Loan Onboarding System

Problem Statement 2

As an experienced architect of a software development team your problem statement is -

1.To design the next generation Loan Onboarding System.

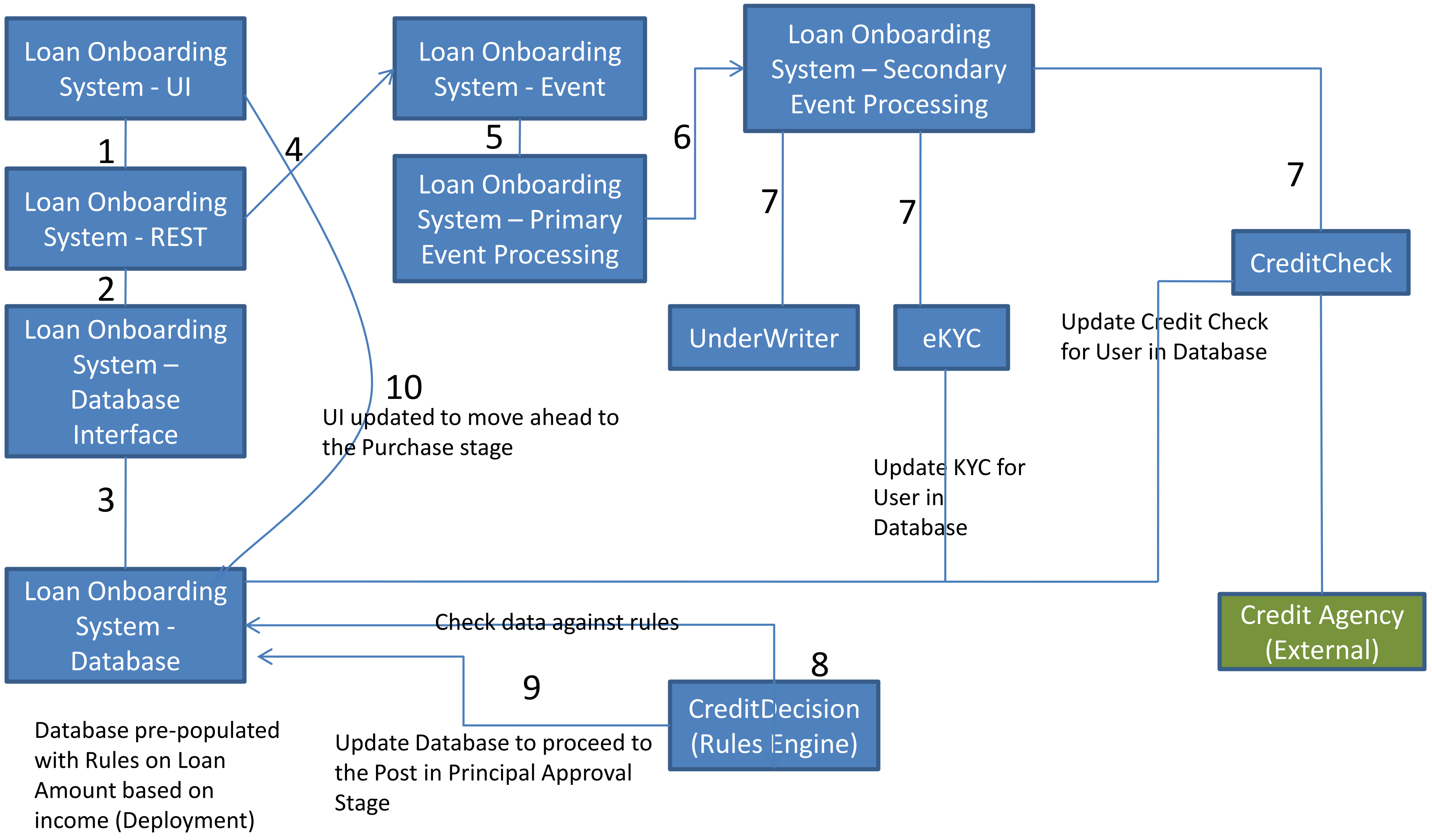
2.A Digital LMS (Mobile / Laptop Loans) typically has following stages -

