**Designing the End to end Loan predicting system:**

The aim is to Design a Loan predicting system by collecting the behaviors of the customers from different sources.

**Functional Requirement:**

1. Data upload
2. Data storage
3. Data processing
4. Machine Learning
5. Get Real-time request
6. Report generator.

**Nonfunctional requirement:**

1. Data collection
2. System should be highly avilaable
3. Low latency
4. Scalable
5. Services should be container based

**Extended Requirements:**

The System should be Rest API based.

**System API:**

**Data upload**

1. **Post : localhost:**

{

{

     "hostID"  : "[hostsToConnect]",

    "lastUpdateTime"  : "12312312312",

“Data” : “”

}

**2) Process data:**

Post: localhost:port/api/v1/hostId/process

{

“executionEngine” : <Spark or MR>

“Source” : DB

“outputPath” : CSV file target path

}

**3) Post :**

localhost:port/api/v1/hostId/generateModel

{

PickleFilePath:” “

}

Generate the model using the CSV file and deploy the pickle file.

**4) Post :** localhost:port/api/v1/hostId/getreport

{

Request: “ “

}

This API call will interact with the deployed ML model and returns the result.

**Database Design:**

Since the data is from various sources so it may not be structural so NoSQL data bases like Cassandra or HBase is need to store the data.

**Basic system design architecture:**

DB Serv

SparkServ

MLServ

Load Balancer

App Server

Report Serv

Kafka Serv

Casandra

**We are going to run different modules using Micro service architecture.**

**Load Balancer:**

User interacts with diffreent application servers. So Load balancer uses the mechanism to distribute the load across different servers. Since it is microservie application with high avilability so there could be multiple instances of different servers running.

**Kafk Server :**

Handles the steamng application. Clients write data to topics and consumers consume them and sends to ML serv to get the report.

**Spark servers:**

Work on batch mode. It reads the data from the database and process them and finally stores them to HDFS location.

**ML server:**

Continuously pools the target and gets the requests and runs the model and pushed the output to the reporting server.

**Reporting servers:** Displays the messages

**DB server:** Used as intermediate storage an historical data storage.

**Caching** is used with ML server to achieve the real time results.

**NonFunctional system design:**

High availabilty is achieved using Micorservice architecture by spanning multiple instances of a service.

Since the services run on cloud so scalability can be achieved as per requiremnt.

Fault tolerance and ressiliency: DB replication,message replication and report output has multiple copies.

The servers can be easily packaged and deployed using docker.

**Technology suggestion:**

1. Cloud based system like AWS,GCP or Orale cloud
2. Docker for easy deployment
3. Bigdata especially Spark
4. Kafka for streaming
5. Cassandra for storing.