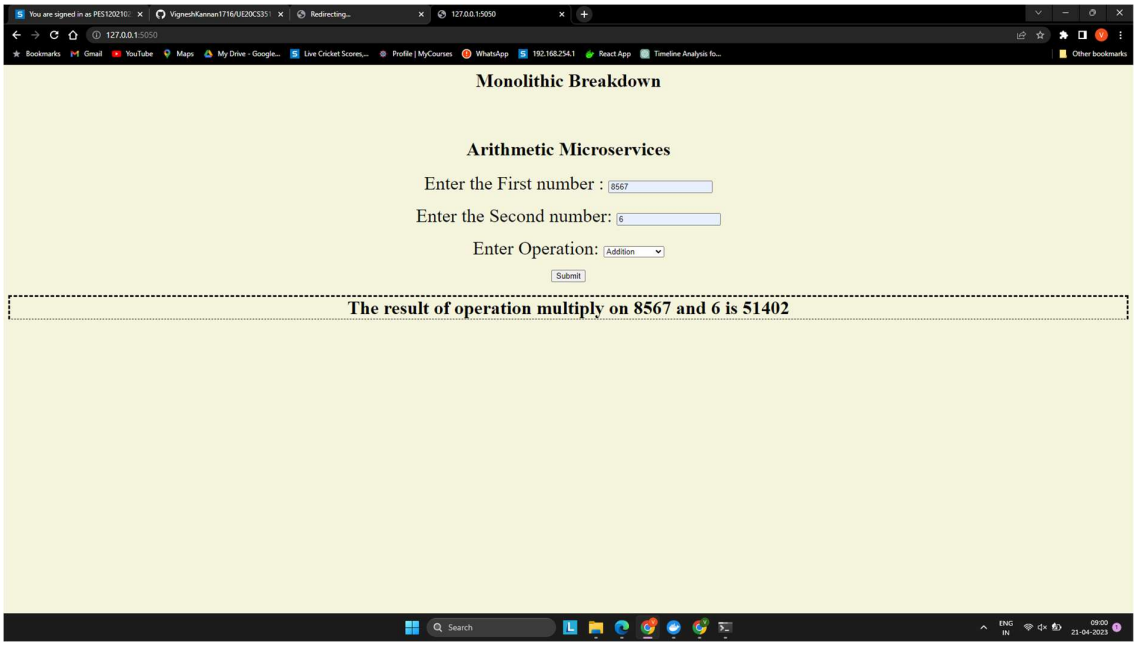
- API Gateway: This approach involves using an API gateway to break the monolithic application into smaller services. The API gateway acts as a proxy for incoming requests and routes them to the appropriate service.

- Containerization: This approach involves packaging each service into a container and deploying them on a container orchestration platform like docker. This allows for easy scaling and management of the individual services.

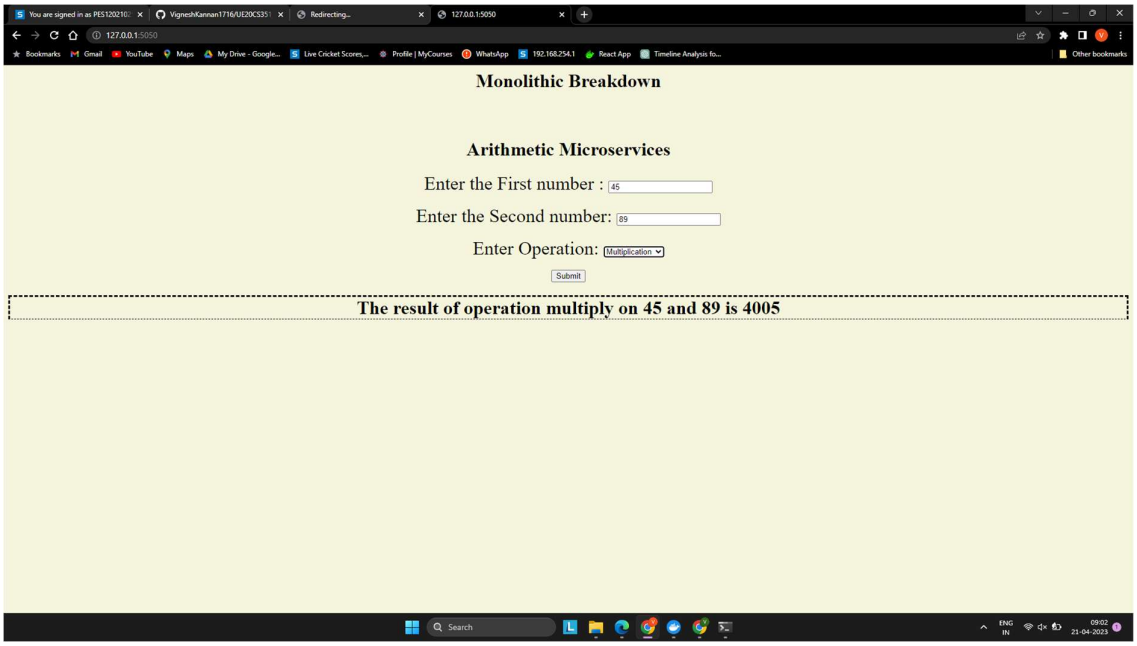
- It's important to note that the choice of methodology depends on factors such as the size of the monolithic application, the level of coupling between its components, and the skills of the development team.