- •Loan Application Entry
- •Workflow
    - •eKYC Checks
    - •Bureau (Credit Score) Hits
    - •Credit Decision (Straight Through Approved & Declined or Queued for Underwriter) via. a Business Rules Engine run over the gathered (KYC & Bureau) and collected (Data Entry) data, BRE
- •Post In Principal Loan Approval (Straight Through or Manually Approved)
    - •Scheme and Product Selection (10% off, Macbook Pro 15 inch 2016 model, 16GB, 256GB)
    - •Banking and References (Account for the advance and subsequent EMIs/AutoDebit/ECS)
    - •OEM Serial Number Validation
    - •Cross Selling (Insurance, etc.)
    - •Delivery Order (For a Dealer like Chroma, Flipkart, Amazon, etc). Proof of bank's or NBFC's nod to loan approval and the amount.
    - •Invoice Generation (Bill for the customer)
    - •Disbursal

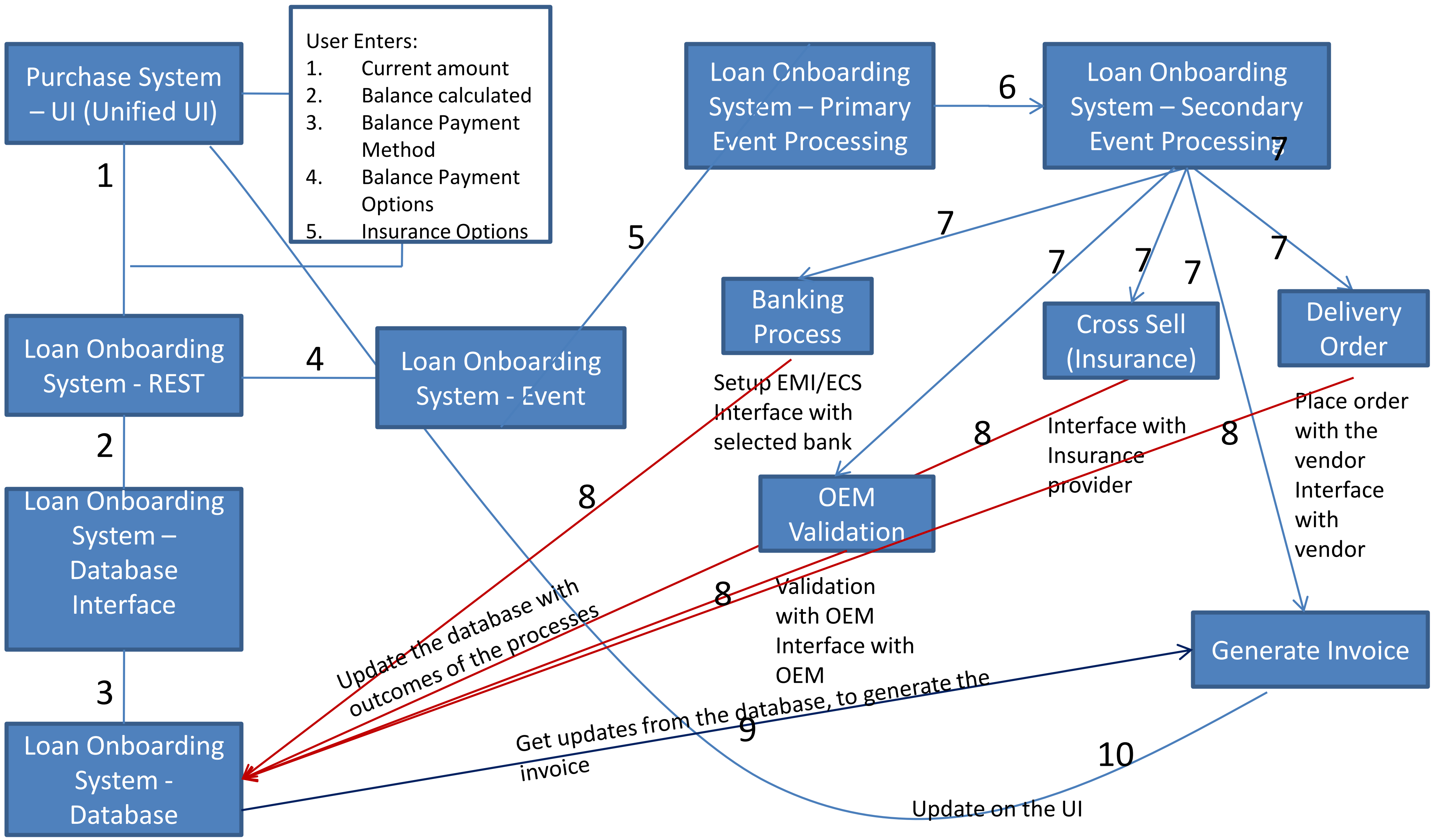
3.The existing system is very crude and non-flexible -

