Microservices in Software Development

PREPARED BY

Manav Garg

CS20B1009

PREPARED UNDER

Dr. Priodyuti Pradhan

Assistant Professor

1. Project Overview

The project showcases the use and importance of microservices in today's software development. It is a simple e-commerce website developed using a microservices architecture. This architecture breaks down the application into smaller, independent services that are easier to manage, scale, and deploy. Each microservice handles a specific aspect of the e-commerce website, such as product management, user authentication, and order processing. These microservices communicate with each other over the network, often using lightweight protocols like HTTP or messaging queues.

2. Earlier Architectures

Monolithic Architecture:

In monolithic architecture, the entire application is developed as a single unit. All the components of the application, such as the user interface, business logic, and data access layer, are tightly integrated and run as a single process. It is simpler to develop and deploy initially, as everything is contained within a single codebase and deployment unit.

Advantages:

1. It is easier to develop and deploy, especially for small applications.
2. It’s easy to maintain consistency here because everything is contained in a single place.
3. The deployment of the application is very simple because everything is in a single unit.

Disadvantages:

1. It is very difficult to scale a single part of the application because to scale that component the whole application needs to be scaled.
2. We can not use different technologies and languages to create the application.
3. The maintenance is also difficult because of the monolithic architecture.

Microservices Architecture:

In a microservices architecture, the application is broken down into smaller, independent services that run as separate processes and communicate with each other over the network. Each service is responsible for a specific business function and can be developed, deployed, and scaled independently.

Advantages:

1. It is easier to scale specific parts of the application independently based on demand.
2. Each service can be developed using different technologies and different languages, allowing for more flexibility and innovation.
3. A failure in one service does not necessarily affect the entire application, So it improves fault tolerance.
4. It is easier to deploy and update individual services, leading to faster release cycles and the ability to deliver new features more quickly.

Disadvantages:

1. It's difficult to manage and coordinate multiple services. Especially in terms of monitoring, testing, and deployment.
2. Services need to communicate over the network, which can introduce latency and overhead.
3. Maintaining consistency across services, such as data consistency, can be challenging.

3. Use of Microservices in Today’s Market

Microservices architecture has gained significant popularity in today's market due to its ability to address the challenges of scalability, flexibility, and agility faced by modern software applications. According to a survey, 91% of organizations are using or have plans to use microservices in their development processes. Let's explore some key statistics and data that highlight the use and impact of microservices in today's market:

1. Adoption Rate of Microservices:

- Survey Results: According to the 2021 State of Microservices report, 88% of respondents reported using microservices in their organization.
- Growth Trends: The adoption of microservices has been steadily increasing, with a 64% increase in adoption from 2018 to 2021.

2. Business Impact:

- Faster Time to Market: Companies using microservices report a 42% improvement in their time to market.
- Improved Scalability: 66% of organizations using microservices report improved scalability.

3. Technology Stack:

- Programming Languages: Java, JavaScript (Node.js), and Python are among the most commonly used languages for building microservices.
- Frameworks and Tools: Spring Boot, Express.js, and Flask are popular frameworks and tools used for building microservices.

4. Industry Adoption:

- IT and Technology industries lead in the adoption of microservices, followed by Financial Services, Healthcare, and Retail.
- Microservices adoption is higher in larger organizations, with 70% of companies with more than 10,000 employees using microservices.

Conclusion:

Microservices architecture is a dominant trend in today's market, offering significant benefits in terms of scalability, flexibility, and agility. Despite challenges such as complexity and integration with legacy systems, the adoption of microservices continues to grow, driven by the need for faster time to market and improved scalability.

4. E-commerce website using Microservices

Microservices Architecture in the E-commerce Website:

Architecture Overview:

- The e-commerce website is built using a microservices architecture, where different services handle specific functionalities such as product management, cart service, and order processing.
- Each microservice is developed, deployed, and scaled independently, allowing for greater flexibility and scalability.

Key Microservices:

1. Product Service: Manages the products available for sale, including their details, pricing, and inventory.
2. Order Service: Manages the creation, processing, and fulfillment of orders.
3. Shipping Service: Manages shipping and delivery logistics for orders.
4. Recommendation Service: Recommends products based on the type of product you are already searching for.
5. Currency Service: Allows you to check the price of products in different currency systems.

Communication:

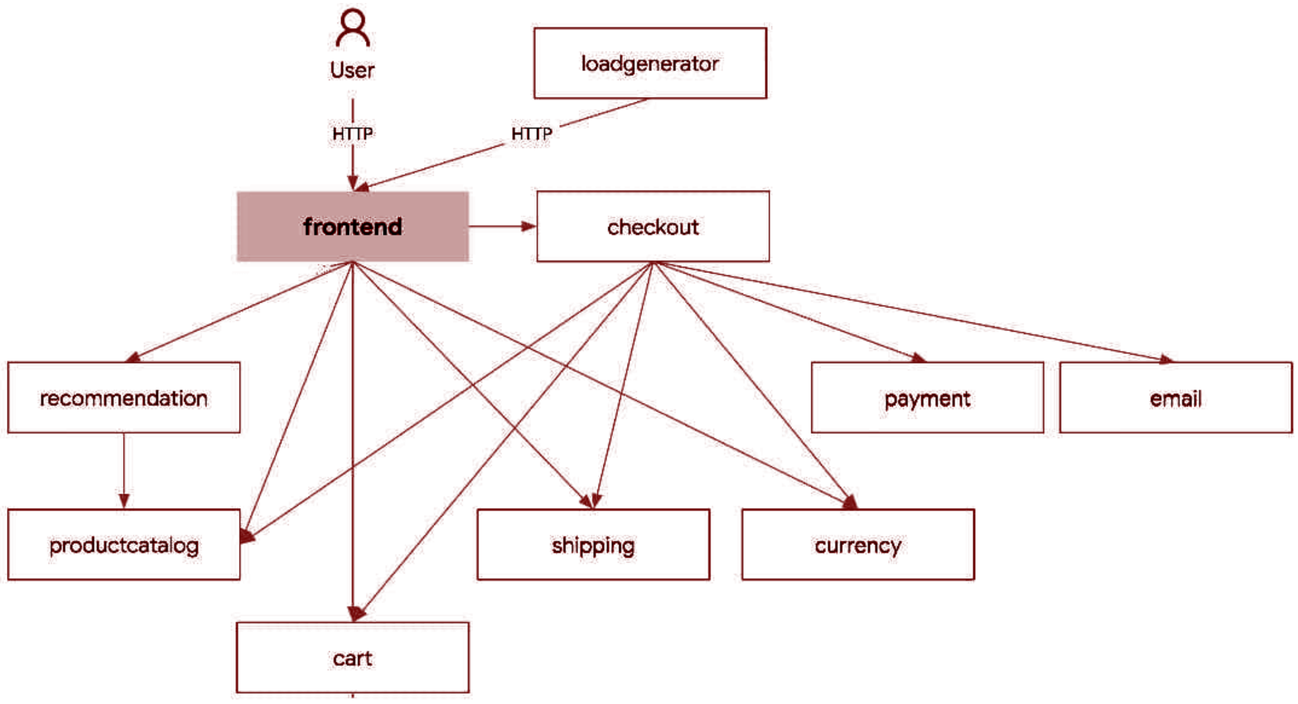
- Microservices communicate with each other using lightweight protocols such as HTTP.
- This decoupled communication allows for better fault tolerance and resilience.

