**Affirm Loan Processing**

**Details:**

**Git Hub** : <https://github.com/bvarla/AffirmLoanProcess>

**Name : Bharat Varla**

**Project Details:**

* 1. **Download SpringBoot project from Github**
  2. **Once ran successfully running java project it runs locally on port 8080 exposing the rest URL -** [**http://localhost:8080/api/v1/assignLoan**](http://localhost:8080/api/v1/assignLoan)

**3. As per time constraints Util class methods are written in service class and Unit tests are skipped.**

**4.One the get URL is hit the output csv files are generated in resource folder as seen in the Image.**

**Graphical user interface, application

Description automatically generated with medium confidence**

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

It took me around three hours. Most difficult part was to understand the corner cases and testing the application on various scenarios (The input provided does not cover some covenant conditions which was challenging).

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?**

According to the SOLID design principle, make fine grained interfaces that are rules specific, and which would adopt dynamic rule changes without breaking the code allowing modification of rules.

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.**

Right now, the data is in csv format to make it production following changes need to done

* Upload CSV data to SQL DB (ACID Compliant)
* Whenever new data CSV is provided by business update the DB with scripts
* Expose End point to Upload the CSV by business to make it dynamic and speed up the process.

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.**

* **PostCall – Request a loan to be processed**
* **GetCall – Funding status of loan**
* **GetCall – Details of remaining capacities in facilities.**

The service that business can call post Method which would process the requested loans and return back the status and other information as needed. Middle tier would take care of backend Data and process the business logic. Database should be ACID compliant to limit the facility to one processing at a time (Maintaining transactional data) rather than parallel to avoiddiscrepancy

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.**

When the loans are processed as batch then we have flexibility of selecting the facility considering the covenant restrictions & loan amounts limits for each loan. One thing we can do is to move the loan from one facility to another facility. Steps flows:

* 1. When a loan is being processed, look for the highest available amount value facility. Example facility\_id: 1
  2. Then loop through each of the remaining facilities that are not fully occupied. Note down those facilities. (facility\_id:2, facility\_id:3)
  3. Now get the loans that are available in highest amount value facility (facility\_id:1) that can fit into other facilities (facilitiy\_id:1, facility\_id:2).
  4. This way free up the available amount in the first facility (facility\_id:1)
  5. When there is a new loan with highest amount needs to be processed then it can now pick up the facility\_id:1 to process the loan.

1. **Discuss your solution’s runtime complexity.**

N – Number of loans to be processed

M – Facilities

Time Complexity: O(N\*M)