Cloud Computing - Mini Project Report
Breaking down monoliths
April 2023

Submitted By:

Name: Sujith B
SRN: PES1UG20CS694
VI Semester Section L
PES University

Name: Supreeth N P
SRN: PES1UG20CS708
VI Semester Section L
PES University

Name: Varun Vishwanatha Avabratha
SRN: PES1UG20CS699
VI Semester Section L
PES University

Name: Srikrshna P
SRN: PES1UG20CS692
VI Semester Section L
PES University

Short Description and Scope of the Project

This project converts a monolithic application to a microservices architecture-based application. Monolith architecture is a software architecture style where all the code for an application is built as a single, indivisible unit, typically stored in a single code repository. In this architectural style, the entire application is designed and developed as a single cohesive unit, with all the components tightly interconnected and interdependent. They are relatively simple to develop and deploy, but they can be difficult to scale and maintain over time, especially as the application grows in size and complexity. Microservices architecture is a software architecture style where a large application is decomposed into smaller, independent services that can be developed, deployed, and scaled separately. Microservices architecture emphasizes the principle of separation of concerns, which means that each microservice should be responsible for a single, specific function or feature. Additionally, microservices architecture can be more resilient and scalable than monolithic architecture, as the individual microservices can be scaled up or down independently to meet changing demands.

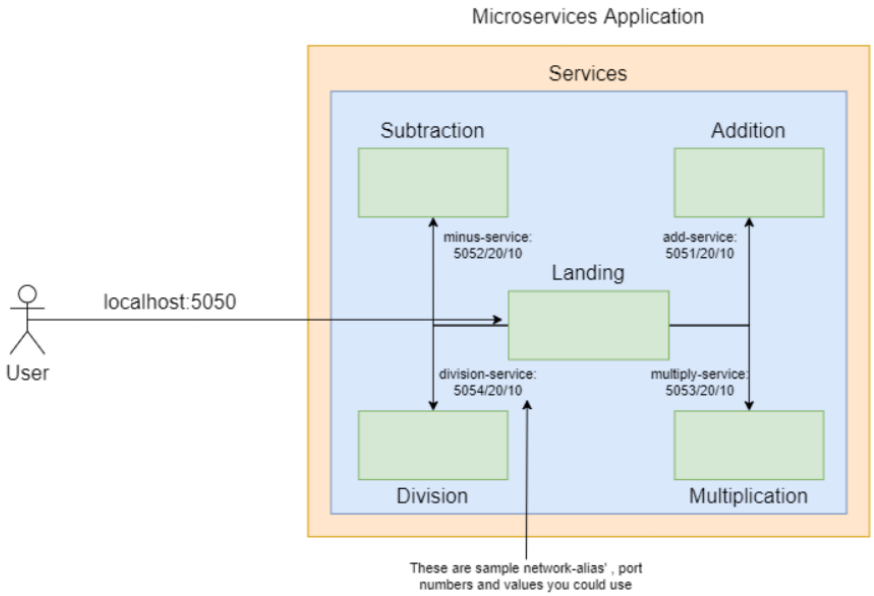
The scope of the project is to break down the monolithic application into microservices by refactoring the monolithic application codebase to extract the individual microservices. This may require making changes to the existing code, such as modularizing the code and decoupling dependencies. It would also involve deploying, monitoring, and managing the microservices using the Docker application.