Benefits of Microservices in the E-commerce Website:

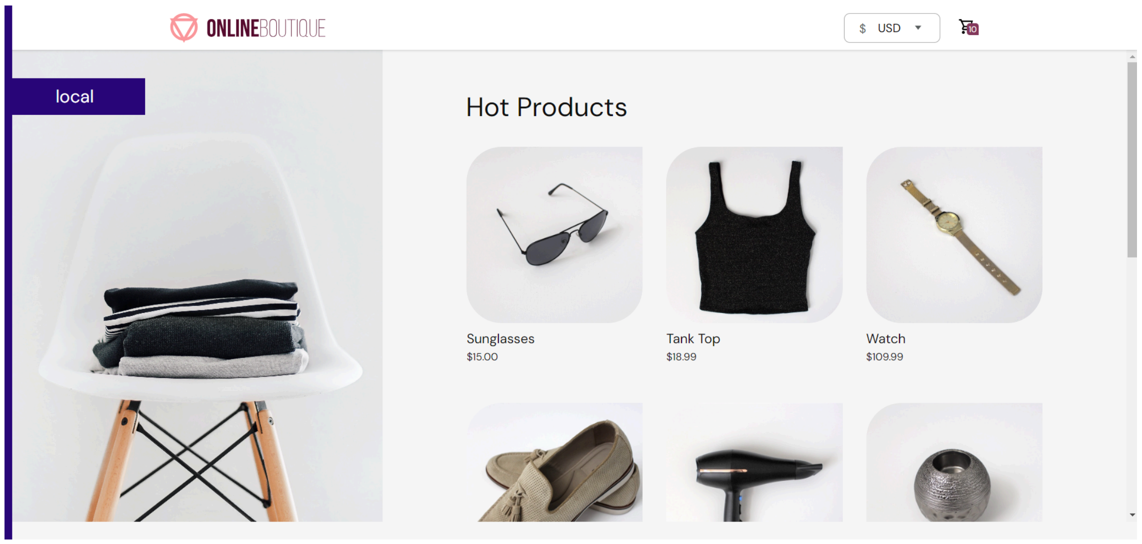
1. Each microservice can be scaled independently based on demand, ensuring optimal resource utilization.
2. We have the flexibility to use different technologies for different microservices, enabling innovation and faster development cycles.
3. A failure in one microservice does not impact the entire system, improving overall fault tolerance.
4. Easier deployment and updates of individual microservices, leading to faster release cycles and quicker delivery of new features.

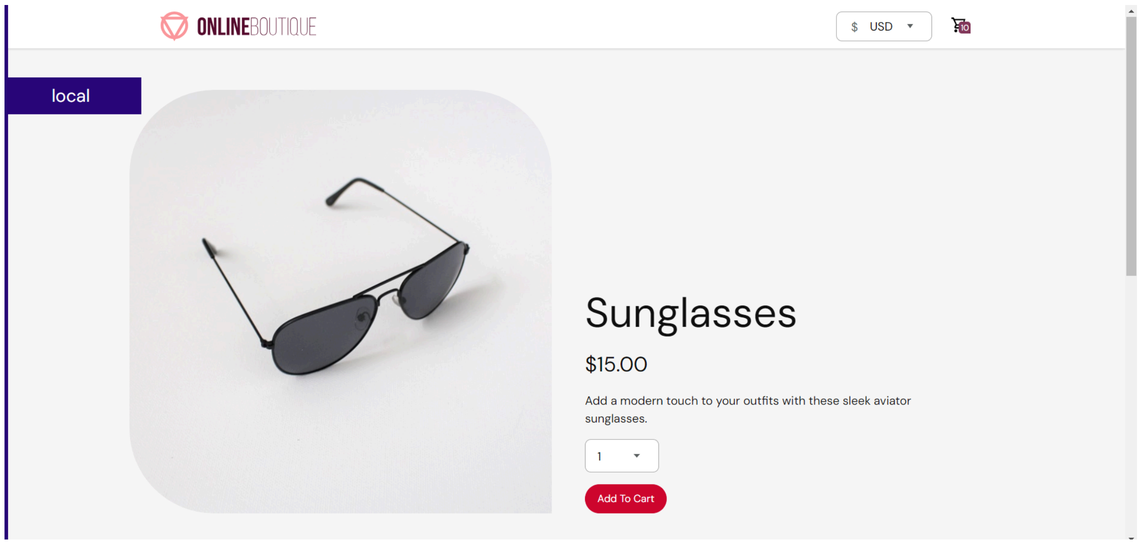
Challenges and Considerations:

1. Complexity: Managing and coordinating multiple microservices can introduce complexity in terms of monitoring, testing, and deployment.
2. Security: Securing the communication between microservices and managing access control is crucial to prevent unauthorized access.

