1. How long did you spend working on the problem? What did you find to be the most difficult part?

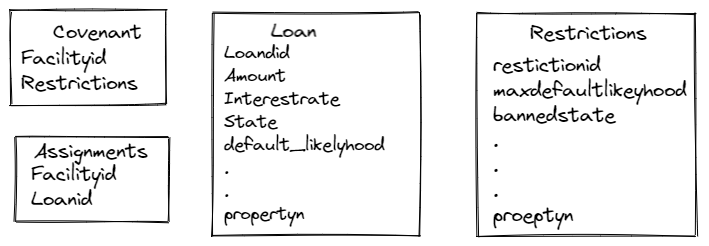
It took 4 hours to complete. I guess I missed the requirement to allocate loans to cheapest facility. Spent some time debugging and then adding the logic so it took some time.

1. How would you modify your data model or code to account for an eventual introduction of new, as-of-yet unknown types of covenants, beyond just maximum default likelihood and state restrictions?

Although I used Pandas here and not OOD how ever Ill explain form the OOD side of it to better understand.

A covenant object should have Facilityid and restrictionID that way if there are similar restrictions applied across Facilities. Here we can assume a restriction logic is tied to the restriction id.

Data model would look like below



A method will take the covenant object and loan object and apply the rules corresponding to the restrictionid

assignmentObject loanAssignments( Covanentobject, loanObject)

{

If ( Covanentobject.restrictions.restrictionid == 1)

{

If( (Covanentobject.restrictions.maxdefaultliklyhood > loanobject.default\_likyhood) && loanobject.state != Covanentobject.restrictions.bannedState

Do some thing

}

If ( Covanentobject.restrictions.restrictionid == 2)

{

If( (Covanentobject.restrictions.propertyn != loanobject.propertyn) && loanobject.state != Covanentobject.restrictions.bannedState

Do some thing

}

Return assignmentobject

}

The advantage with this design is that the loan assignments method will be the central logic to all the restrictions that can be there allowing one place to look for and manage the rules. Alternatively this method can interface with roles engine and achieve the same.

When a new covenant is introduced, this function will need change to incorporate the rule.

Loan object and restriction object should be coordinated so that they have the matching properties to apply the rules.

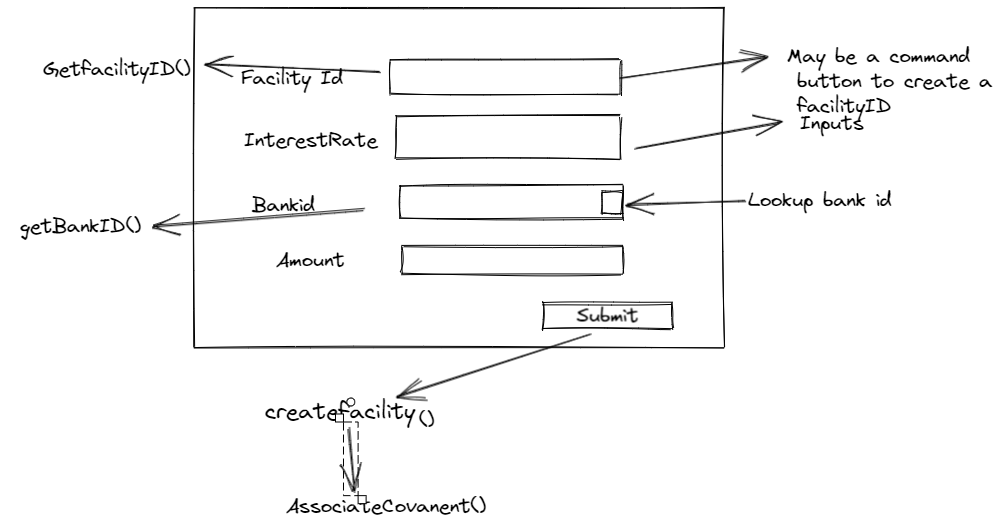
Here I’m excluding the complexities/approach required in terms of transformation form CSV to objects. That will be a separate discussion as there are many ways to achieve it

In terms of testing any change to the loanAssignments method, will need extensive testing to make sure previously coded restrictions are intact. I’m assuming there is automation to unit test nothing else has changed/broken.

The only goal of this approach is being able to add new covenants with min efforts

1. How would you architect your solution as a production service wherein new facilities can be introduced at arbitrary points in time. Assume these facilities become available by the finance team emailing your team and describing the addition with a new set of CSVs.

