System Requirements Specification

Index

For

Loan Application

Version 1.0

Table of Contents

LOAN APPLICATION

System Requirements Specification

BACKEND-SPRING DATA RESTFUL APPLICATION

1 PROJECT ABSTRACT

The **Loan Application** is implemented using Spring Data with a MySQL database. The application aims to provide a comprehensive platform for managing and applying for all loans across different banks.

Following is the requirement specifications:

	Loan Application
Modules	
1	Bank
2	Loan
Bank Module Functionalities	
1	List all banks
2	Get bank by id
3	Create bank
4	Update bank by id
5	Delete bank by id
6	List all banks who gives interest rate below 10.0% **(must use dynamic method)**

Loan Module Functionalities	
1	Create a loan **(must be transactional)**
2	Update loan status **(must be transactional)**
3	List all loans **(must return all loans by applicant names in ascending order and that also in pages)**
4	Get list of all loans by status **(must use custom query)**

2 Assumptions, Dependencies, Risks / Constraints

2.1 LOAN CONSTRAINTS

- When fetching a loan by ID, if the loan ID does not exist, the service method should throw a NotFoundException with “Loan not found.” message.
- When updating a loan, if the loan ID does not exist, the service method should throw a NotFoundException with “Loan not found.” message.

2.2 BANK CONSTRAINTS

- When fetching a bank by ID, if the bank ID does not exist, the service method should throw a NotFoundException with “Bank not found.” message.
- When updating a bank, if the bank ID does not exist, the service method should throw a NotFoundException with “Bank not found.” message.
- When deleting a bank by ID, if the bank ID does not exist, the service method should throw a NotFoundException with “Bank not found.” message.

Common Constraints

- For all rest endpoints receiving @RequestBody, validation check must be done and must throw custom exception if data is invalid
- All the business validations must be implemented in dto classes only.
- All the database operations must be implemented on entity object only
- Do not change, add, remove any existing methods in service layer
- In Repository interfaces, custom methods can be added as per requirements.
- All RestEndpoint methods and Exception Handlers must return data wrapped in **ResponseEntity**

3 Business Validations

Bank

- Name should not be blank.
- Loan type should not be null.
- Interest rate should not be null.

Loan

- Applicant name should not be blank.
- Bank info should not be null and valid.
- Status should not be blank.

4 REST ENDPOINTS

Rest End-points to be exposed in the controller along with method details for the same to be created

4.1 BANKCONTROLLER

URL Exposed		Purpose
1. /api/banks		Fetches all the banks
Http Method	GET	
Parameter	-	
Return	List<BankDTO>	
2. /api/banks/{id}		Get a bank by id
Http Method	GET	
Parameter 1	Long (id)	
Return	BankDTO	
3. /api/banks		Create a new bank
Http Method	POST **The bank data to be created must be received in the controller using @RequestBody.**	
Parameter	-	
Return	BankDTO	
4. /api/banks/{id}		Updates existing bank by id
Http Method	PUT **The bank data to be updated must be received in the controller using @RequestBody.**	
Parameter 1	Long (id)	
Return	BankDTO	

5. /api/banks/{id}		Deletes a bank by id
Http Method	DELETE	
Parameter 1	Long (id)	
Return	-	

6. /api/banks/interest-rate-below/{interestRate}			Fetches all banks having interest rate below given interest rate
	Http Method	GET	
	Parameter 1	Double (interestRate)	
	Return	List<BankDTO>	

4.2 LoanController

URL Exposed			Purpose
1. /api/loans			Creates a loan
	Http Method	POST **The loan data to be created must be received in the controller using @RequestBody.**	
	Parameter	-	
	Return	LoanDTO	
2. /api/loan/{id}/status			Gets the status of loan by id
	Http Method	GET	
	Parameter 1	Long (id)	
	Return	LoanDTO	
3. /api/loan			Gets all loans
	Http Method	GET	
	Parameter	-	
	Return	List<LoanDTO>	
4. /api/loans/status			Fetches the list of all loan with given status
	Http Method	GET	
	Parameter	status	
	Return	List<LoanDTO>	

5 TEMPLATE CODE STRUCTURE

5.1 PACKAGE: COM.LOANAPPLICATION

Resources

LoanApplication(Class)	This is the Spring Boot starter class of the application.	Already Implemented

5.2 PACKAGE: COM.LOANAPPLICATION.REPOSITORY

Resources

Class/Interface	**Description**	**Status**
BankRepository (interface)	● Repository interface exposing CRUD functionality for Bank Entity. ● You can go ahead and add any custom methods as per requirements. ● It must contain a method to fetch all banks having less interest rate than passed one.	Partially implemented.
LoanRepository (interface)	● Repository interface exposing CRUD functionality for Loan Entity. ● You can go ahead and add any custom methods as per requirements. ● It must contain a method to fetch all loans by status. ● It must contain a method to fetch all by applicant name ordered in ascending order.	Partially implemented.

