IOSR Journal of Engineering (IOSRJEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
Vol. 09, Issue 4 (April. 2019), ||S (II) || PP 38-45

Implementing Microservice Architecture for improving E-commerce websites performance

Pranit Mohata[1], Pritish Tijare[2]
[1]M.E. Computer Engineering student, Dept. of CSE, Sipna COET, Amravati, Maharashtra, India
[2]Associate Professor, Dept. of CSE, Sipna COET, Amravati, Maharashtra, India

Abstract: As client requirements are ever changing, continuous modification in the website needed to keep them up to date, so its biggest challenge is to keep the site up and avoid any downtime. Because downtime leads to loss in the earning and also it disappoints the user who uses E-commerce platform. Hence, servers need to be running all the time. We can minimize the downtime of Application by implementing Microservice Architecture. Single Responsibility Principal has been used by Microservice Architecture and they can be deployed independently. They can be scaled and tested independently. They are design and developed as a small application. Now a days, Microservices are gaining momentum across industries to facilitate agile delivery mechanisms for service-oriented architecture. With the help of Microservice architecture, we are able to migrate function-oriented legacy architectures toward highly flexible service orientation. The proposed Microservices Architecture delivers excellent speed and quality over Monolithic Architecture. In this paper, we have implemented Microservices architecture over monolithic architecture which helps in reducing downtime of the application which leads to increase in the performance.

Date of Submission: 27-03-2019 Date of acceptance: 12-04-2019

I. INTRODUCTION

A microservices architecture consists of a collection of services which are small and autonomous. Each service should implement a single business capability and is self-contained. Thus, these applications are loosely coupled. The basic idea of implementing microservice architecture is that, when application when broken down into smaller, composable pieces or components which work together are easier to deploy, build and maintain. Each service has a separate codebase, which can be managed by a small development team. This is in contrast to a traditional Monolithic architecture style where end application is developed all in one piece. In Monolithic Architecture, all the features of the application are written as a separate module that are then packaged into a single main application. We should use Microservice Architecture when there is large application and require a high release velocity, also when application is complex and needs to be highly scalable.

In Microservice Architecture, Services are responsible for persisting their own data or external state which differs from the traditional model, where a separate data layer handles data persistence. In Microservice Architecture, services will communicate with other services by using well-defined APIs where internal implementation details of each service are hidden from other services. Another important feature of Microservice Architecture is that, services don't need to share the same frameworks, technology stack or libraries. When there are large enterprise applications which are developed by teams of geographically and culturally diverse developers, this approach has been proven to be superior. End Applications which is built as a set of individual components are easier to understand and easier to test. The most important advantage of such application is that it is easier to maintain over the life of the application. It helps to achieve much higher agility for the organization and also helps to improve the time it takes to get working improvements to production.

II. LITERATURE REVIEW AND RELATED WORK

One of the intentions of microservice architectures is to overcome the scalability limit of monolithic architectures. A system has a microservice architecture when that system is composed of many collaborating microservices; typically, without centralized control [1]. The philosophy of the microservices architecture is: “Do one thing and do it well." Services might run within the same process, but they should be independently deployable and easy to replace [2]. Microservice architectures provide small services that may be deployed and scaled independently of each other, and may employ different middleware stacks for their implementation [3]. Wilhelm Hasselbring along with Guido Steinacker compared the scalability in both monolithic and microservice architecture in one of the companies. They have also explained how developers are now able to do set up, deploy and scale microservices without any support from operations team. With microservices, applications can be scaled dynamically, depending on the current load that a single microservice is facing. Along with that, they have compared the Number of life deployments per week over the last two years and incidents raised because of that. As per the analysis, despite the significant increase of deployments, the number of live incidents remains on a very low level [3]. Traditionally, information system integration aims at achieving high data coherence among heterogeneous information sources [4], [5]. However, a great challenge with integrated databases is the inherently limited horizontal scalability of transactional database management [6].