- •The entire workflow from data entry to disbursal is sequential and hard coded.
- •Code changes are required to change the workflow execution order.
- •The existing code base also doesn't fare well -
    - •No design patterns used, lots of if-elseif-else loops for different customer and product combinations, cyclomatic complexity is through the roof.
    - •Monolith services with obscenely big methods.
    - •All the database queries are slow, a NoSQL document based database is used.
    - •Clunky REST APIs.
    - •Loans are first class citizens.
    - •No domains like customer, loan app, product, etc.
    - •Painful CI-CD for clunky services, a lot of circular dependency.
    - •Synchronous communication between services.
    - •Data polling based websites.
    - •Lack of runtime static data changes.
    - •There are many other nuances, effectively no multitenancy in terms of customers and products.
- •The new design should address all the above concerns.
- •Demonstrate the new design via. Sequence, Data Flow, State, Deployment and Use Case Diagrams. Also provide a high level services break up, their dependencies and the choice of database, etc. for the next generation Dig

Loan Onboarding System – Loan Entry and Workflow


Loan Onboarding System - UI
1
Loan Onboarding System - REST
2
Loan Onboarding System – Database Interface
3
Loan Onboarding System - Database
4
Loan Onboarding System - Event
5
Loan Onboarding System – Primary Event Processing
6
Loan Onboarding System – Secondary Event Processing
7
7
7
UnderWriter
eKYC
7
CreditCheck
Update Credit Check for User in Database
Credit Agency (External)
10
UI updated to move ahead to the Purchase stage
Update KYC for User in Database
Check data against rules
8
9
CreditDecision (Rules Engine)
Update Database to proceed to the Post in Principal Approval Stage
Database pre-populated with Rules on Loan Amount based on income (Deployment)

Loan Onboarding System – Post Approval – Purchase Stage



Purchase System – UI (Unified UI)
User Enters:
1. Current amount
2. Balance calculated
3. Balance Payment Method
4. Balance Payment Options
5. Insurance Options
Loan Onboarding System – Primary Event Processing
6
Loan Onboarding System – Secondary Event Processing
1
5
7
7
7
7
7
7
Loan Onboarding System - REST
4
Loan Onboarding System - Event
Banking Process
Cross Sell (Insurance)
Delivery Order
Setup EMI/ECS Interface with selected bank
Interface with Insurance provider
Place order with the vendor Interface with vendor
2
8
8
8
OEM Validation
Loan Onboarding System – Database Interface
8
Validation with OEM Interface with OEM
Update the database with outcomes of the processes
3
Generate Invoice
Get updates from the database, to generate the invoice
9
10
Loan Onboarding System - Database
Update on the UI

Loan Onboarding System – Loan Application Entry and Workflow Modules

Loan Onboarding System UI

Customer Data

A. Customer Information:

1. Name
2. Last Name
3. Address
4. Mobile Number
5. Email
6. PAN / UUID
7. Income
8. Domain in case of Enterprise User

B. Product Information

1. Name
2. Model Name
3. Serial Number
4. Uuid (System)

C. Payment Options:

1. Downpayment
2. BalanceMethod
3. BalancePaymentDetails

Loan Onboarding System REST API

REST

1. url: https://url:port/restidentifier/LoanApplication
2. Create Application (POST)
3. Get Application Details (GET – using id of Loan Application)
4. Update Application (PUT/PATCH – using id of Loan Application)
5. Delete / Cancel Application (DELETE)

Loan Onboarding System Event Processing

Event Processing

1. Primary Event Queue Model (JMS)
2. Secondary Event Queue Model (Akka Message Delivery)
3. Message Consumers – eKYC, UnderWriter, Credit Check (interfacing with external agency)
4. Each of the consumer updates database
5. Credit Decision – Rules based Engine – decision based on data from event consumers and rules
6. Rules prepopulated in the database

Loan Application Entry – REST API

- Create Loan Application
  - POST on https://uri:port/restidentifier/LoanApplication
  - Integration with OAuth2 Provider for Authorization (assumption is that the User Authentication is done when logging in)
  - Audit to record the login/requestor
  - Using POJO objects to accept input to the REST Handler making it easier for data verification
  - Input data validation
  - Update the data in the database
  - Use Event mechanism to notify other modules / processes of new Application to trigger further processing

