**Project: OnlineBanking**

|  |
| --- |
| TEAM MEMBERS |

|  |
| --- |
| 1. SHARYU |
| 1. SUSHMA |
| 1. VARALAKSHMI |
| 1. ANUSHA |
| 1. VASAVI |
| 1. HARI |
| 1. SANDEEP |
| 1. PREETHI |
| 1. HEMA |
| 1. FARHA |

FUNCTIONAL SPECIFICATIONS

|  |  |
| --- | --- |
| Project Code |  |
| Project Name | OnlineBanking |

**FUNCTIONAL SPECIFICATIONS**

**Table of Contents**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No. |  | | Page No. |
| 1 | INTRODUCTION | |  |
| 2 | SYSTEM OVERVIEW | |  |
| 3 | DATA ORGANIZATION | |  |
| 4 | REST APIs to be Built |  |  |
| 5 | ASSUMPTIONS | |  |
| 6 | EXPECTATIONS | |  |
| 7 | ACCEPTANCE CRITERIA | |  |
| 8 | TRACEABILITY TO REQUIREMENTS | |  |
| 9 | Acronyms | |  |
| 10 | OUTPUT SCREENSHOTS | |  |

**FUNCTIONAL SPECIFICATION**

**1**.**Introduction**

Online banking offers customers almost every service traditionally available through a local branch including deposits, transfers, and online bill payments. Virtually every banking institution has some form of [online banking](https://www.investopedia.com/articles/pf/11/benefits-and-drawbacks-of-internet-banks.asp), available both on desktop versions and through mobile apps.

With online banking, consumers aren't required to visit a bank branch to complete most of their basic banking transactions. They can do all of this at their own convenience, wherever they want—at home, at work, or on the go.

**Scope and Overview:**

Scope of Online Banking Services  
  
Customers can perform financial transactions like transfer funds online, pay bills, apply for loans and open a savings account among various other debit card transactions.

**2. System Overview**

The “OnlineBanking” should support basic functionalities for all below listed users.

* User

* Admin

**Authentication & Authorization**

**Authentication** is a process designed to confirm whether a user has the right to perform a specific operation or access a specific resource (e.g. a file). The purpose of **authorization** is to control the access, therefore it takes place after authentication, i.e. the operation of establishing user id.

Take the example of an online banking system - where, after logging in, the user can perform certain operations during an active session. When the session expires, the user loses their rights until the next correct login.

**Environment:**

The system will be developed on any Windows OS machine using J2EE, Hibernate and Spring.

* Intel hardware machine (PC P4-2.26 GHz, 512 MB RAM, 40 GB HDD)
* Server – Apache Tomcat 8 or higher
* Database – Mysql or higher
* JRE 8
* Eclipse IDE or Spring Tool Suite

**3**.**Data Organization**

This section explains the data storage requirements of ONLINE BANKING data description along with suggested table (database) structure. The following section explains few of the tables (fields) with description. However, in similar approach need to be considered for all other tables.

**USER DATABASE**

(1) MY\_PROFILE :

|  |  |
| --- | --- |
| Field name | Description |
| Firstname | User firstname |
| Lastname | User lastname |
| Phoneno | 10-digits phone number |
| Emailid | User email id |
| Username | Username |

(2) CHEQUE\_BOOK\_REQUEST :

|  |  |
| --- | --- |
| Field name | Description |
| User name | User name |
| Account type | Type of the account |
| Description | Description |

(3) BANK\_STATEMENT :

|  |  |
| --- | --- |
| Field name | Description |
| Account number | Account number is foreign key references from saving account |
| Statement date | Statement date |
| Particulars | Particulars |
| Debit | Debit the amount |
| Credit | Credit the amount |
| Balance | Remaining balance |

(4) WITHDRAW :

|  |  |
| --- | --- |
| Field name | Description |
| Withdraw date | Date of withdraw |
| Transaction id | Transaction id |
| Account title | Title of account |
| Amount | Withdraw Amount |
| Receiving institution | Receiving institution |

(5) DEPOSIT :

|  |  |
| --- | --- |
| Field name | Description |
| Account number | Account number is foreign key references from saving account |
| Deposit date | Date of deposit |
| Transaction id | Transaction id |
| Account title | Title of account |
| Amount | Deposit amount |
| Delivering institution | Delivering institution |