Microservices are built around business capabilities and did full-stack implementation of software for business area. Automation is key to DevOps success: automated building of systems out of version management repositories; automated execution of unit tests, integration and system tests; automated deployment in test and production environments; including performance benchmarks [7]. Villamizar et al. compared the average response time between a monolithic and a microservice-based system in the cloud. In their test, systems were deployed to Amazon Web Services with similar hardware configuration [8]. Villamizar et al. describe their test results as that there is a less performance impact and both systems would fulfil the case study requirements where they had two services, S1 and S2. The case study requirements were defined as: “The service S1 implements CPU intensive algorithms to generate payment plan and their response time is around 3000 milliseconds.” and “The response time of service S2 is around 300 milliseconds and this service mainly consumes the database.”. Villamizar et al. further explain that hosting their solution would be 17% cheaper for the microservice architecture on Amazon Web Services [8]. Microservices architectures can be consider as departure from traditional Service Oriented Architecture. Influenced by Domain Driven Design, microservices architectures aim to help business analysts and enterprise architects develop scalable applications that embody flexibility for new functionalities as businesses develop, such as scenarios in the Internet of Things (IoT) domain [9].

Microservice Architectures have the potential to increase the agility of software development [10]. Microservices have recently emerged as an architectural style, addressing how to build, manage, and evolve architectures out of small, self-contained units [11].

III. ANALYSIS OF PROBLEM

A monolithic application is built as a single unit which makes the application tightly coupled and entangled as the application evolves, thus making it difficult to isolate services for purposes such as code maintainability or independent scaling. A monolith is in short single logical executable due to which for any alterations to the system, a developer must build and deploy an updated version of the server-side application. Monolithic architectures are much harder to understand, because there may be dependencies, side-effects, etc. which are not obvious when you’re looking at a particular controller or service. Monolithic application is unreliable, even if single feature of the system caused any issue or does not work, then entire application goes down. Development in monolithic application is quite slow and take lot of time, as we need to build each and every module even after small change in the application. It is not even suitable for complex applications as it makes the application tightly coupled which is not recommended while building large application.

We cannot use different technologies while building monolithic application. For example, if the application is made in Java, then all the components needs to be written in Java and which can be packaged into a war file and then deployed to available server such as tomcat, jboss or a jetty server. Sometimes so many iterations during application development make the application bigger than expected and as we need keep on adding new features, it makes it huge. Once it happens, rapid and effective development and delivery becomes much more difficult. Along with that, bug tracking and fixing also becomes more difficult to handle. Thus, these blocking difficulties result in extravagant time usage. Even for a minor bug fixing one will require to re-deploy whole application. Another big disadvantage is scaling of the application. If we want to scale, we need to scale the entire monolithic application, even though we will only need more resources on one of the components of the large application.

IV. PROPOSED WORK

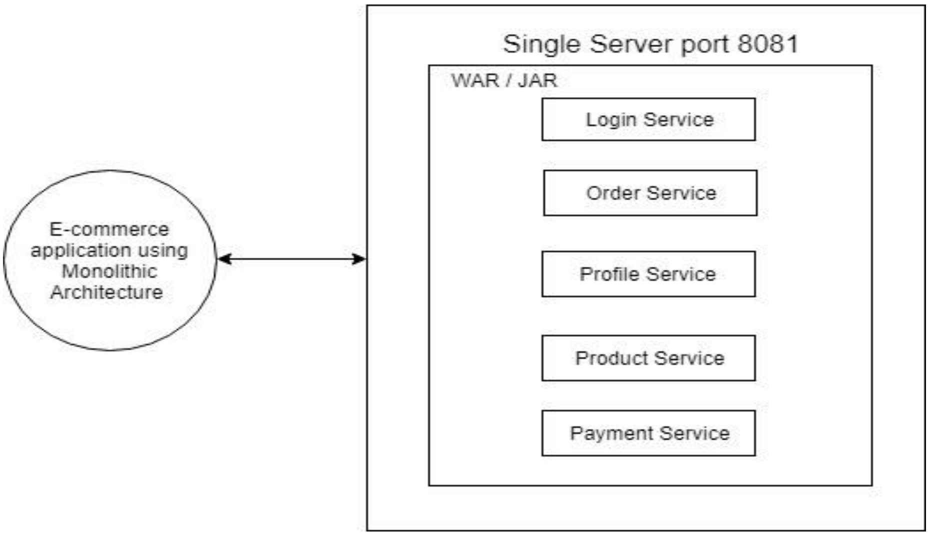
Microservice architectures are typically better organized, as each microservice follows single responsibility principal means each service has a very specific job, and it’s not concerned with the jobs of other components. Thus, making Microservices are loosely coupled. De-coupled services are also easier to reconfigure and recompose to serve the purposes of different applications. They can also perform better depending on how they’re organized because it’s possible to isolate frequently called services usually termed as hot services and scale them independent of the rest of the app. These services will largely de-coupled, so that application can be easily built, altered and scaled. Microservice architecture support agile development, we can easily develop any new feature if required or even we can discard if not in use. We can have independent deployment of any individual module and these can also be developed independently. This approach makes sure that even if any particular feature of the application failed, other part of the application will keep on working. Thus, making the downtime of the application as much less as possible. As downtime of the application and business profit are directly proportional to each other, if there is more downtime of the application, there are greater chances of loss in business revenue. But if there is no or minimal downtime of the application, it will be profitable for the business.

