**Assignment 2**

**Monolithic Architecture**

* In this architecture each component is developed as an independent module, however, they stacked/packaged together and deployed as a monolith.
* Pros:
  + These are easier to develop.
  + These are easier to test by simply launching the application and do end-to-end testing.
  + These are easier to deploy.
  + Easily scalable through horizontal scaling.
* Cons:
  + Huge application size and complexity makes it difficult to make changes to it.
  + These are not update friendly. The whole application has to be redeployed upon each update.
  + These are difficult as each service might have different requirements.
  + As all services are tightly coupled in these, bug in any service can bring down the whole application.

**Microservices Architecture**

* In this the application is divided into smaller, interconnected services called microservices.
* Each microservice can act as an independent application having its own hexagonal architecture.
* Pros:
* It tackles the problem of complexity by decomposing application into a set of manageable services which are much faster to develop, and much easier to understand and maintain.
* It enables each service to be developed independently by a team that is focused on that service.
* It reduces barrier of adopting new technologies since the developers are free to choose whatever technologies make sense for their service and not bounded to the choices made at the start of the project.
* Microservice architecture enables each microservice to be deployed independently. As a result, it makes continuous deployment possible for complex applications.
* Cons:
  + Microservices architecture adding a complexity to the project just by the fact that a microservices application is a [distributed system](http://www.antonkharenko.com/2015/06/notes-on-distributed-vs-non-distributed.html). You need to choose and implement an inter-process communication mechanism based on either messaging or RPC and write code to handle partial failure and take into account other [fallacies of distributed computing](http://www.antonkharenko.com/2015/06/notes-on-fallacies-of-distributed.html).
  + Microservices has the partitioned database architecture. Business transactions that update multiple business entities in a microservices-based application need to update multiple databases owned by different services. Using distributed transactions is usually not an option and you end up having to use an eventual consistency-based approach, which is more challenging for developers.
  + [Testing a microservices](http://martinfowler.com/articles/microservice-testing/) application is also much more complex than in case of monolithic web application. For a similar test for a service, you would need to launch that service and any services that it depends upon (or at least configure stubs for those services).