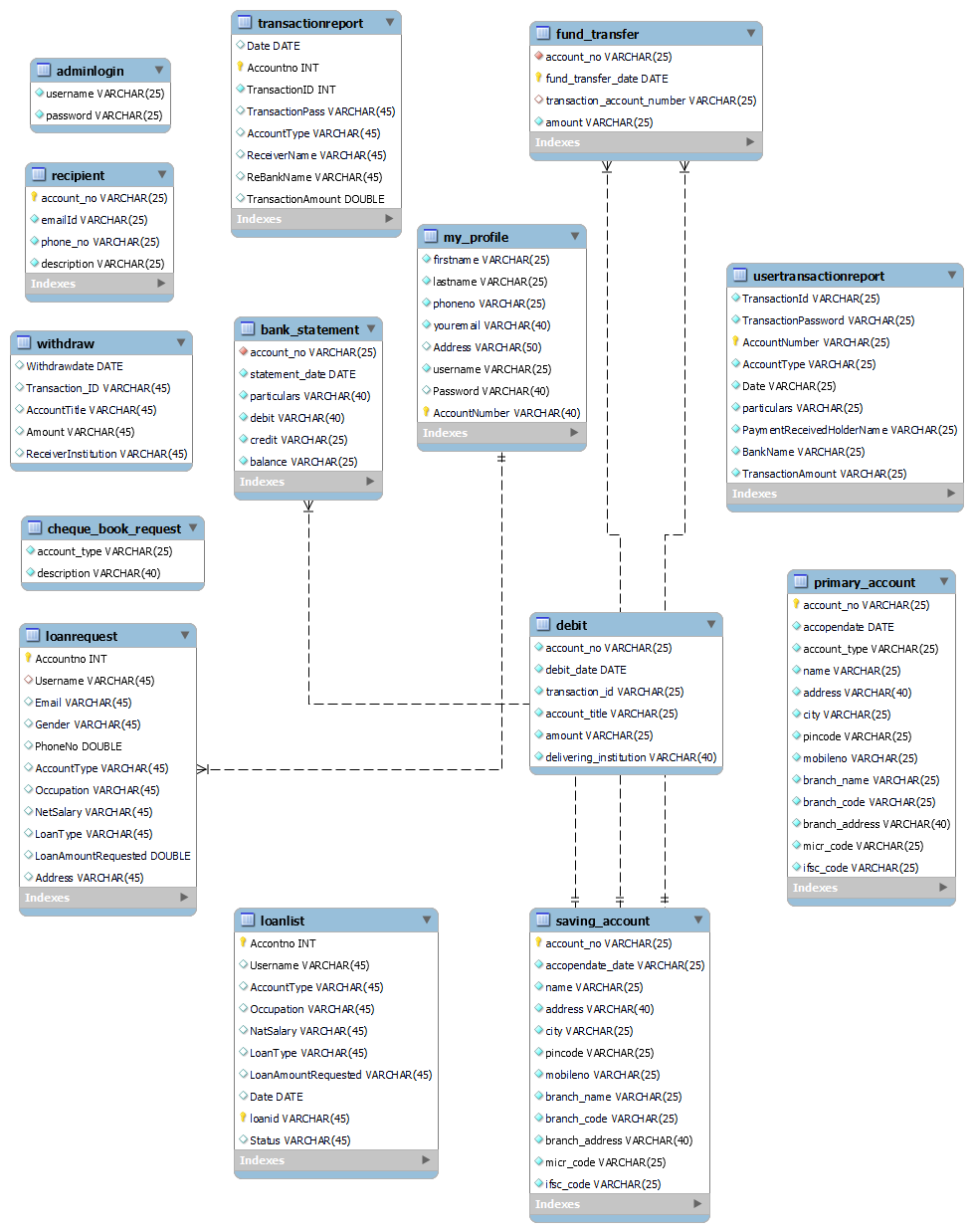
(6) LOAN :

|  |  |
| --- | --- |
| Field name | Description |
| Name | Name of the user |
| Email | Email of the user |
| Phone number | 10-digits phone number |
| Gender | Gender |
| Id number | Id number of user |
| Address | Address of the user |
| Designation | Designation of user |
| Net salary | Net salary of user |
| Type of loan | Loan type |
| Loan amount requested | How much amount requested by user |

(7) FUND\_TRANSFER :

|  |  |
| --- | --- |
| Field name | Description |
| Account number | Here account number is foreign key references from saving account |
| Fund transfer date | Here Fund transfer date is primary key |
| Transaction account number | Here Transaction account number is foreign key references from saving account |
| Amount | Transfer Amount |

**ER Diagram Of Online Banking**



**FUNCTIONAL SPECIFICATION**

**4. REST APIs to be Built**

Create following REST Resources which are required in the application.