The proposed module will implement microservice architecture and will compare the deployment time required for Monolithic application over Microservice application. The result of the project will be analyzed. Our main objective is to reduce the downtime as much as possible by implementing microservice architecture than our legacy monolithic architecture. The proposed project has an approach to software development in which an application is built as a small, independently versioned, and customer-focused services which are scalable and with specific business goals, which will have well defined interfaces and which will communicate with each other over standard protocols. Each service will implement a single business logic and is self-contained.

V. SYSTEM ARCHITECTURE

A. Monolithic Architecture:

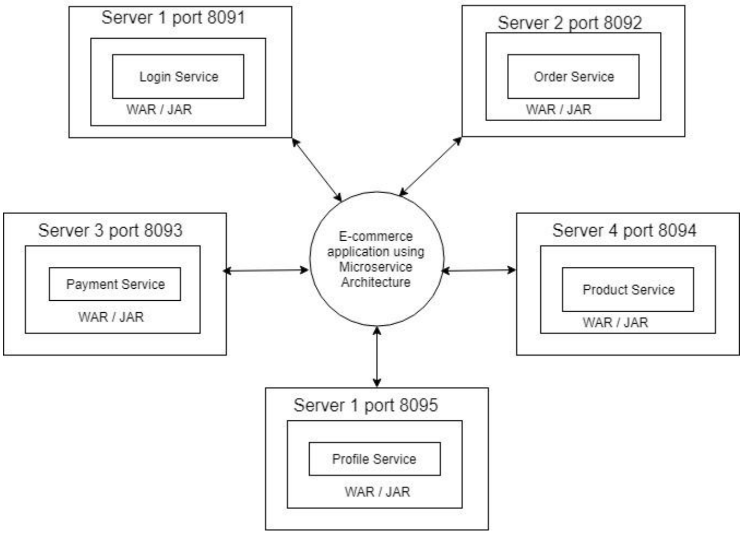
Below figure describe Monolithic Architecture where all the business logic is packaged into single application. There are five different services Login, Order, Profile, Products and Payment service which are doing login into the application, order details for the user, displaying profile information about the logged in user, display product list and display different banks available for payment respectively. These are then deployed as a single application either as war or jar.



B. Microservices Architecture:

Below figuredescribe Microservice Architecture where each and every module has its own functionality. For Example, Login service will do login related work for the user, Profile service will give the profile details of the user. These modules are deployed in different server so that if there is any change in any one of the modules, then in that case only module where changes are required need to deploy, not whole application. Thus, it will reduce the downtime of the application. Even if one particular service has lots of hits, then in that case we can scale only that particular module, not whole application.

Also, in future if we need to change the technology of particular module, then this is also possible in Microservice Architecture. In this way we can have multiple technologies that can exist with each other in Microservice Architecture.



