A Comparative Study: Monolithic vs. Microservices Architectures

Midterm Report

Aytan Gurbanova

July 2023

Original Proposal:

In the initial proposal, the research project aimed to compare monolithic architecture with microservices architecture. The main objective was to evaluate their advantages, disadvantages, and suitability for different use cases. The study would begin by analyzing various industries to determine which architecture is more preferred. The analysis would then delve into exploring key aspects such as complexity, scalability, deployment, monitoring, and other relevant factors.

The plan included conducting a thorough literature review on existing research and case studies related to monolithic and microservices architectures. Based on research and literature review, the study would define specific comparison criteria. These criteria would cover various aspects, including but not limited to development complexity, scalability, performance, fault tolerance, deployment strategies, monitoring and observability, and maintenance.

Upon receiving feedback from the professors, the research project was modified to focus primarily on performance and involve the development of a simple e-commerce application in both architectural styles. This change was made to provide a practical demonstration and better understanding of the two architectures.

Revised Proposal:

The revised proposal includes conducting a comparative analysis of monolithic and microservices architectures, with a primary emphasis on performance. To achieve this goal, an e-commerce application has been selected for development in both architectural styles. The application will consist of five database tables and seven main services. However, for the sake of simplicity, the administration-side functionality will be directly controlled through the database, and only user-side functionality will be implemented.

The e-commerce application development process will involve designing and implementing both a monolithic and a microservices version. The monolithic architecture will have all the services integrated into a single codebase, while the microservices architecture will separate the services into individual components communicating through well-defined APIs.

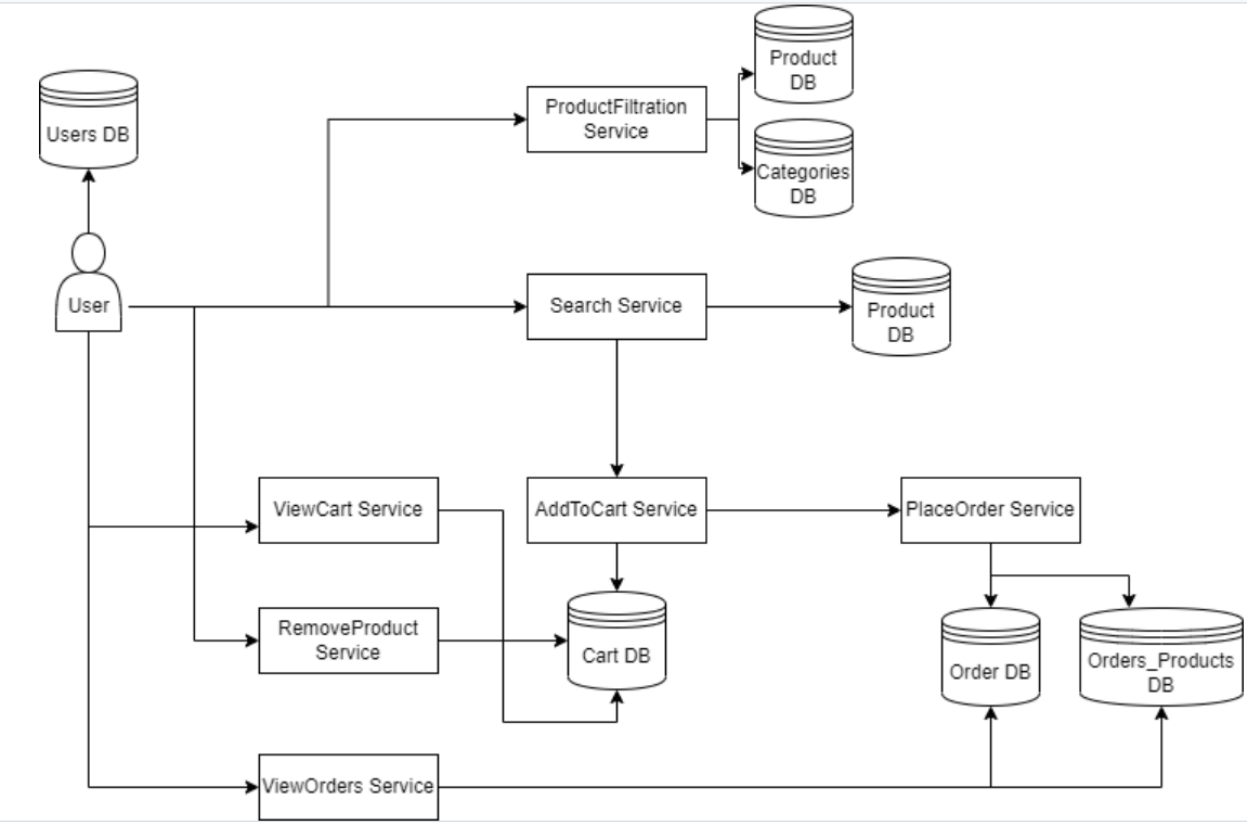
The performance of the two architectures will be evaluated using predefined metrics, such as response time, throughput, and resource utilization.