Loan Application Entry – REST API

- Update Loan Application
  - PATCH / PUT on https://uri:port/restidentifier/LoanApplication/{id}
  - Integration with OAuth2 Provider for Authorization (assumption is that the User Authentication is done when logging in)
  - Audit to record the login/requestor
  - Using POJO objects to accept input to the REST Handler making it easier for data verification
  - Input data validation
  - Update the data in the database
  - Use Event mechanism to notify other modules / processes of application to trigger further processing

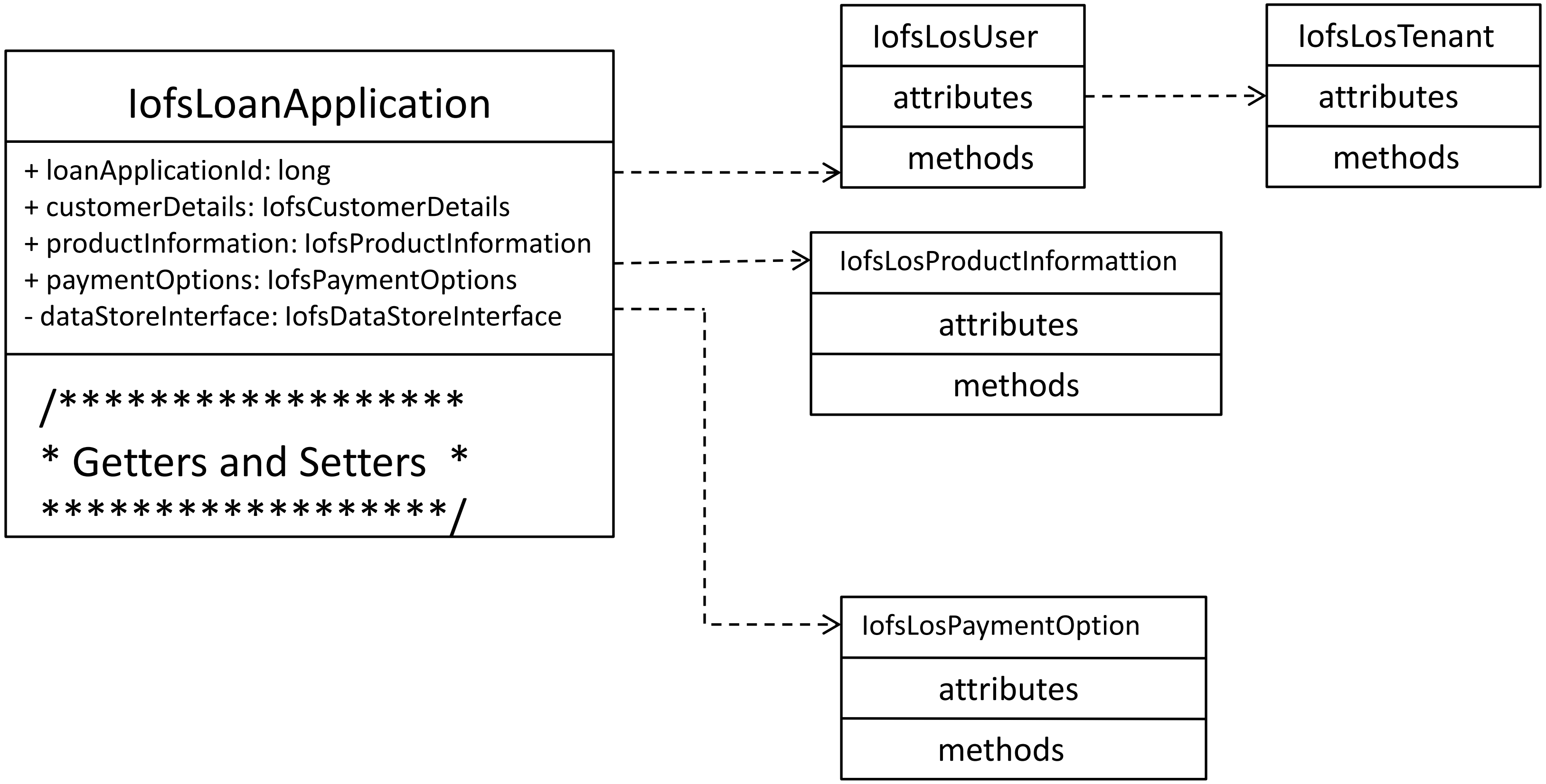
Loan Application Entry – REST API

- Get Loan Application
  - GET on https://uri:port/restidentifier/LoanApplication/{id}
  - Integration with OAuth2 Provider for Authorization (assumption is that the User Authentication is done when logging in)
  - Audit to record the login/requestor
  - Retrieve Data from the database
  - Return the Data in JSON format