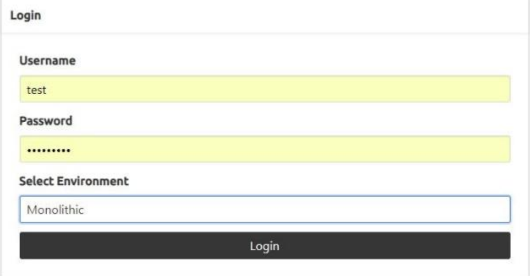
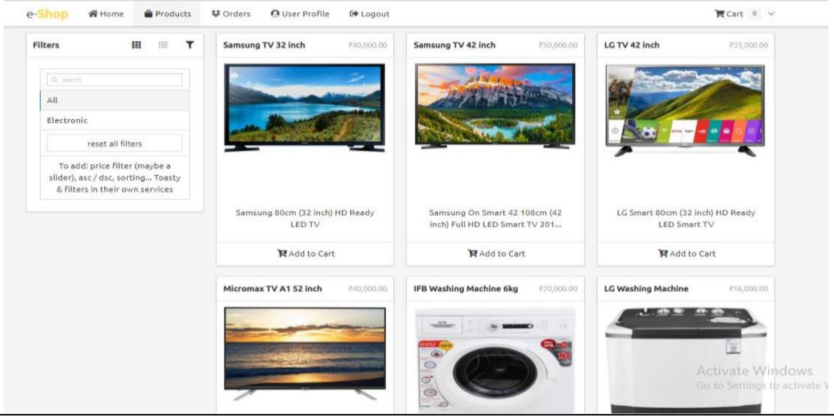
VI. IMPLEMENTATION

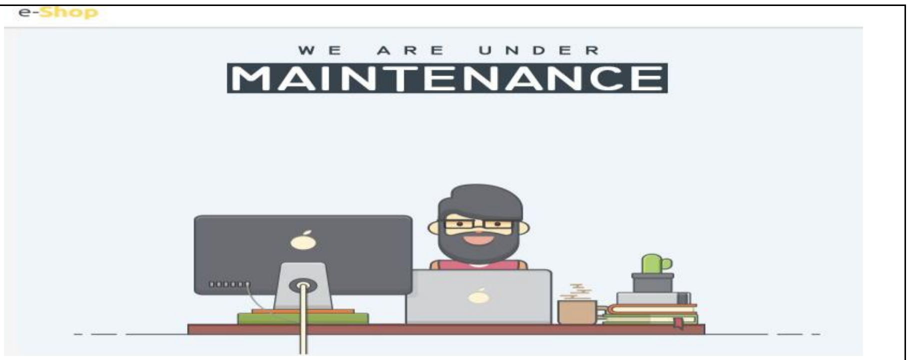
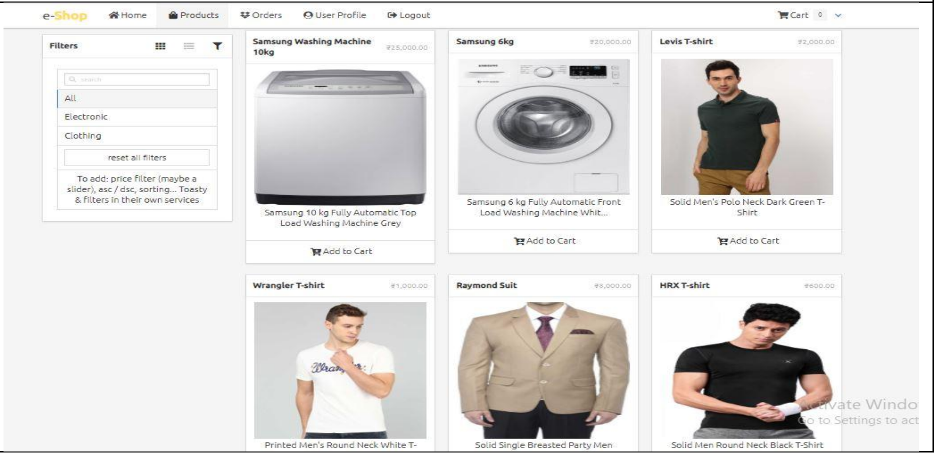
For implementation part, we have developed an E-commerce look alike site which has basic services such as Login, Order, Payment, Profile and Product details. These services have been written for both monolithic application as well as microservice application.