1.Creating Admin Entity: Create Spring Boot with Spring Data JPA

Technology stack:

* Spring Boot
* Spring REST
* Spring Data JPA

Here we will have multiple layers into the application

i.Create an Entity : Admin

ii.Create a AdminRepository interface and will make use of Spring Data JPA

iii. Create a AdminService class and will expose all these services.

iv. Finally, create a AdminController will have the following URI’s:

|  |  |  |  |
| --- | --- | --- | --- |
| URI | Methods | Description | Format |
| /admin | POST | Login user | JSON |
| /admin | GET |  | JSON |

2.Creating UserRegistration Entity: Create Spring Boot with Spring Data JPA

Technology stack:

* Spring Boot
* Spring REST
* Spring Data JPA

Here we will have multiple layers into the application

i.Create an Entity : UserRegistration

ii.Create a UserRegistrationRepository interface and will make use of Spring Data JPA.

iii. Create a UserRegistrationService class and will expose all these services.

iv. Finally, create a UserRegistrationController will have the following URI’s:

|  |  |  |  |
| --- | --- | --- | --- |
| URI | Methods | Description | Format |
| /register | GET | Give all the UserRegistration Details | JSON |
| /registration | POST | Add UserRegistration Details | JSON |
| /registration | POST |  | JSON |

3.Creating Useraccount Entity: Create Spring Boot with Spring Data JPA

Technology stack:

* Spring Boot
* Spring REST
* Spring Data JPA

Here we will have multiple layers into the application

i.Create an Entity : Useraccount

ii.Create a UseraccountRepository interface and will make use of Spring Data JPA

iii. Create a UseraccountService class and will expose all these services.

iv. Finally, create a UseraccountController will have the following URI’s:

|  |  |  |  |
| --- | --- | --- | --- |
| URI | Methods | Description | Format |
| /userdetail | GET | Give all Useraccount Details | JSON |
| /userdetail | POST | Add the Useraccount details | JSON |
| /userdetail | DELETE | Delete a single Useraccount based on username | JSON |

4.Creating Primaryaccount Entity: Create Spring Boot with Spring Data JPA

Technology stack:

* Spring Boot
* Spring REST
* Spring Data JPA

Here we will have multiple layers into the application

i.Create an Entity : primaryaccount

ii.Create a PrimaryaccountRepository interface and will make use of Spring Data JPA

iii. Create a PrimaryaccountService class and will expose all these services.

iv. Finally, create a PrimaryaccountController will have the following URI’s:

|  |  |  |  |
| --- | --- | --- | --- |
| URI | Methods | Description | Format |
| /primaryacdetail | GET | Give all the Primaryaccount Details | JSON |
| /primaryacdetail | POST | Add all the Primaryaccount Details | JSON |

5.Creating Savingaccount Entity: Create Spring Boot with Spring Data JPA

Technology stack:

* Spring Boot
* Spring REST
* Spring Data JPA

Here we will have multiple layers into the application

i.Create an Entity : Savingaccount

ii.Create a SavingaccountRepository interface and will make use of Spring Data JPA

iii. Create a SavingaccountService class and will expose all these services.

iv. Finally, create a SavingaccountController will have the following URI’s:

|  |  |  |  |
| --- | --- | --- | --- |
| URI | Methods | Description | Format |
| /savingacdetail | GET | Give all the Primaryaccount Details | JSON |
| /savingacdetail | POST | Add the Details of Savingaccount | JSON |

6.Creating Accountstatement Entity: Create Spring Boot with Spring Data JPA

Technology stack:

* Spring Boot
* Spring REST
* Spring Data JPA

Here we will have multiple layers into the application

i.Create an Entity : Accountstatement

ii.Create a AccountstatementRepository interface and will make use of Spring Data JPA

iii. Create a AccountstatementService class and will expose all these services.

iv. Finally, create a AccountstatementController will have the following URI’s:

|  |  |  |  |
| --- | --- | --- | --- |
| URI | Methods | Description | Format |
| /statement | GET | Gives all the Details of Accountstatement | JSON |
| /statement/1 | GET | Gives the Details of Accountstatement based on account number | JSON |
| /statement | POST | Add the Details of Accountstatement | JSON |
| /statement | DELETE | Delete the Details of Accountstatement based on account number | JSON |

7.Creating Chequebook Entity: Create Spring Boot with Spring Data JPA

Technology stack:

* Spring Boot
* Spring REST
* Spring Data JPA

Here we will have multiple layers into the application

i.Create an Entity : Chequebook

ii.Create a ChequebookRepository interface and will make use of Spring Data JPA

iii. Create a ChequebookService class and will expose all these services.

iv. Finally, create a ChequebookController will have the following URI’s:

|  |  |  |  |
| --- | --- | --- | --- |
| URI | Methods | Description | Format |
| /chequerequest | GET | Gives all Requests of Chequebook | JSON |
| /chequerequest | POST | Add all Chequebook Requests | JSON |
| /chequerequest | DELETE | Delete the Request based on username | JSON |

8.Creating Loan Entity: Create Spring Boot with Spring Data JPA

Technology stack:

* Spring Boot
* Spring REST
* Spring Data JPA

Here we will have multiple layers into the application

i.Create an Entity : Loan

ii.Create a LoanRepository interface and will make use of Spring Data JPA

iii. Create a LoanService class and will expose all these services.

