Rest API & Microservices

Monolithic – Single System

* User will send request to server and all services are using on TomCat (Single Services)
* Single server and it will talk to DB and back to browser
* Single JAR File that contains everything. Easy to develop, test and deploy.
* Low complexity and developing services is easy.

Disadvantage:

* Tightly coupled: connected with each other. If you only want to work with inventory you cannot. Cannot do changes to one module.
* Hard to scale: If multiple users are accessing and traffic is huge. You need to do load balancing. You need to scale you cannot do on specific component. You have to do on all of them.
* Redirect traffic to different servers but hard to scale and have to deploy entire and manage load balancer.
* Slow performance
* Single point of failure: If system goes down all of it goes down not just shipping.
* Slow continuous development: You cannot design another component until first one is done. CD is not possible.
* Good for smaller application or small environments less developers

Microservices: Split all services into different parts

* Loosely coupled = every component is independent. If there is something not working user can still access other services. Parallel testing and development is also possible
* Agile & flexible: If there is issue on particular module, you can continue working on other
* Independent development and deployment: multiple servers. They talk to each other through API calls
* Fault isolation: easily recognize issues. Can fix only particular instead of all of them
* Mixed technology Stack: You can use multiple technology. Can use JAVA or SQL or other services
  + Disadvantages:  
    Complexity, Consistency (Should get proper response through API services), Automation (more # of components – Infrastructure), Debugging – very tough because there are multiple DB, etc.