A. Application implementing Monolithic Architecture

In case of Monolithic architecture, we have all the above-mentioned services deployed on single server. But considering the fact that for any e-commerce site introduction of new functionality on frequent basis is not new, so if client comes up with new functionality, we need to re-deploy whole application even if the changes required are very less. In this case, our application will be down till deployment with new changed is not completed. Hence it creates unnecessary downtime.

For ex. Initially we have an e-commerce platform which has only electronic items. But client now wants their site should have clothing line-up also. In this case, we need to deploy whole application again to introduce the new changes even though changes are only implemented in Product details service. Because of re-deployment, we will get downtime during deployment process. Once deployment is successful, we will be able to login into application and can view both Electronic as well as Clothing section.

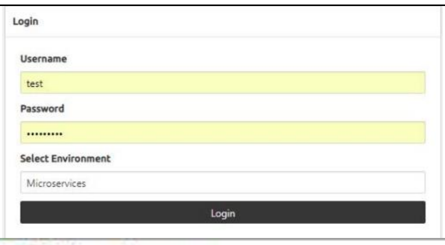
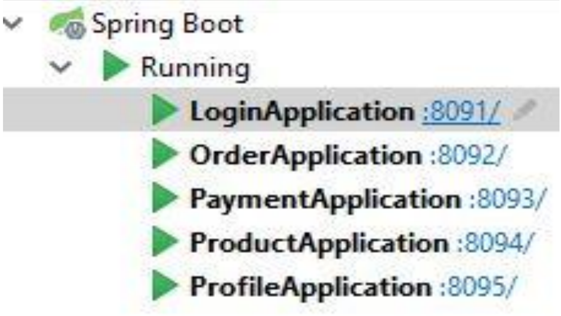
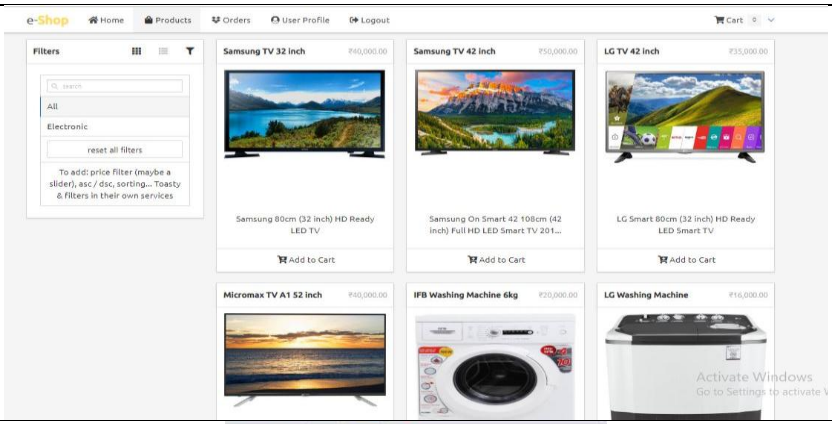
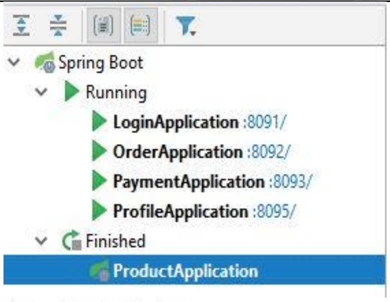
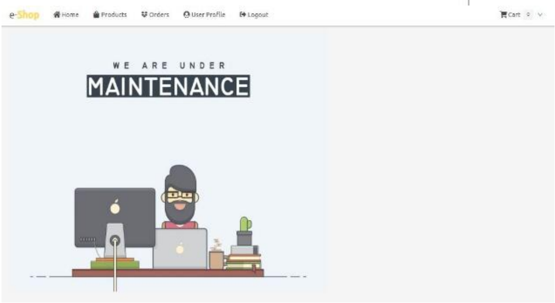
Step 1: Wewill login intoour web application. At present, this site has onlyElectronic items available for purchasing. It willalsogive us Profile details and pastorderdetails of the logged in user	 Login Username test Password ••••••••• Select Environment Monolithic Login
Step 2: After login, we can see that Electronic Product list is displayed. Along with that, it gives a profile details of the logged in use as well as order history for that user.	 e-Shop Home Products Orders User Profile Logout Cart 0 Filters search All Electronic reset all filters To add: price filter (maybe a slider), asc / dsc, sorting... Toasty & filters in their own services Samsung TV 32 inch ₹40,000.00 Samsung 80cm (32 inch) HD Ready LED TV Add to Cart Samsung TV 42 inch ₹50,000.00 Samsung On Smart 42 108cm (42 inch) Full HD LED Smart TV 201... Add to Cart LG TV 42 inch ₹35,000.00 LG Smart 80cm (32 inch) HD Ready LED Smart TV Add to Cart Micromax TV A1 52 inch ₹40,000.00 IFB Washing Machine 6kg ₹20,000.00 LG Washing Machine ₹16,000.00 Activate Windows Go to Settings to activate W