Lets say we can come up with a UI that accepts the information as Below

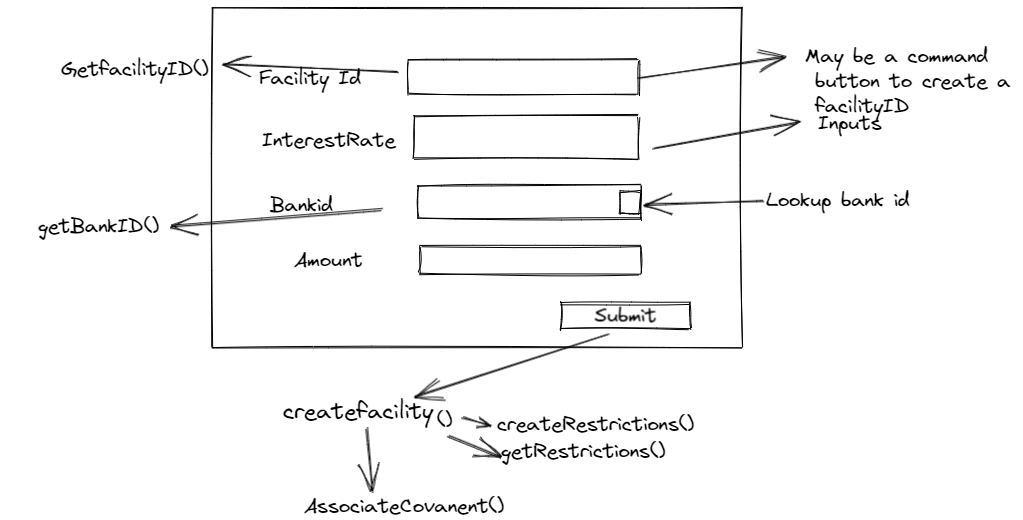


We will need a service that will generate unique facility ID called getFacilityid

We need service that will lookup bank info called getBankid()

When the information is submitted, we need a service to update all the facility related info in the data model, and then assign relevant restrictions to the facility. If we go with the way data model is identified in question 2 then we need additional service that will update the restrictions data model

In this case the design will look like



CreateRestrictions() will add new entry in the restrictions DM, AssociateCovanaet(0 will update coavanent DM and Createfacility() will create new entry into the facilityDM as below.



In this solution I’m not designing anything in terms of Data source ( DB, CSV,XML) The data model is on the Back-end API side and there should be some other systems and logic to update the information on to the data Source. The approach will differ based on that.

1. Your solution most likely simulates the streaming process by directly calling a method in your code to process the loans inside of a for loop. What would a REST API look like for this same service? Stakeholders using the API will need, at a minimum, to be able to request a loan be assigned to a facility, and read the funding status of a loan, as well as query the capacities remaining in facilities

To request a loan to be assigned to a facility we can use the same API defined in the answer to question 2 ( loanAssignments)

The requirement “read the funding status of a loan” I understand it as ability to see if a loan is assigned to a facility or not. In this case we will need an API that takes loanID and returns true or false This API will use assignments DM as described in answer to question 2

Boolean getFundingStatus( loanId)

{

If LoanId is in assignments, then return true

}

For the requirement “capacities remaining in facilities” Need an API as follows

Double getFacilityCapacity( facilityid)

{

facilityAmount = Facility.getAmount()

LoanIDlist = assignments.getLoanid(facilityid);

TotalLoandedAmt = 0

For LoanIDList

{

loanAMT = Laon.getAmount( loaned)

TotalLoandedAmt = TotalLoandedAmt+ loanAMT

}

Return( facilityAmount – TotalLoandedAmt)

}

This design assumes the Data Model as in questions 2 and the similar structure as in CSV for those that are not identified in the question2

The API also assume that there is another API that takes care of unassigning a loan if it doesn’t go thro and the facility is release of that loanID.

1. How might you improve your assignment algorithm if you were permitted to assign loans in batch rather than streaming? We are not looking for code here, but pseudo code or description of a revised algorithm appreciated.

In the absence for the reason for the need to batch the loan assignment, I will make some assumptions here to justify the design. Batch on huge volume of data set will be time consuming and there is a need-to-know which data set is already processed.

One way to do it is adding an additional property in the Loan data model to identify if it was assigned or not. ( Refer to the DM in question 2)

Option1 : This approach holds good if the goal is to have a background process the loan at a regular interval of time. Here the assumption is the data source is streamed but the processing of those data is staggered.

BatchProcessing()

{

Loanobjects = getAllLoanNotProcessed()

Covenants = getCovanents()

Loop Through

{

Assignments = loanAssignments ( covanent, Loan)

updateLoanProcessedfield( assignments.getloanid()) This method updates the field in Loan data model to indicate this loan was assigned.

}

}

Option2: This option can also be run at certain interval, The additional step of getting loan List from 2 different sources and separating them may add addition time and space complexity.

BatchProcessing()

{

LoanidList = assignments.getLoanids()

Loanidlist2 = getallLoansFronLoansDS()

For Loanidlist2 not in LoanidList

Assignments = loanAssignments ( covanent, Loan)

}

1. Discuss your solution’s runtime complexity.

In the code there are 2 areas that require for loop.

1. Loan Assignment: complexity is O(n) where N is the length of Loan Data source

I have not considered the complexity of using Loc which is Pandas operations. In my research it looks like the complexity is O(n) where n is the row count of the dataframe used.

In the code Loc is used to apply the Covenants hence the dataFrame complexity will be O(K) where K is the total length of Covenants. Assuming this size is negligible compared to the total number of loans I will ignore adding this complexity.

1. Yield Calculation: Complexity is O(n) time O(m) where N is the length of Assignments ( facilities) DS and m number of loans assigned to a Facility. This is variable as one facility might get 10000 assignments, and some might have 1 assignment over all I would say the if M>n ( if M is significantly larger than n) then the complexity will be O(M)