iv. Finally, create a LoanController will have the following URI’s:

|  |  |  |  |
| --- | --- | --- | --- |
| URI | Methods | Description | Format |
| /loan | GET | Gives all the Details of Loan | JSON |
| /loan | GET | Gives Details of Loan based on account number | JSON |
| /loan | POST | Add the Details of Loan | JSON |
| /loan | DELETE | Delete Details of Loan based on account number | JSON |

10.Creating Withdraw Entity: Create Spring Boot with Spring Data JPA

Technology stack:

* Spring Boot
* Spring REST
* Spring Data JPA

Here we will have multiple layers into the application

i.Create an Entity : Withdraw

ii.Create a WithdrawRepository interface and will make use of Spring Data JPA

iii. Create a WithdrawService class and will expose all these services.

iv. Finally, create a WithdrawController will have the following URI’s:

|  |  |  |  |
| --- | --- | --- | --- |
| URI | Methods | Description | Format |
| /withdraw | GET | Gives all Withdraw Details | JSON |
| /withdraw | GET | Gives Withdraw Details of Particular Transaction Id | JSON |
| /withdraw | POST | Add Withdraw Details | JSON |

11.Creating deposit Entity: Create Spring Boot with Spring Data JPA

Technology stack:

* Spring Boot
* Spring REST
* Spring Data JPA

Here we will have multiple layers into the application

i.Create an Entity : deposit

ii.Create a depositRepository interface and will make use of Spring Data JPA

iii. Create a depositService class and will expose all these services.

iv. Finally, create a depositController will have the following URI’s:

|  |  |  |  |
| --- | --- | --- | --- |
| URI | Methods | Description | Format |
| /deposit | GET | Gives the details of deposit based on account number | JSON |
| /deposit | POST | Add Details of Deposit | JSON |
| /deposit | DELETE | Deletethe details of deposit based on account number | JSON |

12.Creating Fundtransfer Entity: Create Spring Boot with Spring Data JPA

Technology stack:

* Spring Boot
* Spring REST
* Spring Data JPA

Here we will have multiple layers into the application

i.Create an Entity : Fundtransfer

ii. Create a FundtransferRepository interface and will make use of Spring Data JPA

iii. Create a FundtransferService class and will expose all these services.

iv. Finally, create a FundtransferController will have the following URI’s:

|  |  |  |  |
| --- | --- | --- | --- |
| URI | Methods | Description | Format |
| /transfer | GET | Gives all the details of Fundtransfer | JSON |
| /transfer | POST | Add all the details of Fundtransfer | JSON |

13.Creating Transaction Entity: Create Spring Boot with Spring Data JPA

Technology stack:

* Spring Boot
* Spring REST
* Spring Data JPA

Here we will have multiple layers into the application

i.Create an Entity : Transaction

ii.Create a Transaction Repository interface and will make use of Spring Data JPA

iii. Create a Transaction Service class and will expose all these services.

iv. Finally, create a Transaction Controller will have the following URI’s:

|  |  |  |  |
| --- | --- | --- | --- |
| URI | Method | Description | Format |
| /transaction | GET | Give all the Transaction Details | JSON |
| /transaction | POST | Add all the Transaction Details | JSON |
| /transaction | DELETE | Delete transaction details based on account\_no | JSON |

**5. Assumptions**

· User Interface: The type of client interface (front-end) to be supported - Angular based

· The administrator can add and remove products into the database on a weekly basis.

· You must not allow user to add same product twice.

· When you add product into cart the No. of Products selected will be incremented.

· If you remove the product from the cart, the counter will be decremented.

· The clear will remove all the products so that the No. of products will be zero

· The total amount will be calculated based on the product, accordingly, change the product counter & total amount.

**6. General Expectations**

· Participants must create the Class Diagram, Sequence Diagram and ER Diagram.

· Participants must do Unit testing and Functional Testing using POSTMAN tool.

· Integration of Angular and Spring Boot with Microservices should be done, referring Project 2 -Frond End Development Project.

· The server should be a concurrent server servicing multiple clients.

· Database can be implemented using Oracle 11g or above.

· To begin with, the application should support at least 1 admin and 2 customers.

· Compilation and Build should be done using Eclipse IDE or STS

· Source-code and all documents must be maintained (checked-in) in configuration management system (subversion)

· Coding standards (for Java) should be followed.

NOTE:

1. Validation of user Data

· Spring MVC using JSR-303 annotations.