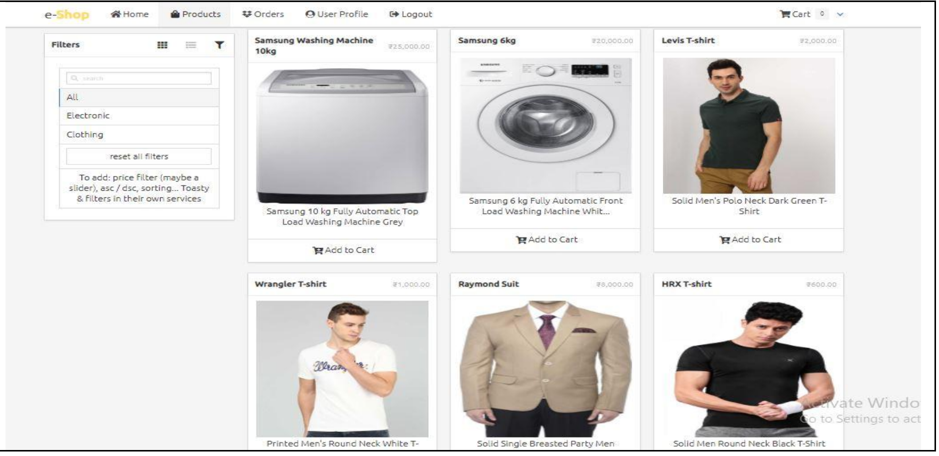
Step 3 : There is requirement change from the customer in our current application. Now customer wants to add clothing along with Electronic section. So, after doing changes in our application, we need to deploy it on server, so that users can see both Electronic item as well as clothing items. Step 4: For deployment purpose, server needs to stop. So, we will stop the server. In case of Monolithic application, our application will be in Maintenance state till the new changes are deployed. Hence, end user will not be able to login into the application	
Step 5: Once deployment is completed, user will be able to login into the web application and new changes will be reflected in our application	 e-Shop Home Products Orders User Profile Logout Cart 0 Filters search All Electronic Clothing reset all filters To add: price filter (maybe a slider), asc / dsc, sorting... Toasty & Filters in their own services Samsung Washing Machine 10kg ₹25,000.00 Samsung 10 kg Fully Automatic Top Load Washing Machine Grey Add to Cart Samsung 6kg ₹20,000.00 Samsung 6 kg Fully Automatic Front Load Washing Machine Whit... Add to Cart Levis T-shirt ₹2,000.00 Solid Men's Polo Neck Dark Green T-Shirt Add to Cart Wrangler T-shirt ₹1,000.00 Printed Men's Round Neck White T- Raymond Suit ₹8,000.00 Solid Single Breasted Party Men HRX T-shirt ₹600.00 Solid Men Round Neck Black T-Shirt ctivate Windo o to Settings to act

B. Application implementing Microservice Architecture

In case of Microservice architecture, we have all the above-mentioned services deployed on different server as Microservice architecture follows Single Responsibility Principal. But considering the fact that for any e-commerce site introduction of new functionality on frequent basis is not new, so if client comes up with new functionality, we need to deploy only that service in which changes needs to be incorporated. In this case, only that service in which changes are introduce will be down until deployment is completed, not whole application.