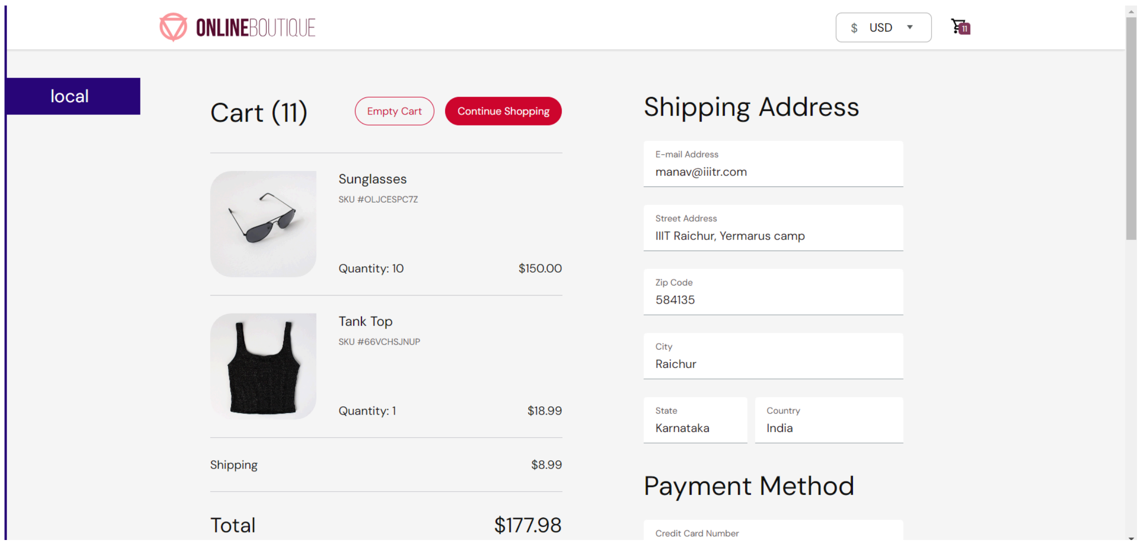

User
HTTP
loadgenerator
HTTP
frontend
checkout
recommendation
payment
email
productcatalog
shipping
currency
cart

5. Website snaps





Add a modern touch to your outfits with these sleek aviator sunglasses.



Cart (11)

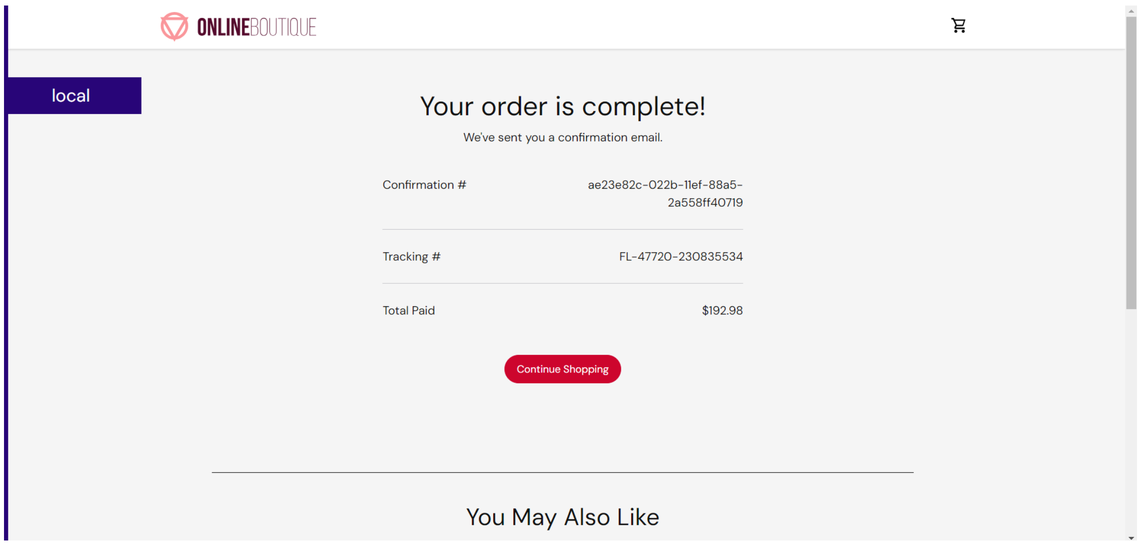
Sunglasses SKU #OLJCESPC7Z	Quantity: 10	$150.00
Tank Top SKU #66VCHSJNUP	Quantity: 1	$18.99
Shipping		$8.99
Total		$177.98

Shipping Address

E-mail Address: manav@iiitr.com
Street Address: IIIT Raichur, Yermarus camp
Zip Code: 584135
City: Raichur
State: Karnataka
Country: India

Payment Method

Credit Card Number

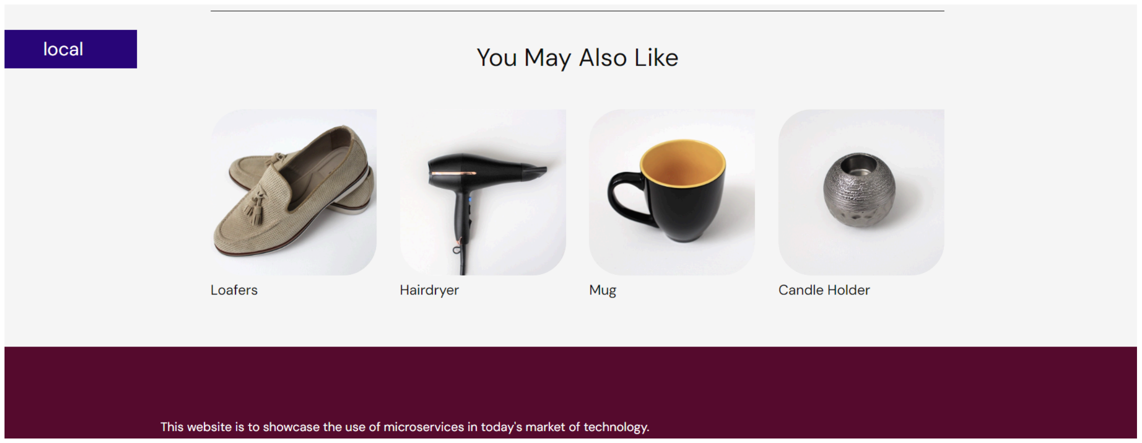


Your order is complete!