Loan Application Entry – REST API

- Delete Loan Application
  - Delete on https://uri:port/restidentifier/LoanApplication/{id}
  - Integration with OAuth2 Provider for Authorization (assumption is that the User Authentication is done when logging in)
  - Audit to record the login/requestor
  - Input data validation
  - Delete data from the database

Loan Application Entry – Loan Application Object


IofsLoanApplication
+ loanApplicationId: long
+ customerDetails: IofsCustomerDetails
+ productInformation: IofsProductInformation
+ paymentOptions: IofsPaymentOptions
- dataStoreInterface: IofsDataStoreInterface
/*****************
* Getters and Setters *
*****************/
IofsLosUser
attributes
methods
IofsLosTenant
attributes
methods
IofsLosProductInformattion
attributes
methods
IofsLosPaymentOption
attributes
methods

Loan Application Entry – User Object, Tenant Object, Payment Options Object

IofsLosUser

+ name: String
+ lastName: String
+ address: IofsAddress
+ phoneNumber: IofsPhoneNumber
+ email: IofsEmail
+ PAN: String
+ UUID: Integer
+ income: Integer
+ Domain (Tenant) Name in case of
  Enterprise User

Getters and Setters

IofsLosProductInformation

+ name: String
+ model: String
+ serialNumber: String
+ vendorName: String
+ UUID (vendor): Integer

Getters and Setters

IofsLosPaymentOptions

+ totalAmount: Integer
+ advanceAmount: Integer
+ balanceAmount: Integer
+ balanceMethod: String
+ balanceDetails: Integer

Getters and Setters

IofsLosTenant

+ name: String
+ address: IofsAddress

Getters and Setters

Loan Onboarding System – Modules

- Loan Onboarding System – REST API (Web Service)
- Loan Onboarding System – Datastore Interface (common Library)
- Loan Onboarding System – Event (common Library)
- Loan Onboarding System – Primary Event Processing – JMS based(Background Process / Web Service)
- Loan Onboarding System – Secondary Event Processing – Akka Message based(Background Process / Web Service)
- Banking Process (Background Process / Web Service)
- OEM Validation (Background Process / Web Service)
- Cross Sell (Background Process / Web Service)
- Delivery Order (Background Process / Web Service)
- Invoice Generation (Background Process / Web Service)
- Credit Check(Background Process / Web Service)
- Credit Decision (Background Process / Web Service)
- eKYC (Background Process / Web Service)
- Common Library – event interface, common interfaces, abstract classes that will be used by the background process or web services
- Logging will be implemented via the slf4j logging framework with default implementation as java.util.logging

Loan Onboarding System - Database

- While SQL works well with Schema oriented data and NoSQL works better with unstructured and un-related data, this solution will propose to have a mix of both
- The loan application, user data will be stored and accessed via SQL, the product data from different vendors will be stored in NoSQL
- For deploy time static data and runtime static data as well as configuration of the services in the system, a zookeeper instance will be used
- The product data from different vendors if fairly static in nature may also be stored and accessed via zookeeper instance

Loan Onboarding System – Some more…

- Performance testing of the services
- Multiple instances of the services deployed for HA

Deployment

- Modules will be designed as Microservices
- Deployment will be done using container based technology such as **Docker**, Kubernetes, etc.
- Each module will be built using the ant or the gradle system
- Devops: CI-CD based pipeline which covers Jenkins, Ansible script framework to build the modules and then use a deployment tool (similar to **Urban Code**, DeployBot, etc.)
- Each module will have the ansible scripts which will be invoked by the Jenkins integration
- Each module will also have components in the code deployment tool chosen
- For creating the image, a base image of the server shall be created, which will be used by the modules to update their respective functionality
- Integration of the CI-CD pipeline with the test suite (unit tests and integration tests, as well as test automation)