**Requirement:**

Write a production ready Java application based on Spring framework.

This can be command line, multithreaded, Web based (with or without UI), cloud ready - choice is yours.

Pick a solution you are most comfortable to implement, solve.

Don’t overcomplicate and have a very clear end goal.

We will review together, the idea being we will pair and understand the intent. Bring notes if you like.

Develop a backend services for **i-Bank Application** with below features

1. Add/update/delete beneficiary accounts with validation
2. Account summary ( transactions list)
3. Update personal details with validation
4. Deposit/withdrawal features.
5. Junit test

**Table Structure**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table** | **Accounts** |  |  |  |
| **ACCOUNT\_ID** | **ACCOUNT\_NAME** | **PHONE** | **EMAIL** | **STATUS** |
| 1 | Vinod | 12345 | [vinod@ibank.com](mailto:vinod@ibank.com) | Active |
| 2 | Hafeez | 23456 | [hafeez@ibank.com](mailto:hafeez@ibank.com) | Active |
| 3 | Gobi | 6543 | [gobi@ibank.com](mailto:gobi@ibank.com) | Active |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table** | **Beneficiary** |  |  |  |
| **ACCOUNT\_ID** | **BENE\_ACCOUNT\_ID** | **BENE\_IFSCCODE** | **BENE\_NAME** | **STATUS** |
| 1 | 123094355 | IFSC00001 | Sample 1 | Active |
| 1 | 444535466 | IFSC00002 | Sample 2 | Inactive |
| 2 | 75443554 | ISFC0003 | Sample 3 | Active |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Table** | **Transactions** |  |  |  |  |  |
| **ID** | **Account\_Id** | **Date** | **Type** | **Amount** | **Status** | **Remarks** |
| 1 | 1 | 18-Sep-22 | Credit | 1000 | Success |  |
| 2 | 1 | 18-Sep-22 | Debit | 500 | Failure |  |
| 3 | 2 | 19-Sep-22 | Credit | 5000 | Success |  |
| 4 | 3 | 19-Sep-22 | Debit | 1000 | Success |  |

|  |  |
| --- | --- |
| **Table** | **Account\_Balance** |
| **Account\_Id** | **balance** |
| 1 | 10000 |
| 2 | 5000 |
| 3 | 3000 |

**Submitting:**

The task should be completed to a standard that you consider production ready. Treat it as you would something that you plan to issue a pull request for. You can use any JVM language: Java, Scala, Kotlin, Groovy.

We recommend that you use the language that will show your work in the best light. If you are more familiar with Java but decide to write in Scala, you will not be graded "on a curve"; we will be looking for high quality idiomatic code regardless.

The test is quite open-ended and there is no one "correct solution". While you are free to put as much time into this as you want or as little, a decent solution should be achievable in a few hours.

The completed project should include a README which should include instructions on how to run and test.

Clearly document all assumptions you have made.

* The project should be committed to github/bitbucket and share the link.
* Please do not use any string with BNP Paribas in your code.
* Do not email the zip file itself as it may be blocked by security filters

**Extra credit:**

1. You can also print additional information if you find it useful, but don't make noisy output.
2. If something is not clear - make sensible assumptions, justify them and implement the application logic according to them.
3. Write comments only if they are necessary. Don’t clutter code.