We've sent you a confirmation email.

Confirmation #	ae23e82c-022b-11ef-88a5-2a558ff40719
Tracking #	FL-47720-230835534
Total Paid	$192.98

You May Also Like



You May Also Like

Loafers Hairdryer Mug Candle Holder

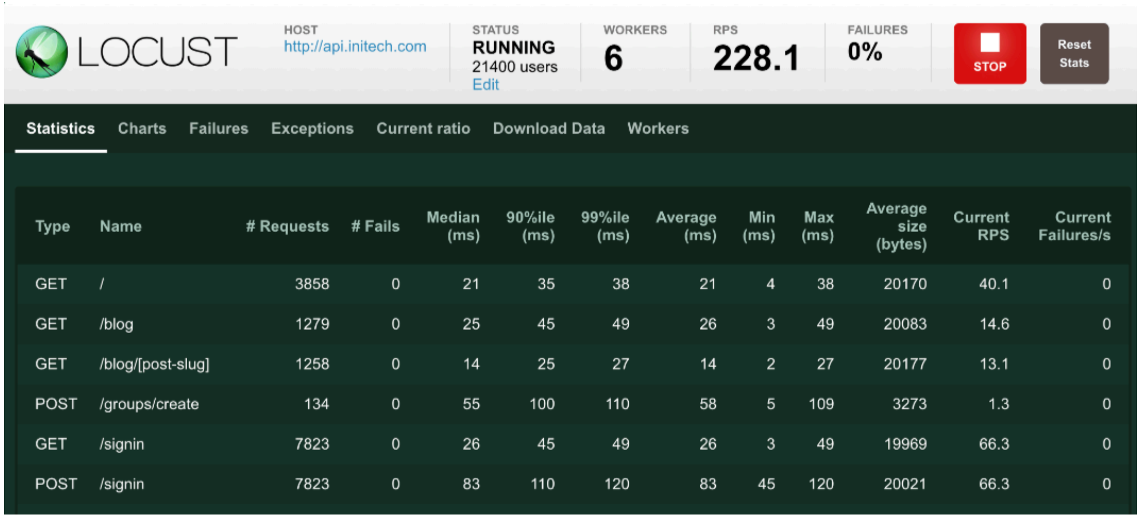
This website is to showcase the use of microservices in today's market of technology.

6. Testing

The testing of the website is done using locust. Locust is an open-source load-testing tool written in Python. It's designed to be highly scalable, allowing you to simulate thousands of users concurrently accessing your system.

With Locust, you can define user behavior using Python code, specifying the number of users to simulate, the rate at which they spawn, and the tasks they perform. This makes it flexible for testing various scenarios, such as different user flows, API endpoints, or application functions.

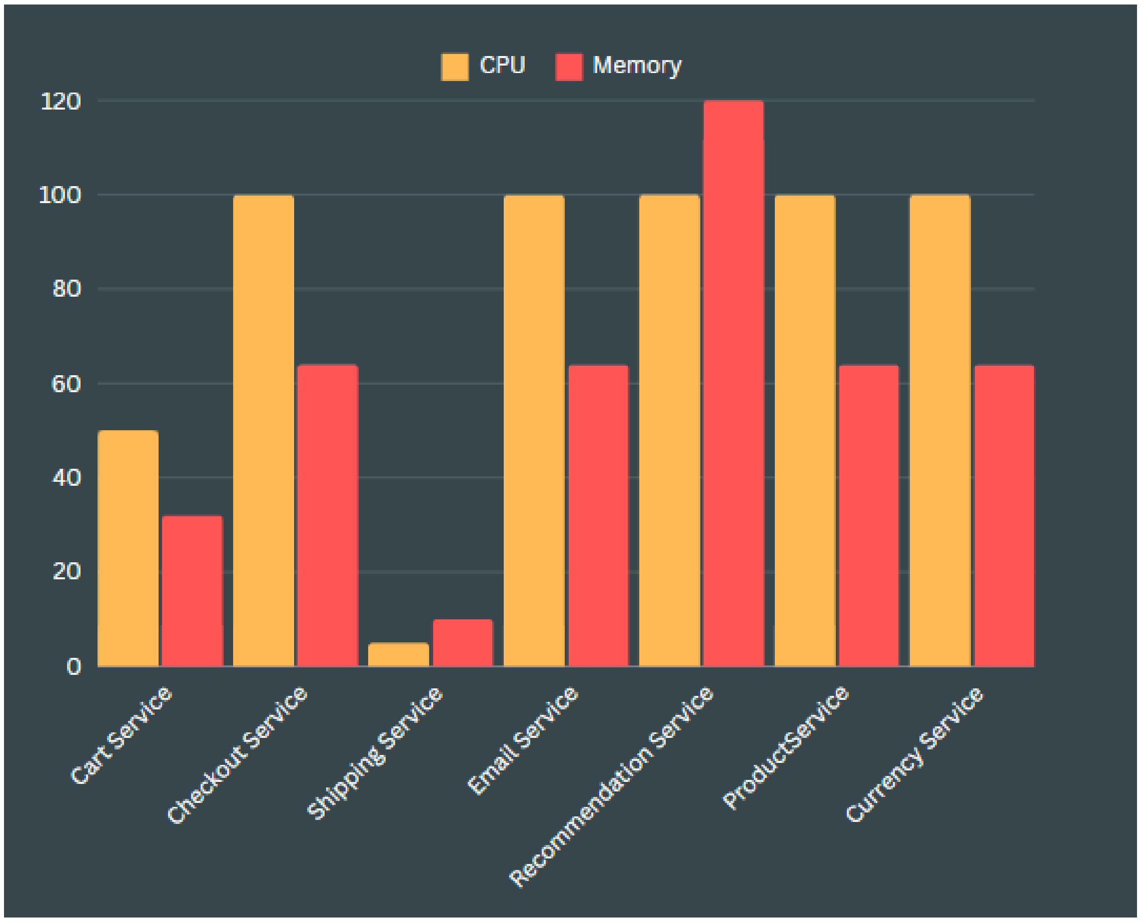
Locust provides a web-based interface where you can monitor the test progress in real time, view detailed statistics, and analyze the performance of your system under different loads. It's widely used for load-testing web applications and APIs to ensure they can handle the expected traffic and perform well under stress.