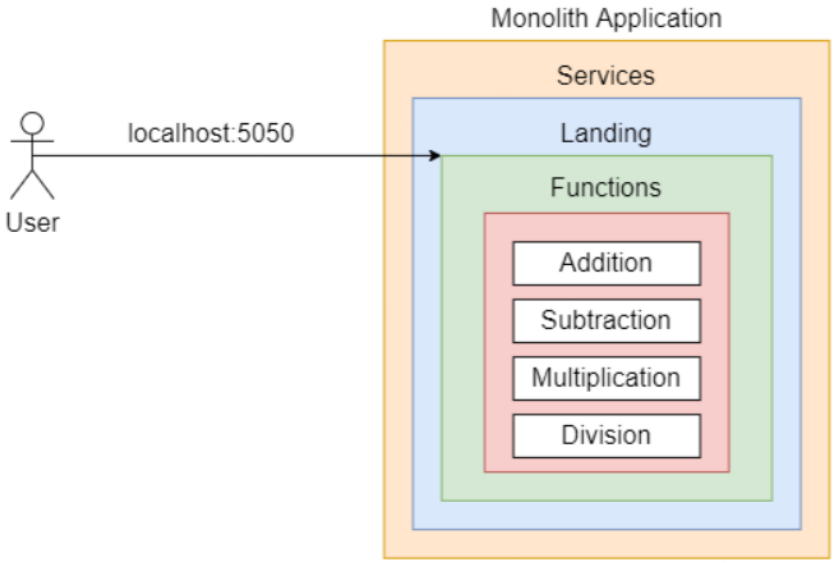
Methodology

In this project, we will be creating 7 microservices that implement basic arithmetic operations by taking two numbers as input and generating a result of the operation. This is achieved by using docker-compose commands to build the application based on the Dockerfiles defined in each of the microservice. The Python-3.8 Alpine image is pulled using docker to build the application. Each application will consist of a class that inherits the Resource class of the flask_restful module. The flask module is used to communicate the queries and results over the network using distinct ports for each microservice.

The following seven microservices are implemented as part of the project:

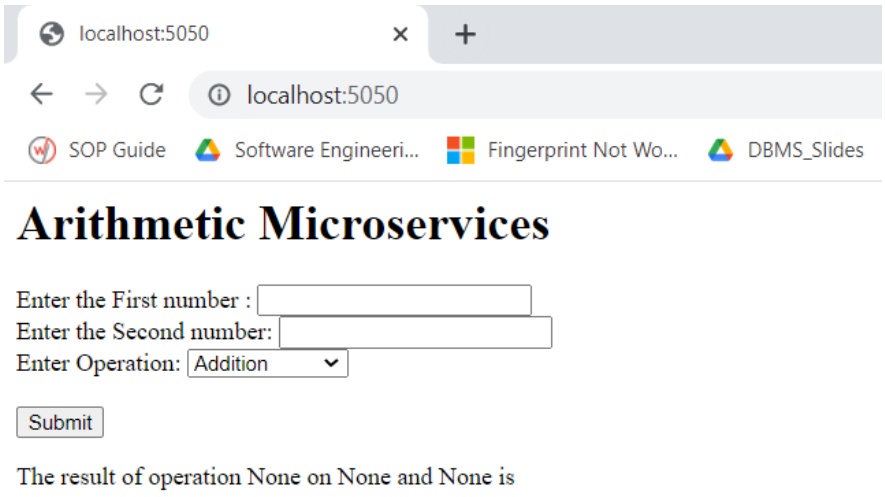
1) Addition - uses Port 5051
2) Subtraction - uses Port 5052
3) Multiplication - uses Port 5053
4) Division - uses Port 5054
5) Modulus - uses Port 5055
6) Lesser than - uses Port 5056
7) Greater than - uses Port 5057



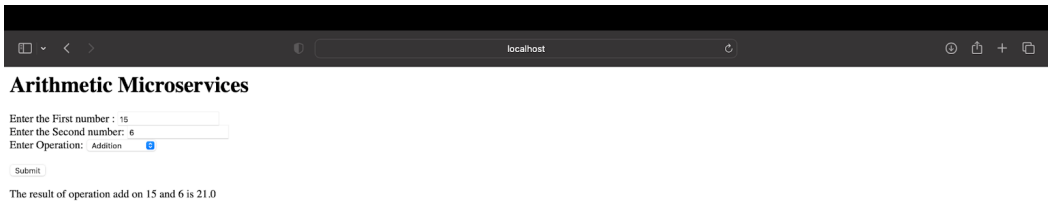

Monolith Application
Services
Landing
Functions
Addition
Subtraction
Multiplication
Division
localhost:5050
User