Testing

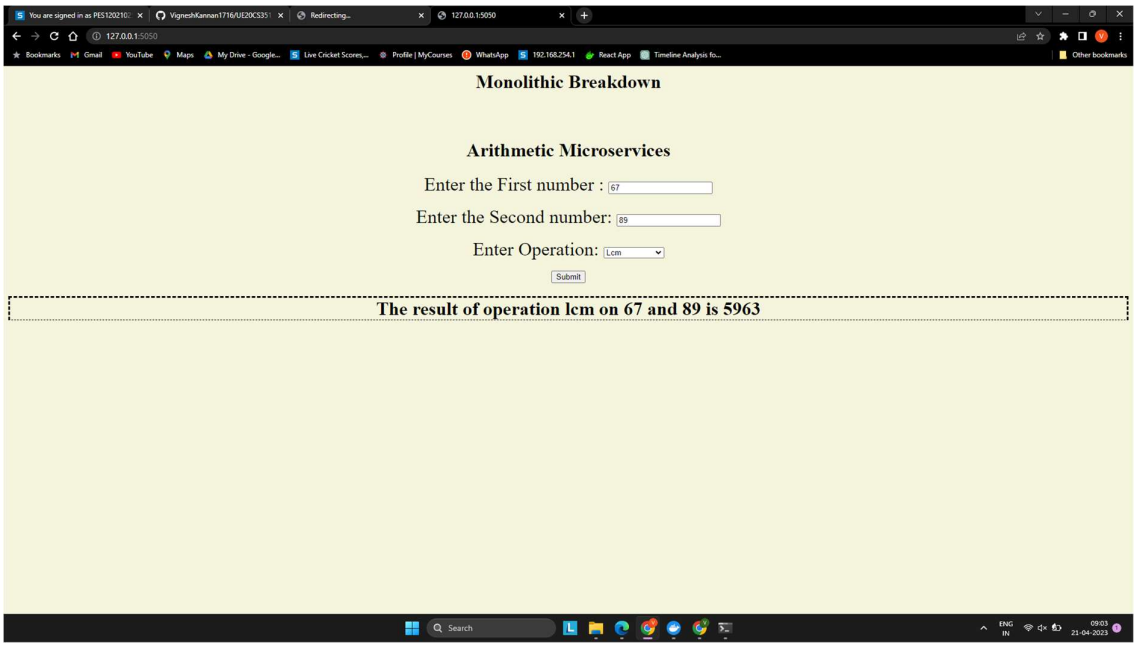
Addition Operator:



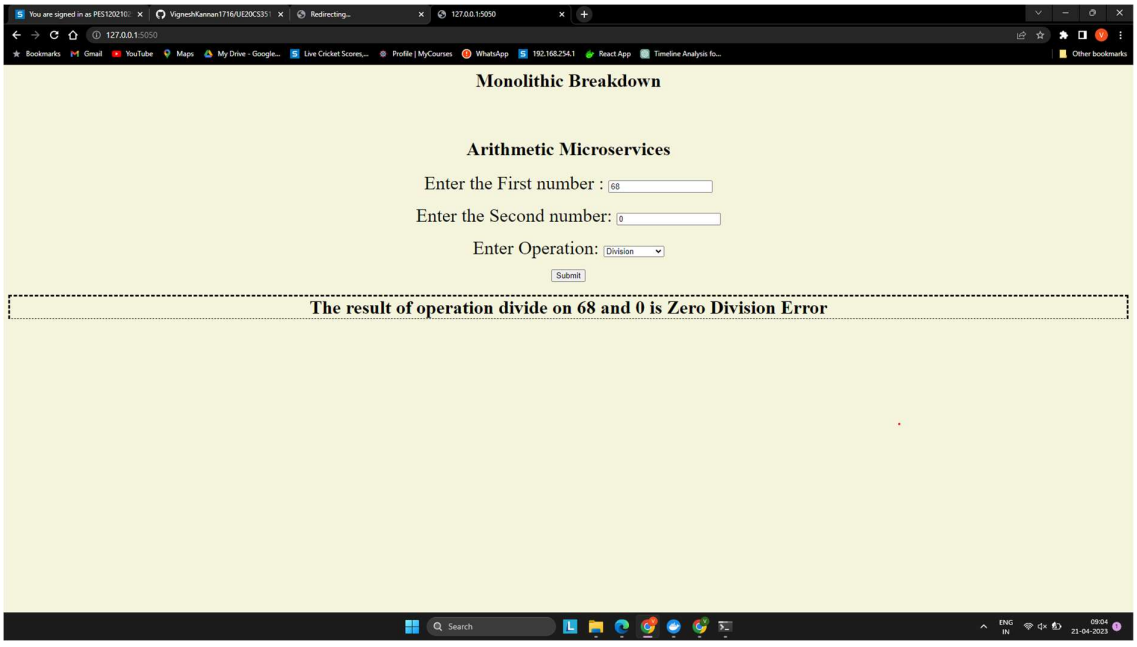
Multiplication Operator:



LCM Operator:



Zero Division Error:



Results and Conclusions:

- To mitigate the risk of monolithic breakdown, organizations can adopt a microservices architecture, which breaks down the monolithic application into smaller, more manageable components.
- The original functions (addition, subtraction, multiplication etc.. have been separated into individual services that will process the data received from the main service and provide the output as a result. In other words, each function has been split into a separate service that will handle its specific task and send back the processed data to the main service.
- Basic error handling has been implemented to guarantee that an error message is shown in case of an error happening. In other words, measures have been taken to handle errors, so that if an error occurs, the appropriate error message will be displayed.
- More services like modulus, gcd, lcm, etc. were included in the application to expand its capabilities. In other words, the application was enhanced by adding extra services to provide additional features such as modulus, gcd, lcm, etc.