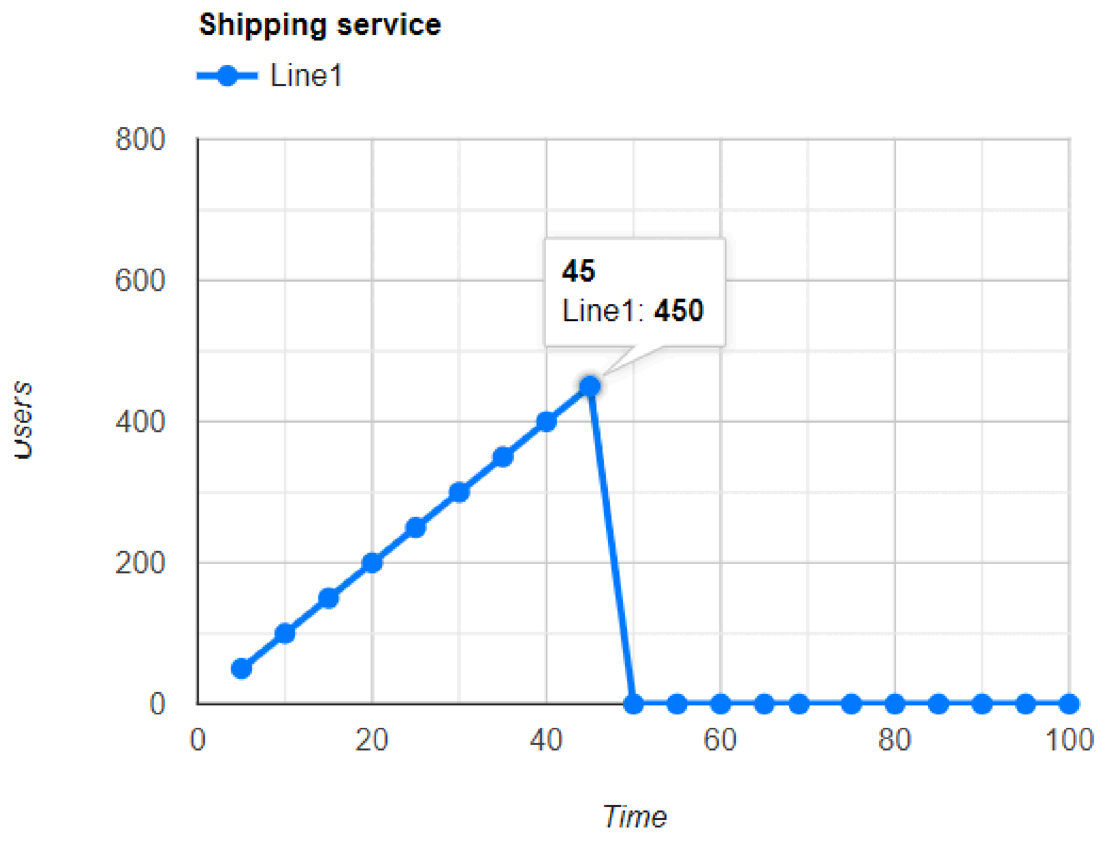


Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/	3858	0	21	35	38	21	4	38	20170	40.1	0
GET	/blog	1279	0	25	45	49	26	3	49	20083	14.6	0
GET	/blog/[post-slug]	1258	0	14	25	27	14	2	27	20177	13.1	0
POST	/groups/create	134	0	55	100	110	58	5	109	3273	1.3	0
GET	/signin	7823	0	26	45	49	26	3	49	19969	66.3	0
POST	/signin	7823	0	83	110	120	83	45	120	20021	66.3	0

A. PHASE I

The website has been tested for a large number of users using the locust software. The website has been given a large number of users per second until the time it crashed and we noted down at what time it crashed and made a graph of a certain service.

