**Security**:

* The System should be secured using Oauth 2.0 or OpenID.
* Services should be exposed over HTTPS over TLS to prevent any Man in the Middle Attacks.
* While using a underlying DB, injection attacks should be taken care of using Prepared statements and should not expose table column names in the responses.
* http-only header cookies should be used.
* CORS should be enabled for trusted origins/domains.
* *While on Cloud*:
  + Services should be deployed in a virtual private cloud and exposed through a NAT/Bastian instance or Gateway.
  + Proper Access Control Lists should be specified.
  + Services should be deployed behind an API gateway and throttling should be implemented in order to prevent DDOS type of attacks.
  + Instances should be assigned specific roles in order to access specific resources e.g. if an EC2 instances needs to communicate to a RDS instance then EC2 instance should have the specific role.
  + If possible maintain a event store of transactions for future analysis purpose.

**Persistence** :

* The current system relies on a in memory concurrent hash map for persistence which is ephermal which has following problems:
  + For every Restart the Data would get purged.
  + With a surge in usage the memory utilization of application would increase.
  + We can not scale out the application as each instance would be runnig in isolation with its own resources.
  + Acquire optimistic locking on the transactions to maintain consistency.
* *Recommendations*:
  + a RDBMS e.g. MySQL or Oracle.
  + Scale out the DB when usage increases as Master Slaves. (Read/ Write Dbs)
  + Periodic Snapshots for Disaster Recovery
  + *While on Cloud*:
    - Any Managed Service like RDS could be used.
    - Read Replication can be configured.
    - Multi-AZ can be configured for Disaster Recovery.

**Deployment:**

* Repository Management Using GIT/BitBucket.
* Jenkins or any other tool for continous integration/ continous deployment.
* Static Code analyzer like SonarQube, PMD, checkstyle for code quality builds.
* Application should run behind a load balancer and reverse proxy preferably using round robin mechanism.
* Containerization could be done using Docker.
* *While on Cloud*:
  + Expose through API Gateway.
  + Use Elastic Load balancers.
  + EC2 instances or ECS
  + Virtual Private Cloud with proper NACL rules.
  + Deployed in multiple subnets in multiple Availablity Zones.