**Global Bank Admin Management Application**

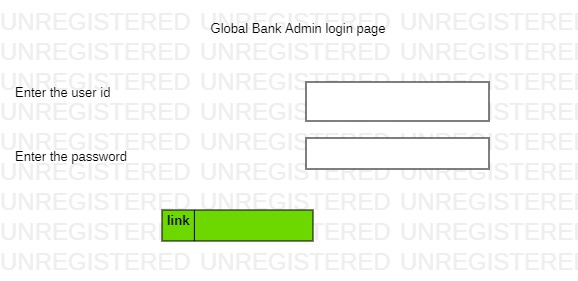
**Description:**

The Global Bank application allows admin to log in and performs admin-related activities. This application will allow performing the below operations:

1. Admin can be able to delete any customer
2. Admin can be able to add branch details
3. Admin can be able to delete branch details
4. Admin can be able to open an account for registered customer
5. Admin can be able to close an account for registered customers
6. Display all customers details
7. Displays all branch related details

**Proposed Wireframes**:

1. A standard login screen to validate admin credentials



1. Screen for admin operations menu



1. Screen for display customers details

Table

Description automatically generated

1. Screen for Display Branch Details

Table

Description automatically generated

1. Screen for admin to add or open customer account

Diagram

Description automatically generated

1. Screen for admin to add or create a branch details

Graphical user interface

Description automatically generated

**3. Toolchain**

Databases: MySQL

Presentation or View Layer: React

Backend processing: Spring and Springboot

Database Connectivity: Springboot data jpa

Version control systems: Git

Build Tools: Maven

**Development Flow**

The application development should be completed in 40 hours, as per the below order

Phase -1 : Backend Development: Backend Tasks – Code Project panel presentation

Phase -2: Frontend Development: Frontend Tasks – Code project panel presentation

**Business Requirement:**

There will be 5 main user stories required to be implemented covering the below use cases:

**User Stories**

|  |  |
| --- | --- |
| User story Id | Us-01 |
| User story title | Admin\_login |
| User Story Details | 1. admin should provide the user id and password for validation. |
| Acceptance Criteria | 1. Both username and password are mandatory, if not provided, error messages should be displayed. 2. Successful validation should redirect to the menu page   Note: here admin credentials are assumed to exist in the database, we don’t allow application to accept the admin details. |

|  |  |
| --- | --- |
| User story Id | Us-02 |
| User story title | Admin\_menu |
| User story details | This screen should display the menu for the admin functionalities like   1. Display Customers and provide provision for delete any customer 2. Display Branch details and provide provision for delete any branch details 3. Add customer account 4. Add branch details |

|  |  |
| --- | --- |
| User Story Id | Us-03 |
| User Story Title | Bank Branch registration |
| User story Details: | 1. Admin should be able to provide the below details for the new branch to setup 2. Branch\_id 3. Branch\_name 4. Branch\_city |
| Acceptance Criteria | 1. Front end is stable, neat and user friendly 2. All fields are mandatory and not filled respective error messages should be displayed. 3. Upon successful insertion of data, it should redirect to login page. 4. Branch id and branch name are manually entered and branch city must be selected from the drop-down list. 5. Once entered all the details, should re direct to menu page. |

|  |  |
| --- | --- |
| User\_Story id | Us\_04 |
| User story Title | Opening customer account |
| User Story Details: | Admin should be able to enter below details to open customer account:   1. account number 2. customer\_number 3. Branch id 4. Opening balance 5. Account\_opening\_date 6. Account\_type 7. Account\_status |
| Acceptance Criteria | Upon selecting this page, the following criteria must be satisfied   1. Customer number should be selected from the existing list 2. Branch id must be selected from the pop-menu only. 3. Opening balance must be greater than or equal to 5000 4. Account type must be either savings or current and should be selected from the existing list. 5. Account status must be either active or inactive and should be selected from the option box. |

|  |  |
| --- | --- |
| User\_Story id | Us\_05 |
| User story Title | Display All customer account details |
| User Story Details: | When admin opened this page, all the details of the customers are to be displayed in tabular form. |
| Acceptance Criteria | There must be an option to delete any particular record |

|  |  |
| --- | --- |
| User\_Story id | Us\_06 |
| User story Title | Display All branch details |
| User Story Details: | When admin opened this page, all the details of the bank branch details are to be displayed in tabular form. |
| Acceptance Criteria | There must be an option to delete any particular record |

**Backend layer user stories**

|  |  |
| --- | --- |
| User story Id | Us-01 |
| User story title | Admin Login |
| User Story Details | 1. Should be able to extract the values from request body using @RequestBody 2. Read the user details from database using spring data jpa and validate it with the UI values. 3. After validating should send response to React UI 4. Must use GET Method of communication |
| Acceptance Criteria | 1. Once user validation is done, view must return the main menu in react 2. If validation fails, view must return to registration page. 3. All validations must be performed at backend only |

|  |  |
| --- | --- |
| User story Id | Us-02 |
| User story title | Bank branch registration |
| User Story Details | 1. Should be able to extract the values from request body object 2. Read the user registration values from UI and pass it to service layer further to dao layer to perform insertion of record in database 3. Should return the model object after successful insertion of data 4. Must use POST method of communication |
| Acceptance Criteria | 1. Once branch details are successfully inserted, must return Boolean value and view must return the main menu 2. If failed appropriate exception should be raised and handled. |

|  |  |
| --- | --- |
| User story Id | Us-03 |
| User story title | Customer Account Creation |
| User Story Details | 1. Should be able to extract the values from request body object 2. Read the customer account details UI and pass it to service layer further to dao layer to perform insertion of record in database 3. Should return the Boolean object after successful customer account submission 4. Must use POST method of communication |
| Acceptance Criteria | 1. Only registered customers can open an account. 2. Account type can either savings or current |

|  |  |
| --- | --- |
| User story Id | Us-04 |
| User story title | Display Customer Details |
| User Story Details | 1. Should fetch all the customer details from the database and return it in the form of list. |
| Acceptance Criteria | 1. Should return only in the java list object |

|  |  |
| --- | --- |
| User story Id | Us-05 |
| User story title | Display Branch Details |
| User Story Details | Should fetch all the branch details from the database and return it in the form of list. |
| Acceptance Criteria | Should return only in the java list object |

**Database layer user stories**

|  |  |
| --- | --- |
| User story | User Story Details |
| Us\_01 | 1. DB Schema creation and setup in mysql database 2. Springboot project setup creation. 3. Develop the post method api to read data from view page. |
| Us\_02 | 1. Set up the appropriate methods to perform functions like branch registration, branch deletion, customer account opening and deletion, display all the customer details. |

**Presentation:**

1. No custom CSS, UX framework like bootstrap must be used
2. An Appropriate GoF design pattern should be implemented to compose and process the data received from backend APIs
3. SOLID principles should be implemented to develop reusable and modular components
4. UI app should have appropriate client-side validations
5. UI app should have the latest versions of available imported packages and libraries

**Methodology:**

Agile-based development methodology should be used to track and manage the progress of the whole process. As a developer, it is expected to update the Agile tools like JIRA with status updates and impediments (Optional)

Day wise plan for user stories

|  |  |
| --- | --- |
| Day -1 | Database Layer Us\_01 , Us\_02 ,  Frontend US\_01 |
| Day-2 | Frontend Us\_02, US\_03  Backend Us\_01, US\_02 |
| Day-3 | Front end Us\_04  Backend US\_03 |
| Day-4 | Front end Us\_05 and Us\_06  Backend US\_04, US\_05 |
| Day- 5 | Unit test cases, testing and ppt preparation. |