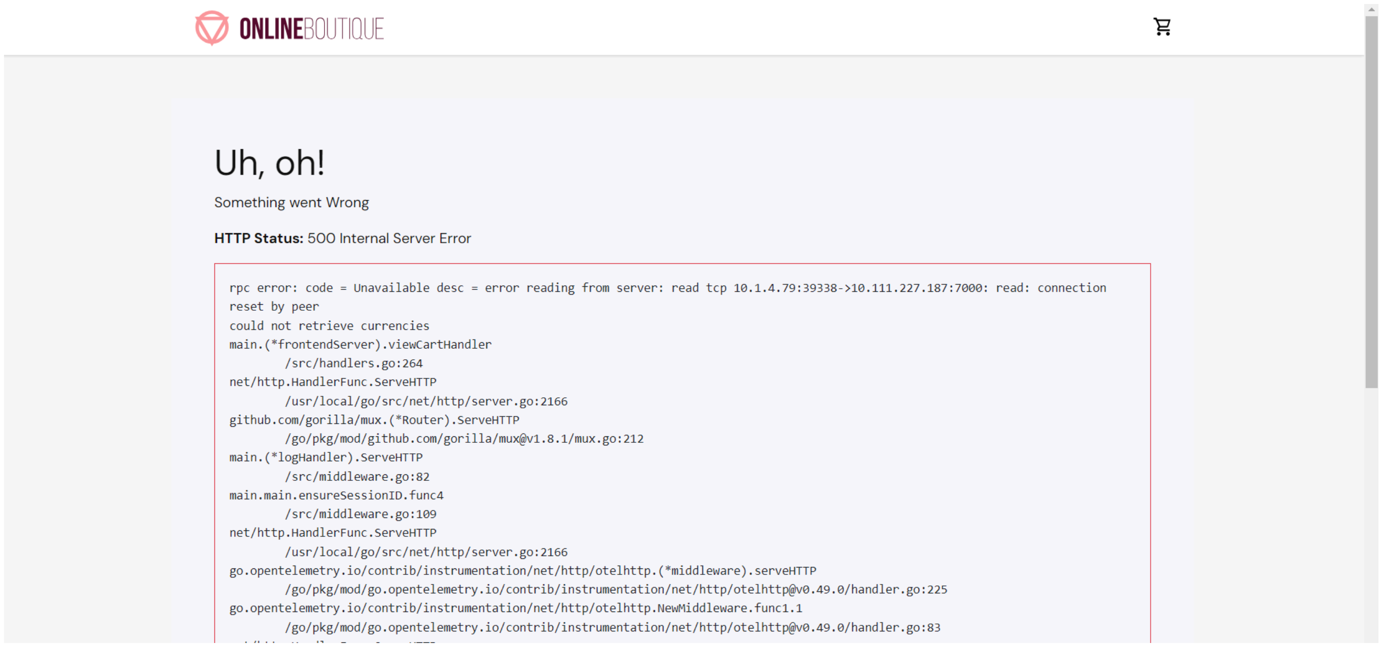




In the first graph, we can see the amount of resources provided to the shipping service. We deliberately kept the resources provided to the shipping service less so that it crashes and we can see when that happens.

B. PHASE II

Crash Report

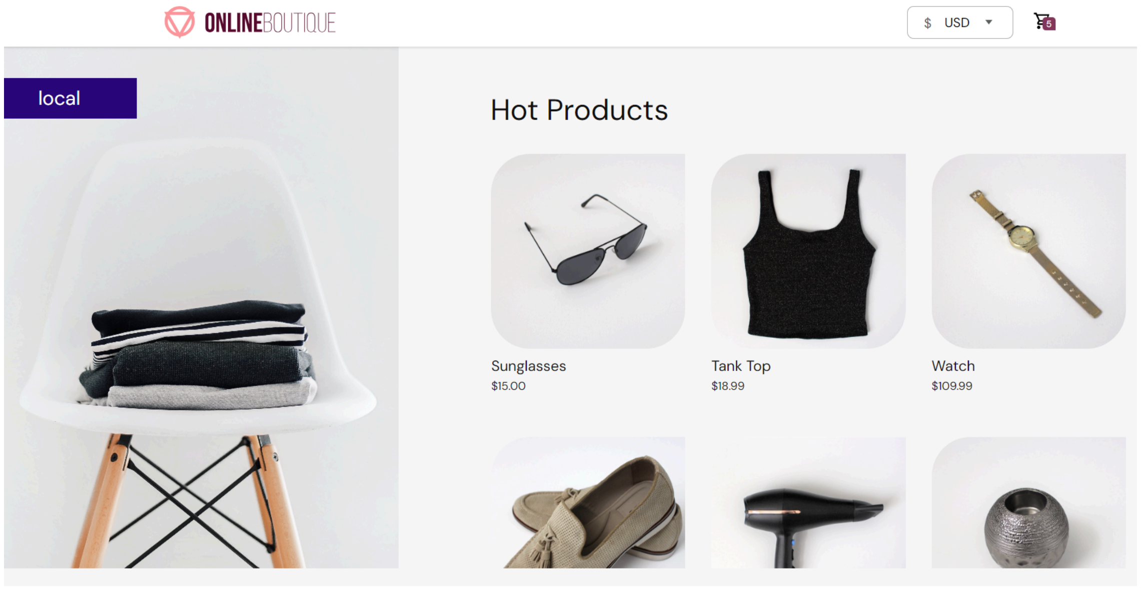


Uh, oh!

