# Assessment Parameters

* Complete flow of the application  with validation and exception handling -70%
* Comments/best practice, coding standards – 10%
* Execution of the application (Output) – 20%
* ScreenShot should be submitted along with the solution
* The solution(Project) created by the trainee should have the name like AppName\_Empid Ex:ABCCorp\_675467
* Code with compilation errors will not be considered for evaluation

# XYZ Banking Application

**Problem Statement**: An XYZ bank requires an application which allows its employees to enter the demand draft details provided by the customer. (Assume that the details are printed on the demand draft. Also, the scope of the application is not extended to search the branch code and print the branch name)

***SCHEMA***

CREATE TABLE demand\_draft(transaction\_id NUMBER, customer\_name VARCHAR2(20),in\_favor\_of VARCHAR2(20), phone\_number VARCHAR2(10), date\_of\_transaction DATE, dd\_amount NUMBER, dd\_commission NUMBER, dd\_description VARCHAR2(50));

CREATE SEQUENCE Transaction\_Id\_Seq

START WITH 10001;

Display menu to the employee should be as follows

* 1. Enter Demand Draft Details
  2. Exit

1. **Enter Demand Draft Details [55 Marks]**

When the employee selects this option, it allow the employee to enter the demand draft details as shown in the below sample screen. When all the details entered are added successfully to the above table, it should print the message “**Your Demand Draft request has been successfully registered along with the <transaction ID>**”.

**Note**:

* + 1. Transaction ID should be generated using sequence
    2. DD commission should be calculated according to the range provided in the below table
    3. Validation should be taken care for all input fields

|  |  |
| --- | --- |
| **Amount (in Rs)** | **Commission (in Rs)** |
| Upto 5,000 | 10 |
| 5,001 – 10,000 | 41 |
| 10,001 – 1,00,000 | 51 |
| 1,00,001 – 5,00,000 | 306 |

**Sample Screen**

**Enter the name of the customer** : John

**Enter customer phone number**: 9768587350

**In favor of**: Capgemini

**Enter Demand Draft amount (in Rs)**: 45000

**Enter Remarks**: DD taken in favor of Capgemini

The below message should be generated

**Your Demand Draft request has been successfully registered along with the 10001**

* **Marks Distribution:**

|  |  |
| --- | --- |
| Inserting the details with proper use of service + DAO layers + Bean | 30 |
| Displaying Transaction ID | 8 |
| Validation and Error messages | 5 |
| Calculating commission | 7 |
| Comments and best practices/standards | 5 |
| Proper Exception handling | 5 |

1. Exit

**[5 Marks]**

1. Each of the DAO operation should be logged into a file using Log4J

**[5 Marks]**

1. Write JUNIT test cases for the methods in the dao class

**Classes to be created**

**com.capgemini.bank.ui**

Class Client {

public static void main (

// User Interface which display the menu and accept the Input from user

// Create object for service and execute the respective methods}

**com.capgemini.bank.bean**

Class DemandDraft { // Bean class for Demand Draft with all required properties}

**com.capgemini.bank.service** // Service utility classes and Interfaces

Interface IDemandDraftService {

int addDemandDraftDetails (DemandDraft demandDraft);

DemandDraft getDemandDraftDetails (int transactionId);

}

Class DemandDraftService implements IDemandDraftService {

int addDemandDraftDetails (DemandDraft demandDraft){ . . . . . }

DemandDraft getDemandDraftDetails (int transactionId){ . . . . . }

}

**com.capgemini.bank.dao //** Class to perform Data Accessing logic

Interface IDemandDraftDAO{

int addDemandDraftDetails (DemandDraft demandDraft);

DemandDraft getDemandDraftDetails (int transactionId);

}

Class DemandDraftDAO implements IDemandDraftDAO{

int addDemandDraftDetails (DemandDraft demandDraft){ . . . . . }

DemandDraft getDemandDraftDetails (int transactionId){ . . . . . }

}

Add appropriate user defined exception classes and any other supporting classes required.