For ex. Initially we have an e-commerce platform which has only electronic items. But client now wants their site should have clothing line-up also. In this case, we need to deploy only product details service again to introduce the new changes. Once deployment is successful, we will be able to see both Electronic as well as Clothing section

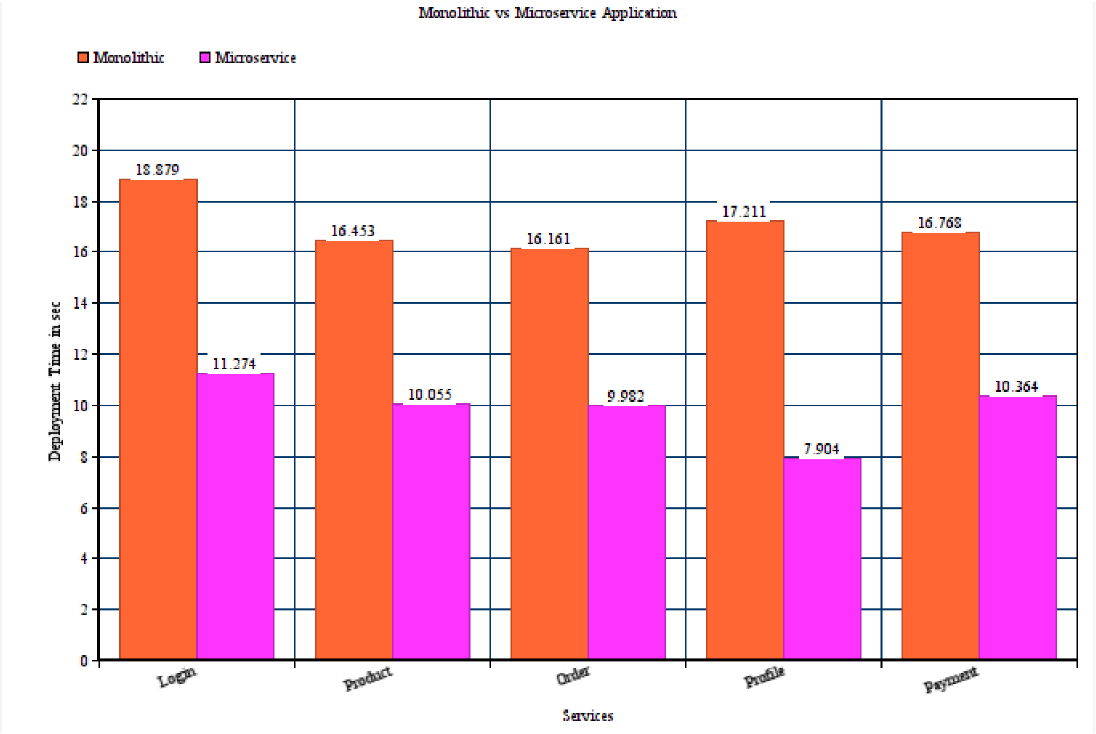
Step 1 : Wewill login intoour web application. At present, this site has onlyElectronic items available for purchasing. It willalsogive us Profile details and pastorderdetails of the logged in user.	 Login Username test Password ••••••••• Select Environment Microservices  Spring Boot Running LoginApplication :8091/ OrderApplication :8092/ PaymentApplication :8093/ ProductApplication :8094/ ProfileApplication :8095/
Step 2 : After login, we can see that Electronic Product list is displayed. Along with that, it gives a profile details of the logged in use as well as order history for that user.	 Filters search All Electronic reset all filters To add: price filter (maybe a slider), asc / dsc, sorting... Toasty & filters in their own services Samsung TV 32 inch ₹40,000.00 — Samsung 80cm (32 inch) HD Ready LED TV — Add to Cart Samsung TV 42 inch ₹50,000.00 — Samsung On Smart 42 108cm (42 inch) Full HD LED Smart TV 201... — Add to Cart LG TV 42 inch ₹35,000.00 — LG Smart 80cm (32 inch) HD Ready LED Smart TV — Add to Cart Micromax TV A1 52 inch ₹40,000.00 IFB Washing Machine 6kg ₹20,000.00 LG Washing Machine ₹16,000.00 Activate Windows Go to Settings to activate V
Step 3: There is requirement change from the customer in our current application. Now customer wants to add clothing along with Electronic section. So, after doing changes in our application, we need to deploy it on server, so that users can see both Electronic item as well as clothing items. Step 4: For deployment purpose, server needs to stop. So, we will stop the server. In case of Microservices application, we need to stop only that server where Product details service is called. Hence, we will stop Product service only. In this case, user will be able to login into our web application. He will get Maintenance page on Product list page, but user will be able to see his rest of the details such as Profile and Order history, etc.	 Spring Boot Running LoginApplication :8091/ OrderApplication :8092/ PaymentApplication :8093/ ProfileApplication :8095/ Finished ProductApplication  WE ARE UNDER MAINTENANCE