Something went Wrong

HTTP Status: 500 Internal Server Error

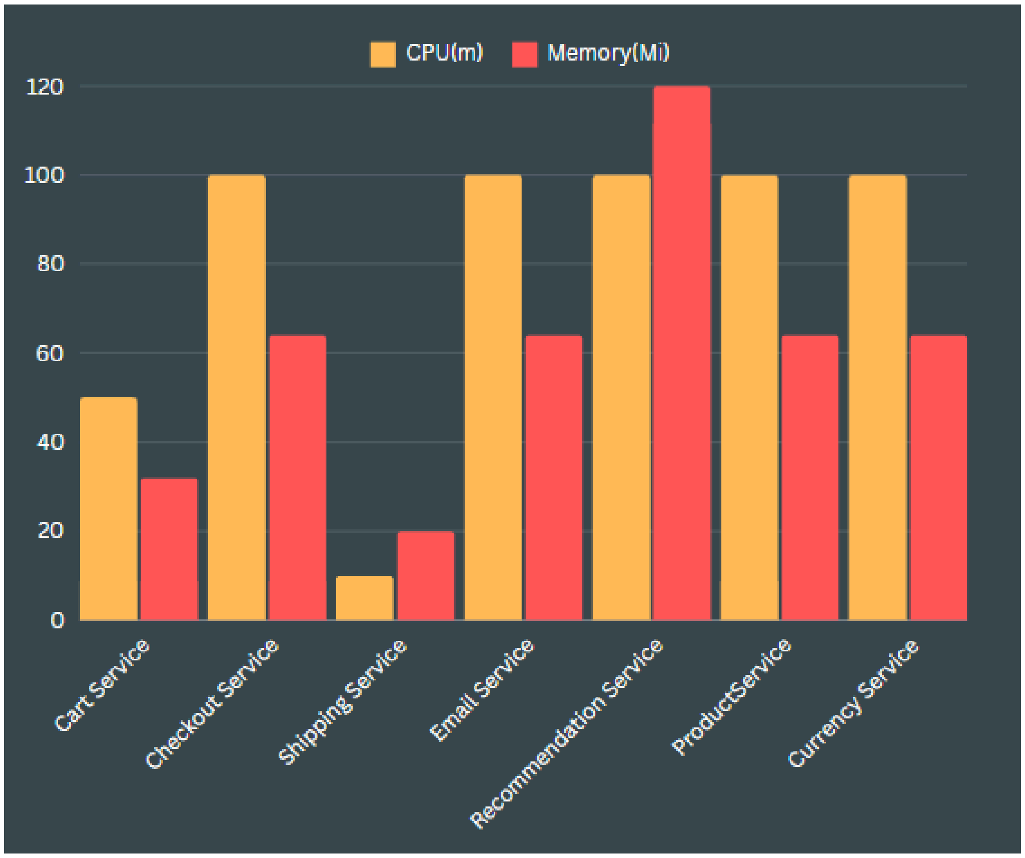
```
rpc error: code = Unavailable desc = error reading from server: read tcp 10.1.4.79:39338->10.111.227.187:7000: read: connection
reset by peer
could not retrieve currencies
main.(*frontendServer).viewCartHandler
        /src/handlers.go:264
net/http.HandlerFunc.ServeHTTP
        /usr/local/go/src/net/http/server.go:2166
github.com/gorilla/mux.(*Router).ServeHTTP
        /go/pkg/mod/github.com/gorilla/mux@v1.8.1/mux.go:212
main.(*logHandler).ServeHTTP
        /src/middleware.go:82
main.main.ensureSessionID.func4
        /src/middleware.go:109
net/http.HandlerFunc.ServeHTTP
        /usr/local/go/src/net/http/server.go:2166
go.opentelemetry.io/contrib/instrumentation/net/http/otelhttp.(*middleware).serveHTTP
        /go/pkg/mod/go.opentelemetry.io/contrib/instrumentation/net/http/otelhttp@v0.49.0/handler.go:225
go.opentelemetry.io/contrib/instrumentation/net/http/otelhttp.NewMiddleware.func1.1
        /go/pkg/mod/go.opentelemetry.io/contrib/instrumentation/net/http/otelhttp@v0.49.0/handler.go:83
```

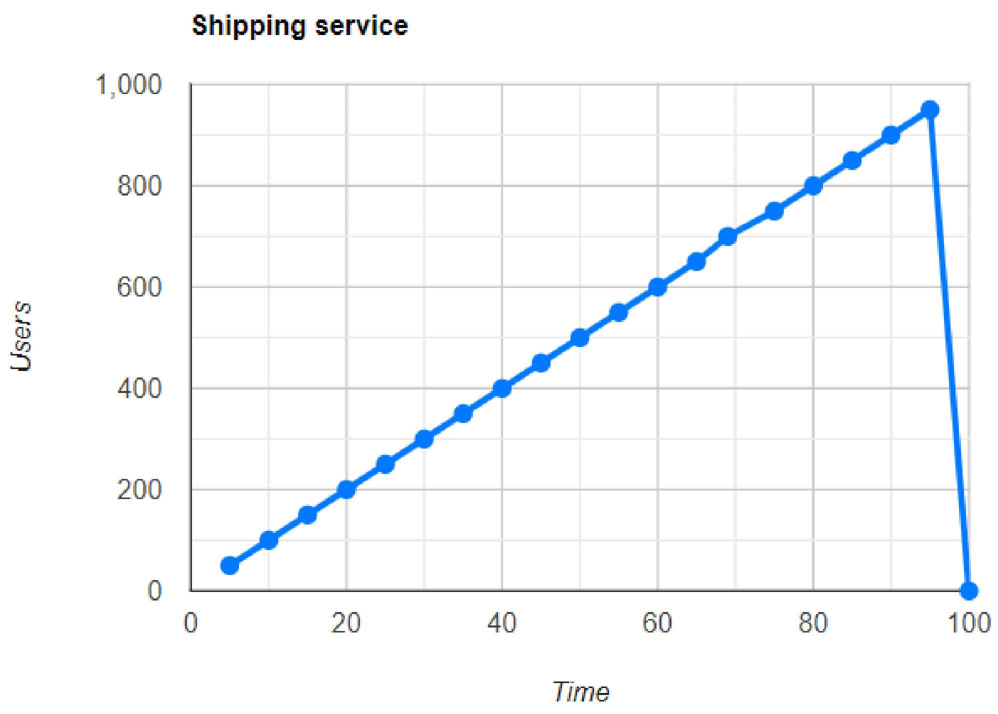


We can notice that even though one of the app services has crashed because of a huge load on its server other services are completely intact and working.