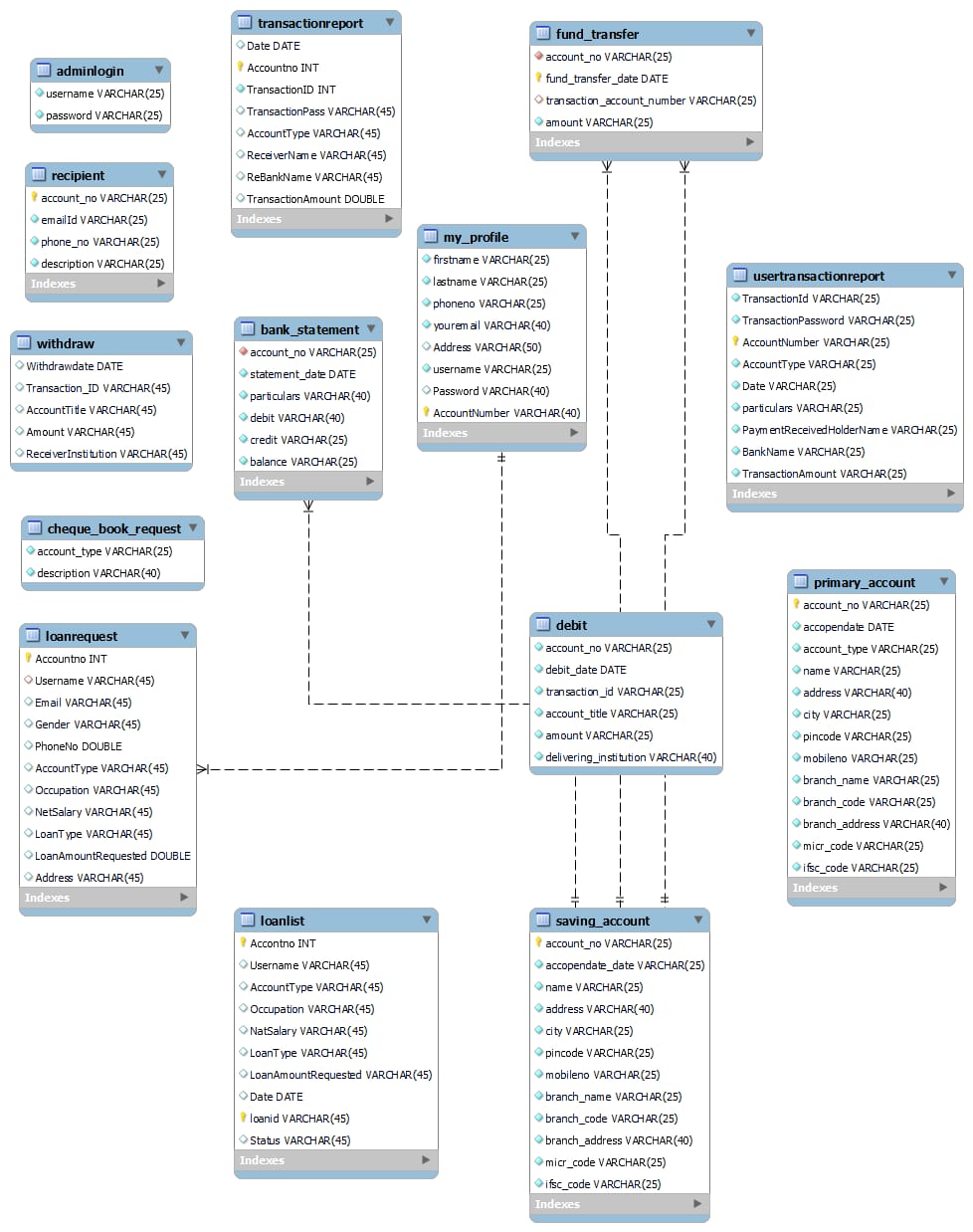
· AJAX validation without forcing the page to reload (Wherever applicable)

· JavaScript validation (if necessary)