5.3 PACKAGE: COM.LOANAPPLICATION.SERVICE

Resources

Class/Interface	**Description**	**Status**
BankService (interface)	● Interface to expose method signatures for bank related functionality. ● Do not modify, add or delete any method.	Already implemented.
LoanService (interface)	● Interface to expose method signatures for loan related functionality. ● Do not modify, add or delete any method.	Already implemented.

5.4 PACKAGE: COM.LOANAPPLICATION.SERVICE.IMPL

Class/Interface	**Description**	**Status**
BankServiceImpl (class)	● Implements BankService. ● Contains template method implementation. ● Need to provide implementation for bank related functionalities. ● Do not modify, add or delete any method signature	To be implemented.
LoanServiceImpl (class)	● Implements LoanService. ● Contains template method implementation. ● Need to provide implementation for loan related functionalities. ● Do not modify, add or delete any method signature	To be implemented.

5.5 PACKAGE: COM.LOANAPPLICATION.CONTROLLER

Resources

Class/Interface	**Description**	**Status**
BankController (Class)	● Controller class to expose all rest-endpoints for bank related activities. ● May also contain local exception handler methods	To be implemented
LoanController (Class)	● Controller class to expose all rest-endpoints for loan related activities. ● May also contain local exception handler methods	To be implemented

5.6 PACKAGE: COM.LOANAPPLICATION.DTO

Resources

Class/Interface	**Description**	**Status**
BankDTO (Class)	Use appropriate annotations for validating attributes of this class.	Partially implemented.
LoanDTO (Class)	Use appropriate annotations for validating attributes of this class.	Partially implemented.

5.7 PACKAGE: COM.LOANAPPLICATION.ENTITY

Resources

Class/Interface	Description	Status
Bank (Class)	● This class is partially implemented. ● Annotate this class with proper annotation to declare it as an entity class with **id** as primary key. ● Map this class with a **bank table**. ● Generate the **id** using the IDENTITY strategy	Partially implemented.
Loan (Class)	● This class is partially implemented. ● Annotate this class with proper annotation to declare it as an entity class with **id** as primary key. ● Map this class with a **loan table**. ● Generate the **id** using the IDENTITY strategy	Partially implemented.

5.8 PACKAGE: COM.LOANAPPLICATION.EXCEPTION

Resources

Class/Interface	Description	Status
NotFoundException (Class)	● Custom Exception to be thrown when trying to fetch or delete the bank/loan info which does not exist. ● Need to create Exception Handler for same wherever needed (local or global)	Already implemented.

6 EXECUTION STEPS TO FOLLOW FOR BACKEND

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers need to go to the Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal.**
3. **cd into your backend project folder**
4. **To build your project use command:**

   mvn clean package -Dmaven.test.skip

5. **To launch your application, move into the target folder (cd target). Run the following command to run the application:**

   java -jar <your application jar file name>

6. **This editor Auto Saves the code.**
7. **If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.**
8. **These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.**
9. **To test any Restful application, the last option on the left panel of IDE, you can find ThunderClient, which is the lightweight equivalent of POSTMAN.Please use 127.0.0.1 instead of localhost to test rest endpoints.**
10. **To test any UI based application the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.**
11. **Default credentials for MySQL:**
    a. **Username: root**
    b. **Password: pass@word1**

11. **To login to mysql instance: Open new terminal and use following command:**

    a. **sudo systemctl enable mysql**

    b. **sudo systemctl start mysql**

    NOTE: After typing the second sql command (sudo systemctl start mysql), you may encounter a warning message like :

    ```
    System has not been booted with systemd as init system (PID 1). Can't operate.
    Failed to connect to bus: Host is down
    ```

    >> Please note that this warning is expected and can be disregarded. Proceed to the next step.

    c. **mysql -u root -p**

    The last command will ask for password which is ‘pass@word1’

12. **Mandatory: Before final submission run the following command:**

    mvn test

13. **You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.**