Testing

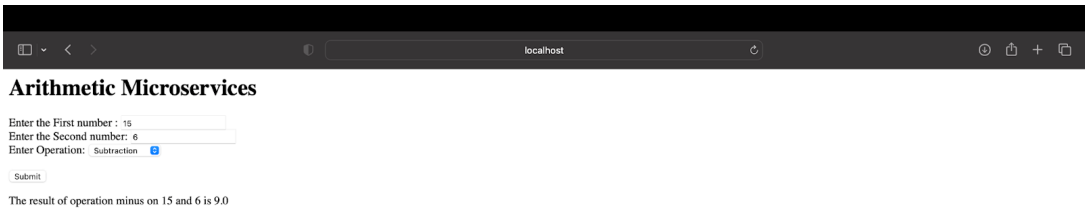
1) Landing Page



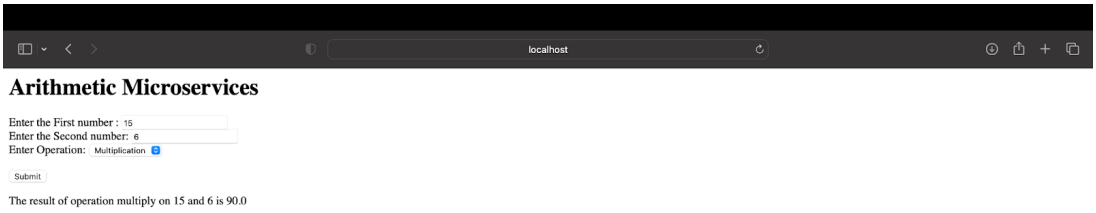
2) Addition



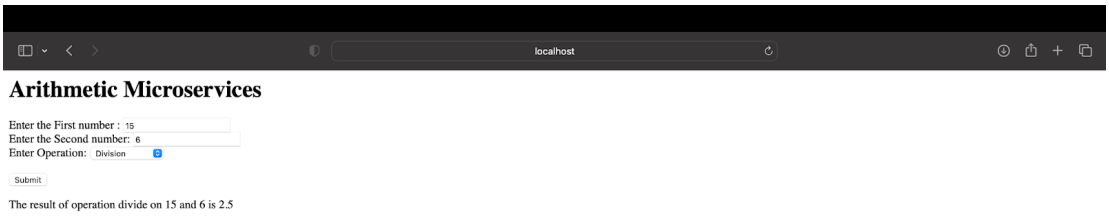
3) Subtraction



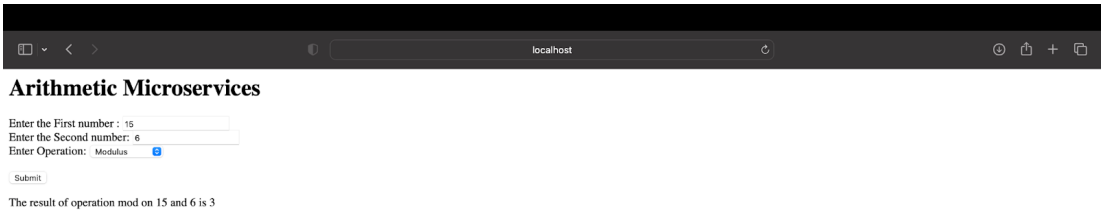
4) Multiplication



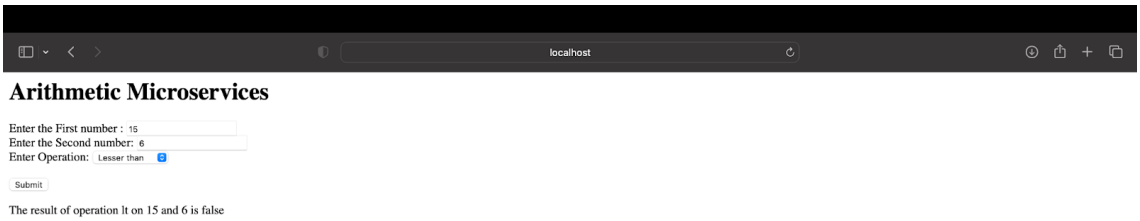
5) Division



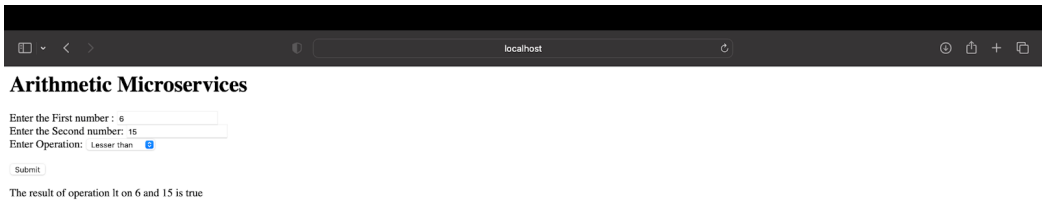
6) Modulus



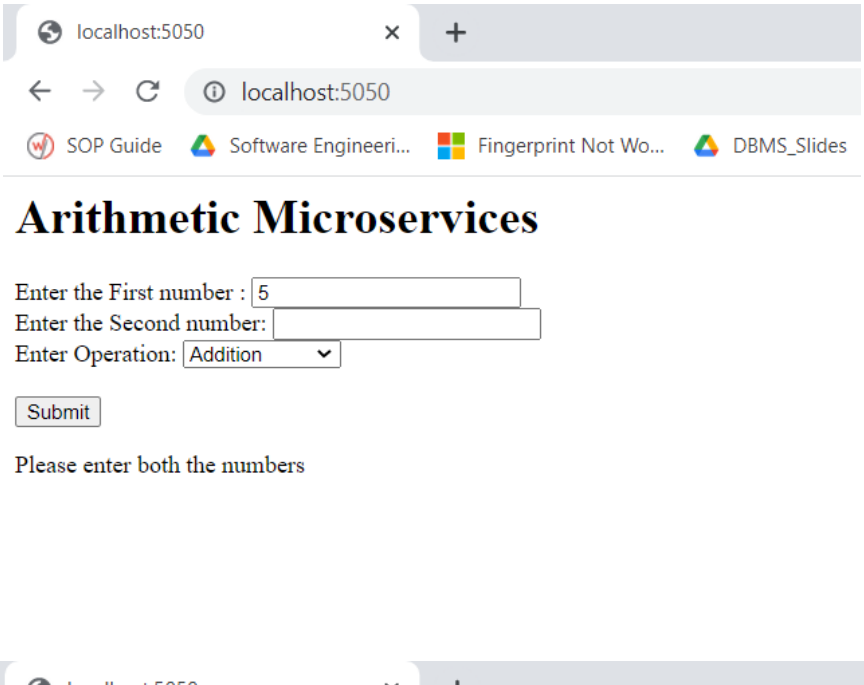
7) Lesser than

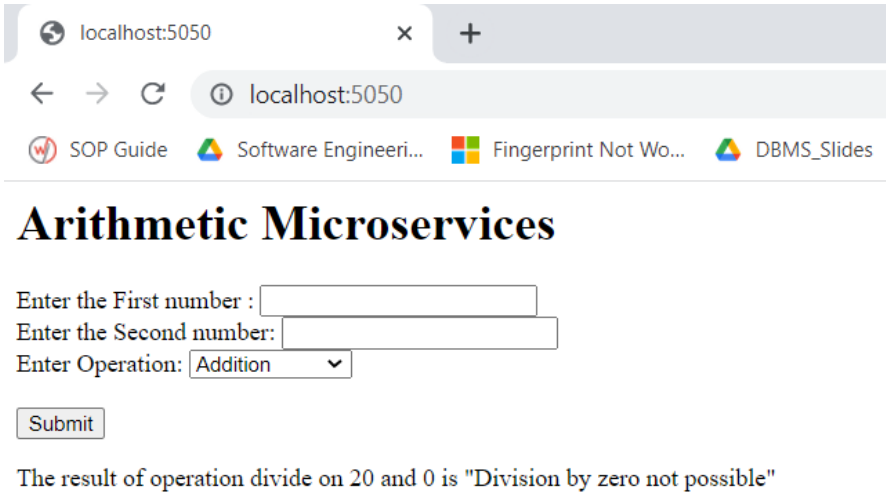


8) Greater than



9) Invalid Inputs





Results and Conclusions

As shown in the above testing process, the application works for all the operations defined and handles all the edge cases without getting crashed by unforeseen exceptions and errors.

The conversion to monolithic architecture provides the following pros and cons:

1) Improved scalability: Microservices architecture allows for the independent scaling of individual services, which can lead to better performance and cost efficiency.
2) Increased flexibility: With microservices, it's easier to make changes to individual components of the system without affecting the entire application. This can help to make updates faster and reduce downtime.
3) Better fault isolation: Since microservices are designed to be independent, if one service fails, it's less likely to affect the entire application. This can improve overall system resilience and reliability.
4) Greater development and deployment overhead: Developing and deploying microservices requires additional tooling and infrastructure, which can be costly and time-consuming to set up.
5) Need for strong DevOps practices: As microservices architectures require more coordination between teams and more frequent deployments, strong DevOps practices become critical to ensure that the system remains stable and reliable.

Overall, the decision to convert a monolithic architecture into microservices requires careful consideration of the benefits and drawbacks, as well as the specific needs and constraints of the system and organization.