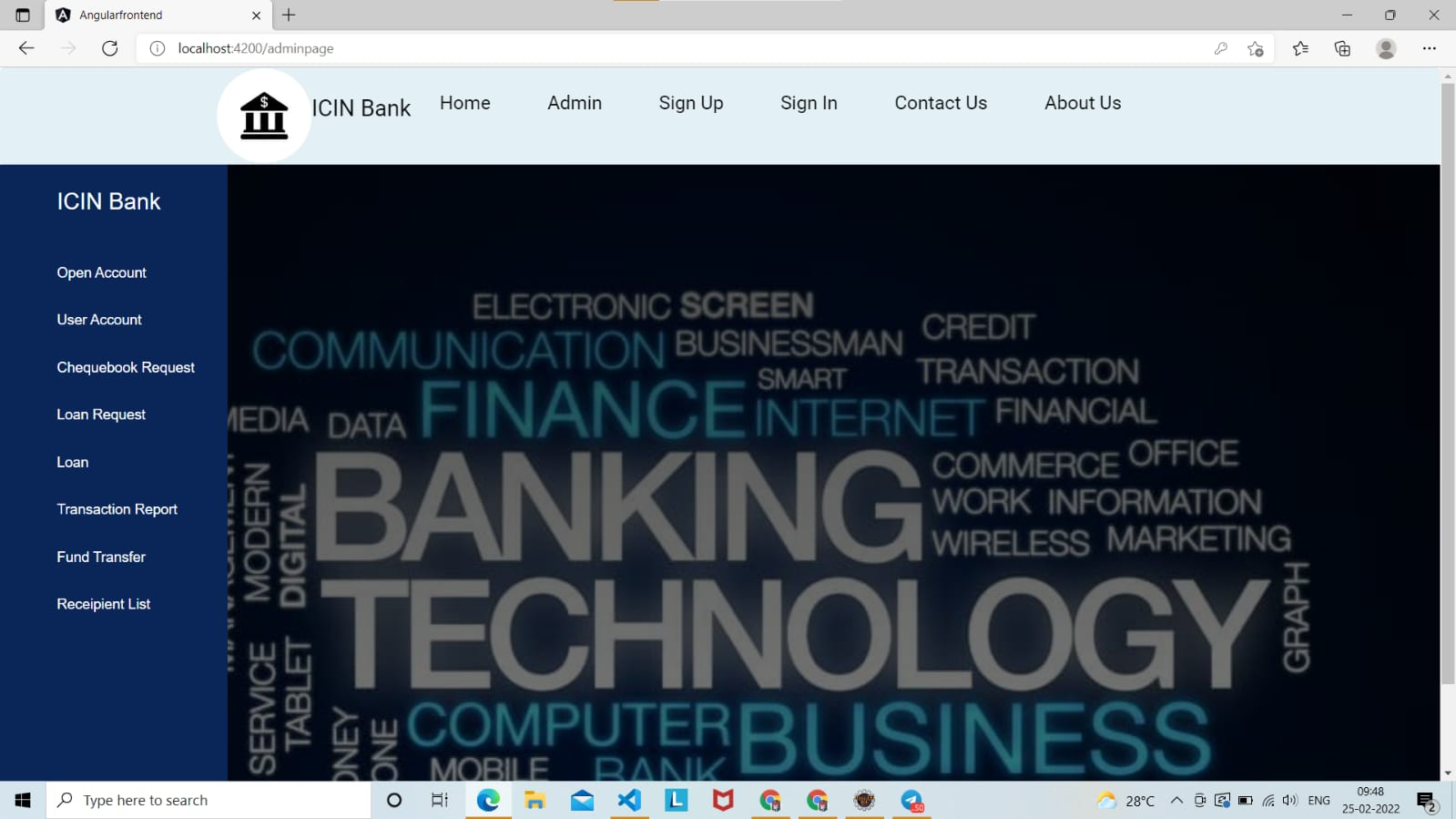
2. UI Design – (for Web Application) Use DIV/CSS or Semantic Elements to control the