Additionally, qualitative factors like development productivity, ease of maintenance, and extensibility will be assessed through developer feedback and subjective evaluation.

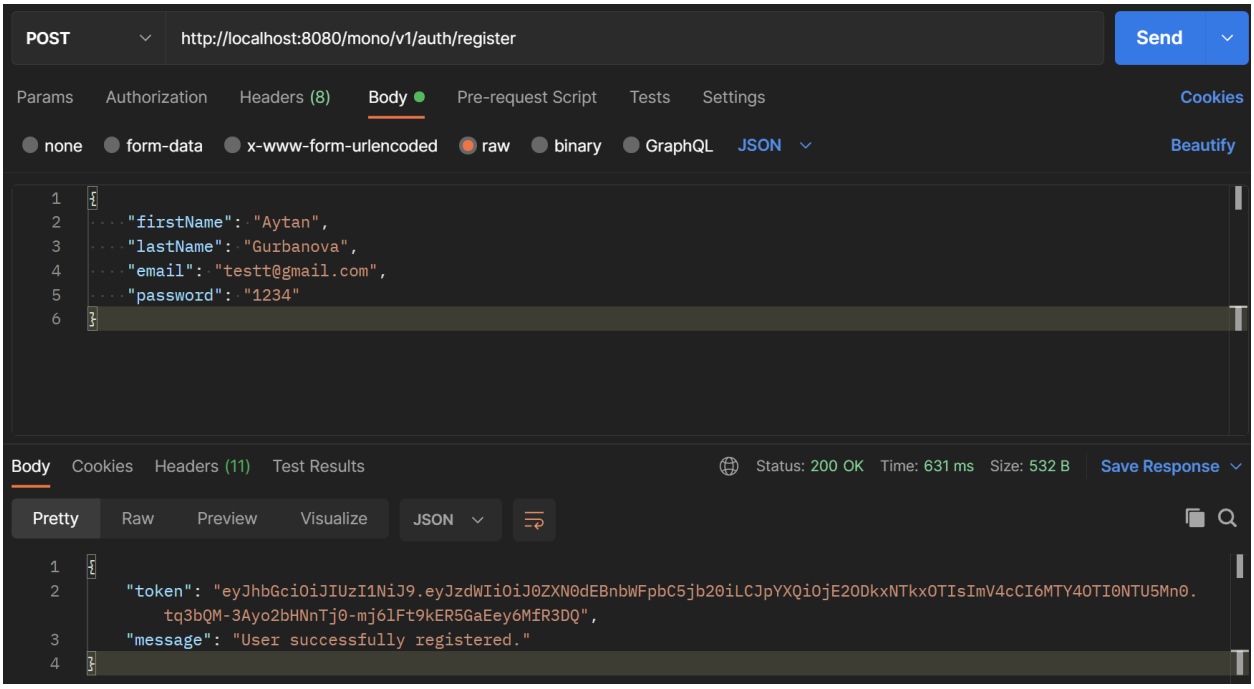
Midterm Progress:

For the original proposal, my purpose for the midterm was to finish analyzing relevant papers for literature review, and read about both architectures separately. Since my proposal was changed from purely theoretical to practical and theoretical, I could not get ready literature review. Even though I have analyzed some relevant papers, my main focus has been on building a simple e-commerce application using a monolithic architecture. I have almost completed this application, with only the unit tests and some refactoring remaining.

For the framework, I have chosen Spring Boot, and I am using Postgres as the database. As mentioned earlier, I have only implemented the user-side of the application to keep things simple. The application is designed as follows:



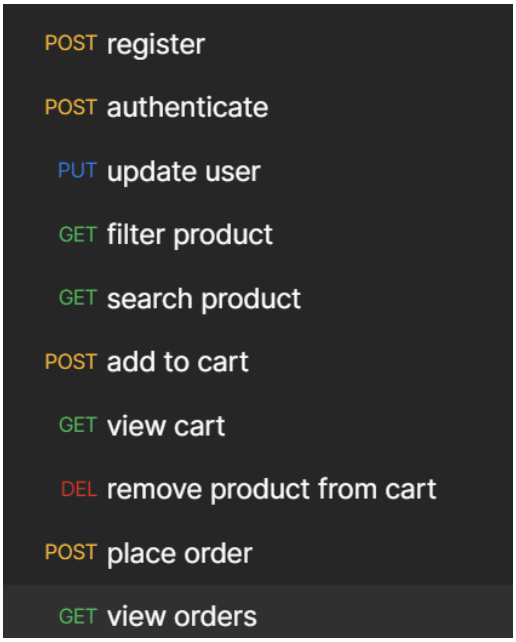
For authentication, a JWT token is required. An example of a user registration request is provided below:



Once the user is created, an authentication request is sent with the email and password, which returns a token for accessing other methods. Currently, the expiration time for the token is set to 24 hours for testing convenience.

Additionally, I have not yet implemented a refresh token method. I may consider creating one after completing the microservices and if I have spare time before the end of the term.

Currently, there are ten services. Instead of going details of each one I have only included screenshot of the list from Postman below:



Currently, there are ten services in the application. Instead of going into the details of each one, I have included a screenshot below which shows the list in Postman.