Step 5: Once deployment is completed, user will be able to login into the web application and new changes will be reflected in our application	

e-Shop
Home
Products
Orders
User Profile
Logout
Cart 0
Filters
search
All
Electronic
Clothing
reset all filters
To add: price filter (maybe a slider), asc / dsc, sorting... Toasty & Filters in their own services
Samsung Washing Machine 10kg
₹25,000.00
Samsung 10 kg Fully Automatic Top Load Washing Machine Grey
Add to Cart
Samsung 6kg
₹20,000.00
Samsung 6 kg Fully Automatic Front Load Washing Machine Whit...
Add to Cart
Levis T-shirt
₹2,000.00
Solid Men's Polo Neck Dark Green T-Shirt
Add to Cart
Wrangler T-shirt
₹1,000.00
Printed Men's Round Neck White T-
Raymond Suit
₹8,000.00
Solid Single Breasted Party Men
HRX T-shirt
₹600.00
Solid Men Round Neck Black T-Shirt
Activate Windo
Go to Settings to act

VII. RESULT

We have compared the deployment time required for both Monolithic and Microservice architecture for each service present in our application. Its graphical representation can be shown as follow:



From above graph, we can conclude that the time required for deploying monolithic application is greater than that of application build on Microservice Architecture. This results in less downtime in case of Microservice implemented application than that of Monolithic application.

VIII. CONCLUSION

We have created a system which will help to understand the benefits and advantages of microservices over monolithic architecture by comparing downtime, Complexity and scalability of application.

REFERENCES

[1]. S. Newman, Building Microservices. O'Reilly, 2015.

[2]. https://dzone.com/articles/benefits-amp-examples-of-microservices-architectur

[3]. Microservice Architectures for Scalability, Agility and Reliability in E-Commerce by Wilhelm Hasselbring and Guido Steinacker

[4]. "Information system integration," W. Hasselbring, Communications of the ACM, vol.43, no. 6, pp. 32–36, 2000

[5]. IEEE Multimedia, vol. 9, no. 1, pp. 16–25, 2002. “Web Data Integration for E-Commerce Applications,”.

[6]. M. Abbott and M. Fisher, The Art of Scalability, 2nd ed. Addison- Wesley, 2015

[7]. J. Waller, N. C. Ehmke, and W. Hasselbring, ACM SIGSOFT Softw. Eng. Notes, vol. 40, no. 2, pp. 1–4, Mar. 2015 “Including performance benchmarks into continuous integration to enable DevOps,”

[8]. M. Villamizar, O. Garcés, H. Castro, M. Verano, L. Salamanca, R. Casallas, and S. Gil. Evaluating monolithic and microservice architecture pattern to deploy web applications in the cloud. In 2015, 10th Computing Colombian Conference (10CCC), pages 583–590, Sept 2015.

[9]. Dharmendra Shadija, Mo Rezai, and Richard Hill. 2017. Towards an Understanding of Microservices. In 23rd IEEE International Conference on Automation and Computing. IEEE Computer Society, Huddersfield, UK.

[10]. Microservices: Granularity vs. Performance by Dharmendra Shadija, Mo Rezai and Richard Hill

[11]. Thones. 2015. Microservices. IEEE Software 32, 1 (2015)

Pranit Mohata. “Implementing Microservice Architecture for improving E-commerce websites performance.” IOSR Journal of Engineering (IOSRJEN), vol. 09, no. 04, 2019, pp. 38-45.