style and layout.

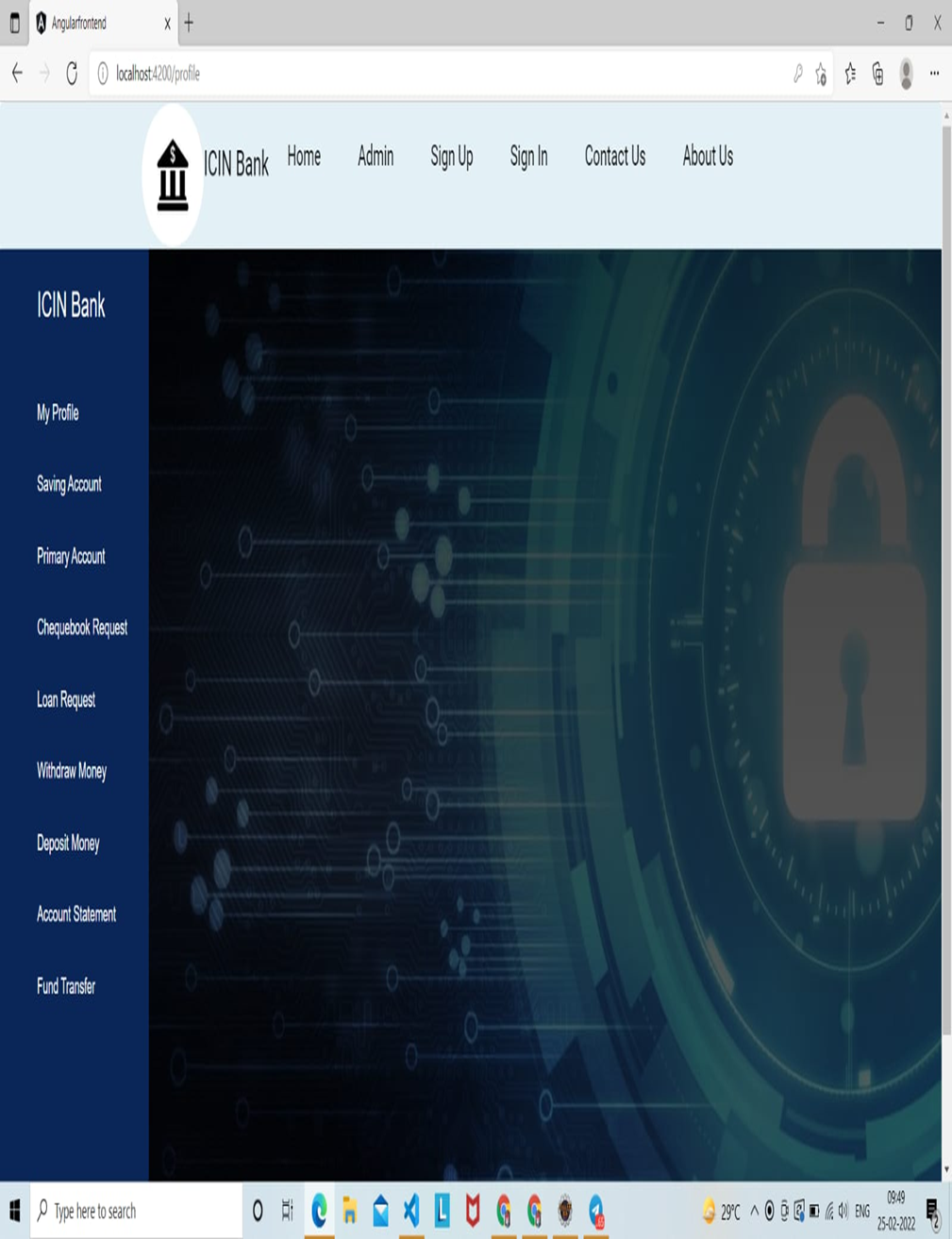
3. Create at least one SQL DML-statement inside PL/SQL blocks