All of these services are functioning as intended. Once the microservices are completed, I plan to create various test scenarios by combining these services to assess their performance.

You can find the application at the following GitHub repository: https://github.com/ADA-GWU/guidedresearchproject-aytan-gurbanova/tree/08b34bd3830cb20814332905a26017255b67122d/code/monolith/ecomm-mono

Finally, the database tables are designed as following:

A high-level explanation of the tables above:

 Users: This table stores information about users who register and interact with the e-commerce application, including their email, password, name, contact details, role, and account status.

 Categories: This table represents the different categories or types of products available in the e-commerce application.

 Products: This table stores information about the products available in the e-commerce application, including their name, description, price, quantity, brand, associated category, and timestamps for creation and updates.

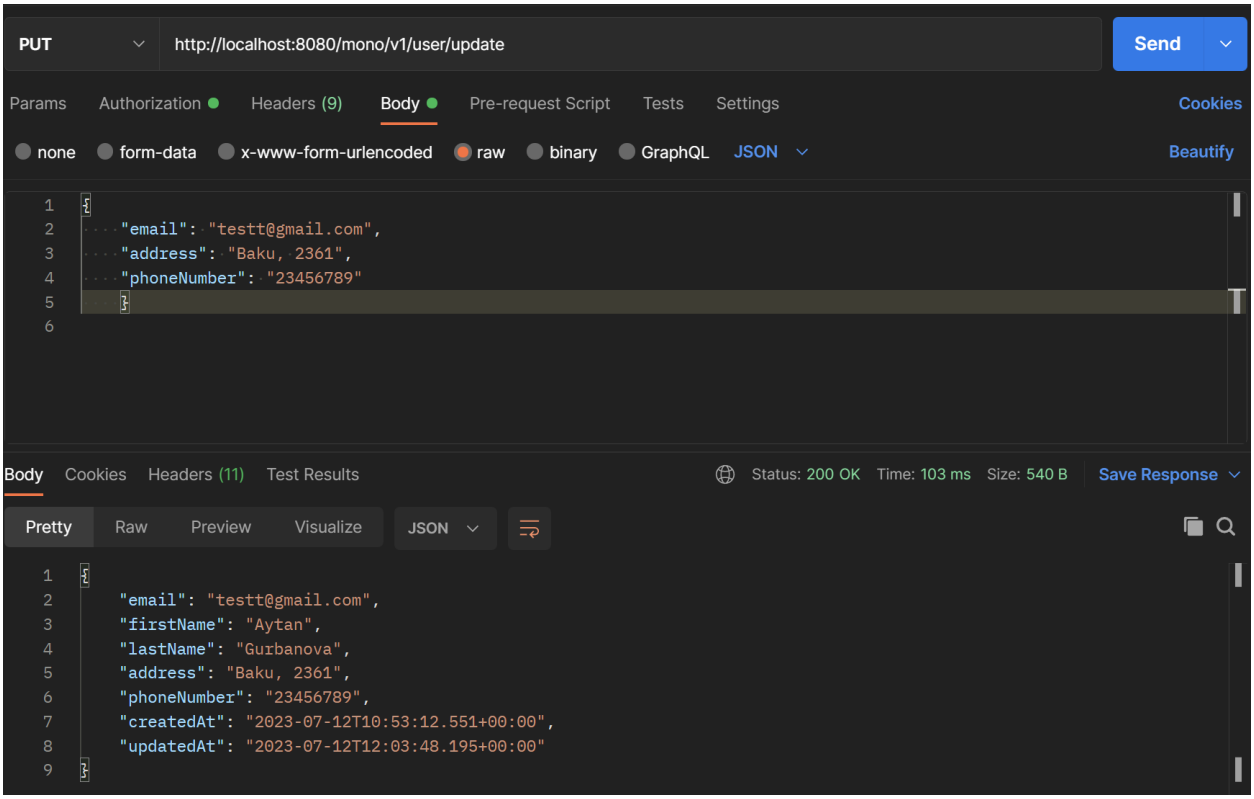
 Carts: This table represents the shopping carts of users, storing information about the products added to the cart, along with the quantity and timestamps. When a new product is added to the cart, it first checks for its existence. If the product already exists in the cart, only the quantity is increased. Otherwise, the product is added as a new record.

 Orders: This table is used to track customer orders, including the user who placed the order, the total order

amount, and timestamps for creation and updates. For the sake of simplicity, the flow ends when an order is

created; there is no payment integration, etc in the application.

In the users table, there are several fields, including some related to Spring Security, such as 'locked' and 'enabled'. The 'user role' field is set to 'USER' by default since the admin-side has not been implemented. Additional user information, such as address and phone number, can be updated by the user at a later stage:



Future Plans:

With just a few refactoring tasks remaining, the development of the monolithic application is almost complete. Therefore, my next goal is to start researching microservices architectures and begin development right away. Since the same services will be utilized, I expect that the development of microservices won't require a significant amount of time. Within a few weeks, I plan to commence deployment and subsequently conduct performance comparisons. I will then share the results in my final report.