C. PHASE III

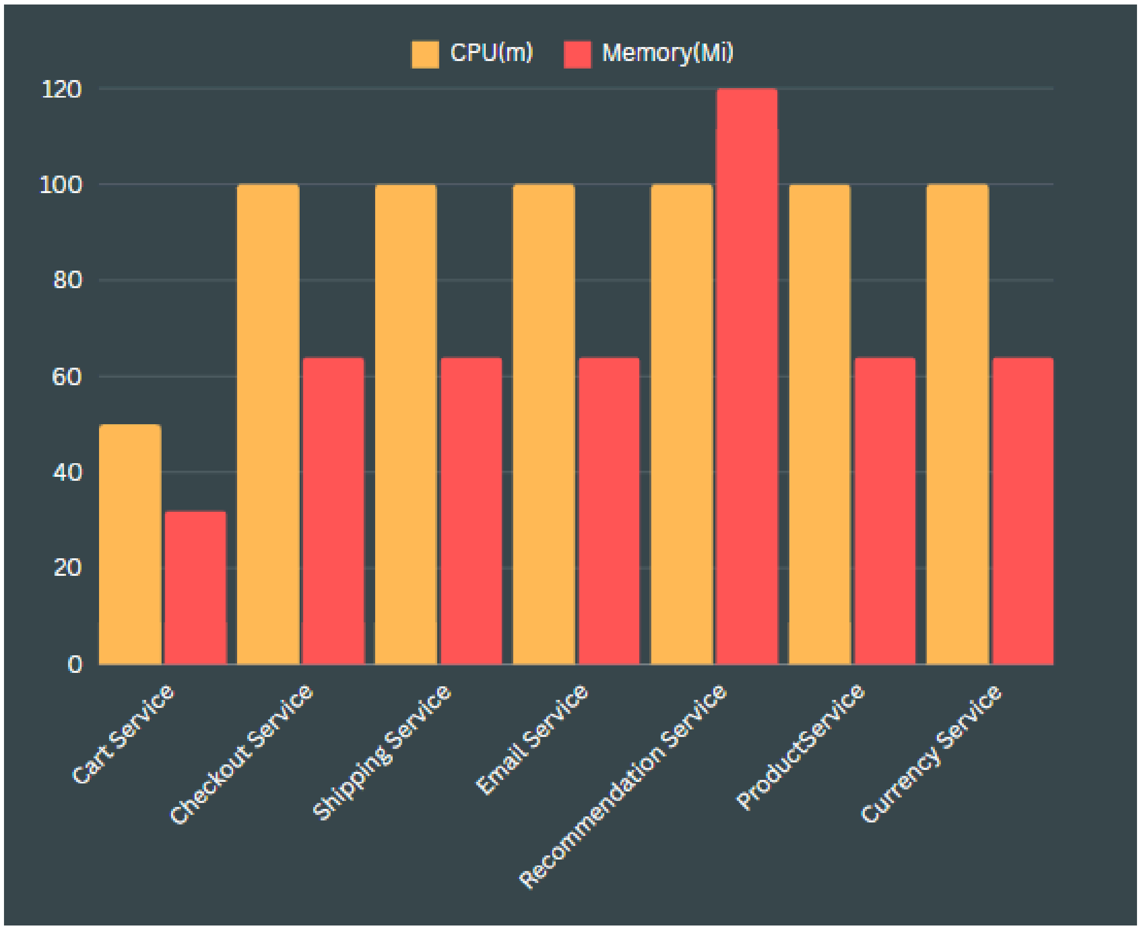
In the 3rd phase, we provided the shipping service with double the amount of resources that we provided earlier so that we could see the time at which it crashes and we could make an estimate about what amount of resources are needed when the number of users is huge.

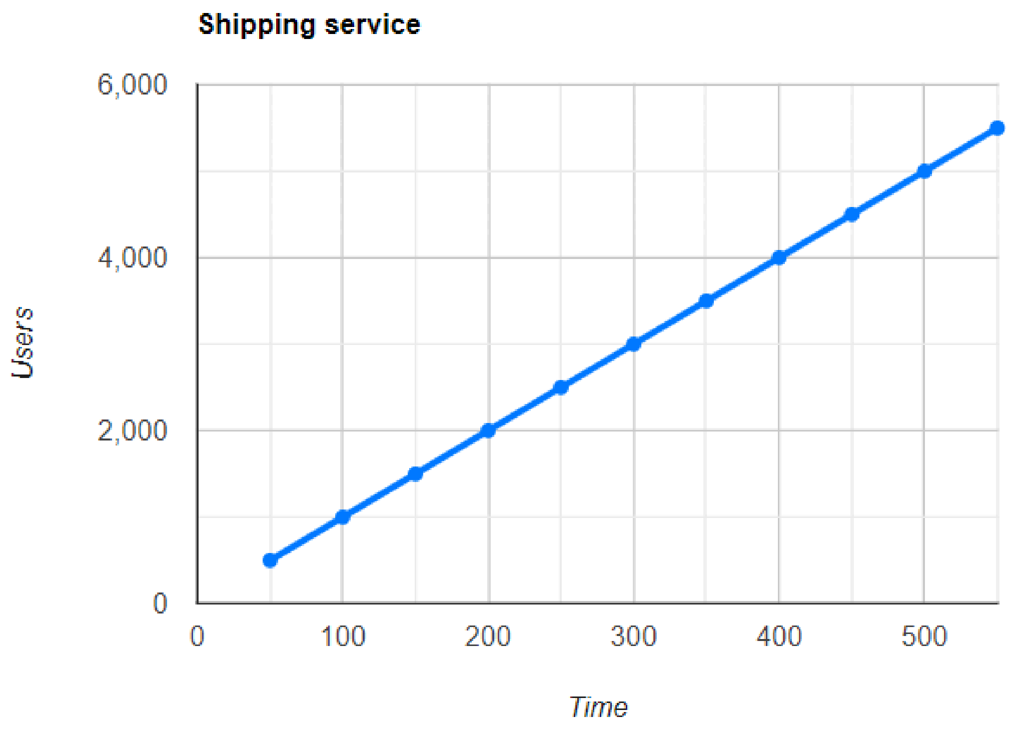




D. PHASE IV

In the last phase, we provided the service with a generous amount of resources to see the proper functioning of the website when it encounters a huge load on its servers.


CPU(m)
Memory(Mi)
120
100
80
60
40
20
0
Cart Service
Checkout Service
Shipping Service
Email Service
Recommendation Service
ProductService
Currency Service


Shipping service
Users
Time
6,000
4,000
2,000
0
0
100
200
300
400
500

7. Technologies

A list of software technologies that are used in the development of the software.

Languages:

- **Java**
- **Python**
- **Yaml**

Softwares:

- **Docker**
- **Kubernetes**
- **Locust**

8. Milestones

Milestone	Tasks
1 - Analysis	
1.1	Impact on market
1.2	Architecture design
1.3	System Requirements
2 - Development	
2.1	Learn New Technologies
2.2	Start creating a base app
2.3	Create GUI
2.4	Integrating with the server
3 - Testing	
3.1	Testing using locust
3.2	Testing for 100 Users
3.3	Testing for 10000 users per sec
3.4	Finalise documentation

9. Credits

- Aditya Aggarwal (Software Developer @ Amazon)
- (O'Reilly survey)
- Camunda Report
- Locust
- Internet