**ER Diagram:-**

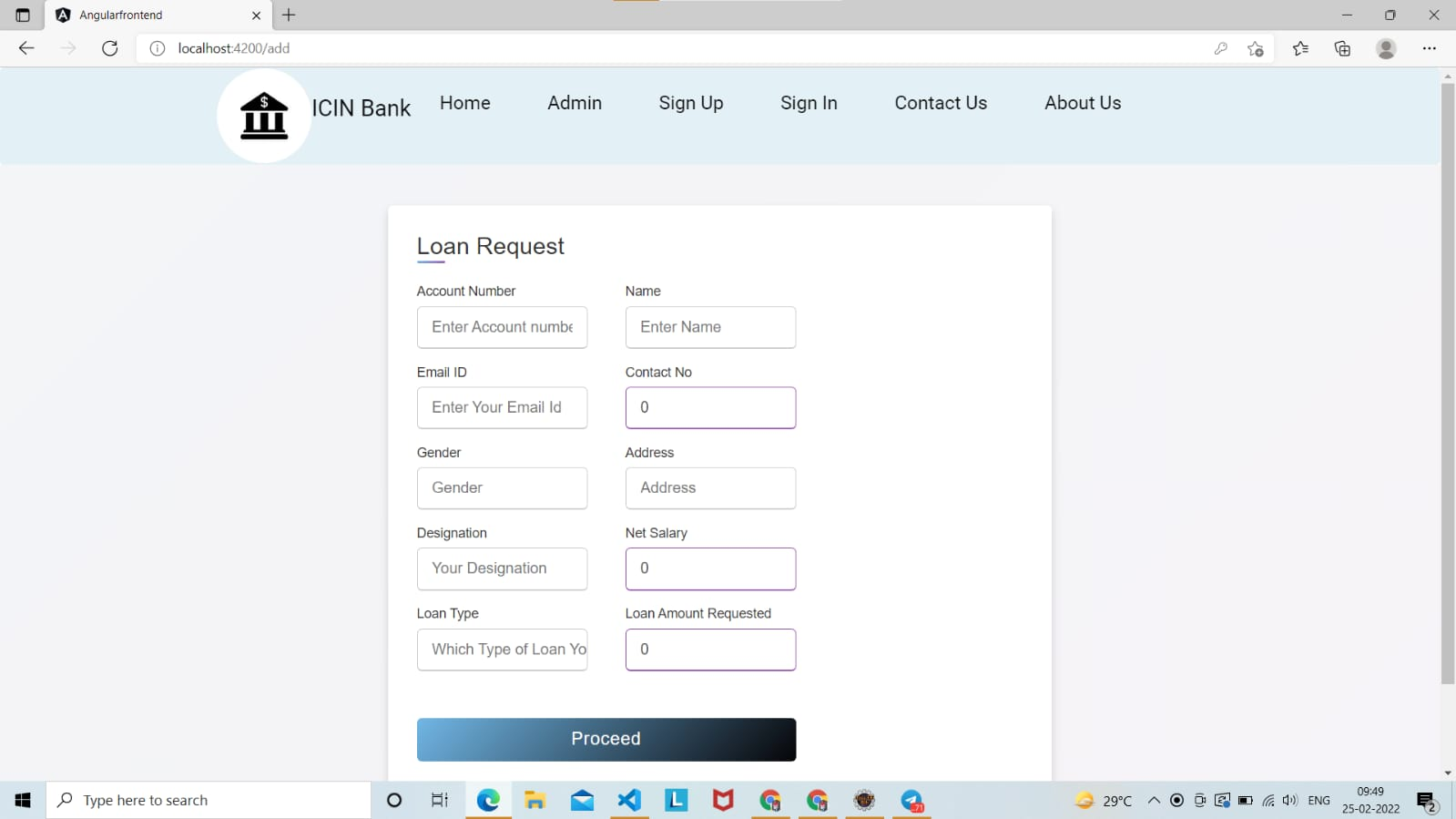


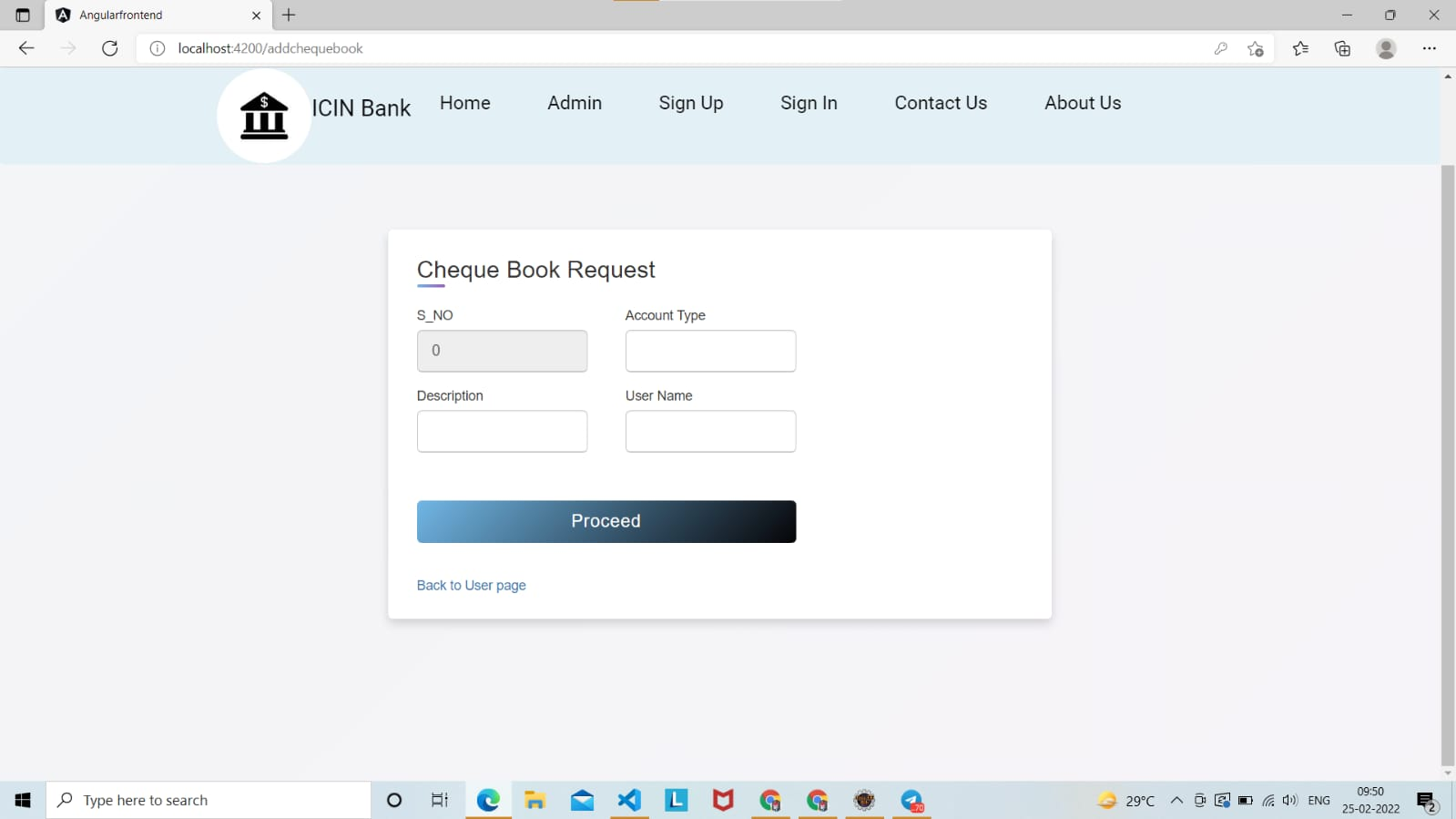
10. Output Screenshots

****

****

****

****

****

