1. In last lecture, we saw the issues we can face in microservices and the role of resiliency in dealing those issues.
2. Let’s see a typical use-case that can occur in any microservice network.
   1. We have 3 microservices (accounts, loans, cards).
   2. There are multiple apps invoking our microservices.
   3. App1: For customer Executive App.
      1. Any customer using this app to get accounts, loans and cards details.
      2. This will invoke API on account micro and then will invoke loans and loan invokes cards.
      3. So, they collectively work with each other and return a consolidated response to app one.
   4. App 2: Invoking API on accounts micro only.
   5. App 3: Invoking API on Loans micro only.
   6. App 4: Invoking API on Cards micro only.
   7. Suddenly, Cards DB starts responding slowly so does your cards micro.
   8. App 1 and 4 depend on Cards Micro, so they also face issue.
   9. See, there may be thousands of customers using App 1 and so many requests from App1 so thousands of threads are being created and the resource are being allocated and they keep on waiting on Cards Micro.
   10. This is cascading effect just because one microservice is not working properly.
   11. If we have resilience pattern implemented in our microservice architecture, then only app 1 and 4 will get affected as these app are dependent on Cards micro.
   12. If we have fallback mechanism so return default response from cards micro, then app 2 and 3 will work fine and in worst case customer executive app can’t see cards details.