

**Team Members:**

|  |  |
| --- | --- |
| Team 5 | Team 6 |
| Rohan Patil (Scrum Master) | Abhisek Kumar (Scrum Master) |
| Anwesha Das | Aditi Singh |
| Arpan Mondal | Anindya Sundar Rana |
| Mansi Agarwal | Rishabh Rai |
| Rishav Dev | Rohit kumar |
| Anish Kumar Basu | Vidushi Razdan |
| Kumar Saurabh |  |

Capgemini India Technology Services

Plot 72 & 73, EPIP Zone, Vijayanagar, KIADB Export Promotion Industrial Area, Whitefield, Bengaluru, Karnataka 560066

Table of Contents

1. Introduction
2. Overview
3. Team Member Work Allocation
4. Epic & Stories
5. Use Cases
   * + Login
     + Account

* Add Account
* Update Account
* Delete Account
  + - Transaction
* Credit Using Slip
* Debit Using Slip
* Credit Using Cheque
* Debit Using Cheque
  + - Loan
* Loan Request
* Loan Disbursal
  + - Passbook
* Account Summary
* Passbook Maintenance

1. Test Case Scenario
2. BDD
3. Conclusion

Introduction

# Pecunia₹ Finance Limited, established in the year 2015, is an Indian private sector [bank](https://en.wikipedia.org/wiki/Bank) headquartered in Bangalore and is functioning only in Bangalore, [India](https://en.wikipedia.org/wiki/India). In January 2019, [Reserve Bank of India](https://en.wikipedia.org/wiki/Reserve_Bank_of_India) (RBI) issued the license to Pecunia₹ Finance Limited, to carry on banking business.

# They have one of the largest and most respected Wealth Management teams in India providing the widest range of solutions to individuals and employed professionals. As of December 30, 2018 Pecunia₹ has a net-worth of approximately Rs 730 crore and micro-finance loans of Rs 300 crore in its books.

# It offers a wide range of banking products and financial services for retail customers through a variety of delivery channels in the areas of [personal finance](https://en.wikipedia.org/wiki/Personal_finance),  [life insurance](https://en.wikipedia.org/wiki/Life_insurance) etc.

Overview

# Pecunia₹ Finance Limited has recorded substantial profit growth (18% profit in the third financial quarter of 2018-2019) in all financial quarters, and its customer base is increasing, targets other metropolitan cities of India to open new branches and offer financial services.

# The market research conducted by the bank in the cities Mumbai, Chennai and Hyderabad to understand the satisfaction of customers about their bank’s financial services revealed the following data.

|  |  |  |
| --- | --- | --- |
| Sl No |  |  |
| 1 | SBI | 3.1 |
| 2 | Vijaya bank | 2.8 |
| 3 | Canara Bank | 3.0 |
| 4 | Syndicate bank | 2.1 |
| 5 | ICICI | 3.6 |
| 6 | HDFC | 3.7 |
| 7 | Kotak | 3.3 |
| 8 | HDFC | 3.9 |

# Above data clearly shows that there is a still scope for offering financial services in the above cities and meet customer’s expectations.

# To operate in the above mentioned cities, Pecunia₹ Finance Limited is expanding its business and is looking for centralized application which helps them to offer financial services in other cities too.

# Pecunia₹ Finance Limited has a target of opening 20 new branches in the current financial year across major cities of India.

## Current IT Environment

# Pecunia₹ Finance Limited has a legacy system, and much of the IT systems are old.

# Most of the systems need human intervention. Some of them listed below,

# Legacy system will not support for accessing real time data in other branches.

# Extensive paperwork

# Managing data in a local system where the chances of vulnerability and loosing is high.

# Tedious loan disbursement process.

# Payments are inefficient

# So Centralized Online real-time Exchange banking application offers Pecunia₹ truly browser based / web-based solution with the opportunity to meet present challenges with reduced costs and enhanced customer management measures. Information management and Risk Management is the key.

# • Core Banking Solution will help streamline the current business processes with a supporting technology.

# • Bottlenecks and inefficiencies in the present system requiring process changes will be addressed.

Team Member Work Allocation

|  |  |
| --- | --- |
| Name | Work Allocated |
| Kumar Saurabh | Login |
| Vidushi Razdan | Add Account |
| Aditi Singh | Update Account |
| Rohit Kumar | Delete Account |
| Arpan Mondal | Credit Using Slip |
| Rohan Patil | Credit Using Cheque |
| Anwesha Das | Debit Using Slip |
| Anish Kumar Basu | Debit Using Cheque |
| Rishabh Rai | Loan Request |
| Anindya Sundar Rana | Loan Disbursal |
| Rishav Dev | Account Summary |
| Mansi Agarwal | Passbook Maintenance |
| Abhisek Kumar | Test Case & BDD |

Epic & Stories

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epic | Stories | As a/an | I want to | So that … |
| Login | Login | Employee | Log in | Employee can log in and perform any operation |
| Account | Account Creation | Employee | Create a new Account | A new account is added to the database |
| Update Customer Name | Employee | Update the customer name | The name of the customer is updated in the database |
| Update Customer Contact | Employee | Update the customer contact | The contact of the customer is updated in the database |
| Update Customer Address | Employee | Update the customer address | The address of the customer is updated in the database |
| Delete Account | Employee | Delete an existing account | The account status is set to “Closed” in the database |
| Transaction | Credit Using Slip | Employee | Credit amount by slip | Employee can credit amount in customer’s account |
| Credit Using Cheque | Employee | Credit amount by Cheque | Employee can credit amount in customer’s account |
| Debit Using Slip | Employee | Debit amount by slip | Employee can debit amount from customer’s account |
| Debit Using Cheque | Employee | Debit amount by Cheque | Employee can debit amount from customer’s account |
| Loan | Loan Request | Employee | Create request for Loan | Employee could see Loan Applicant’s loan details like Loan Amount, Tenure, and Credit Score. The loan applicants should have account in this bank. |
| Loan Disbursal | Employee | Approve or reject loan requests | Employee could accept or reject the loan requests and update the loan customers account balance |
| Passbook | Passbook Update | Employee | Update the passbook | Recent transactions canbe printed in the passbook |
| Account Summary | Employee | Display transactions between two given dates. | Track of transaction can be seen for a specific duration. |
| BDD & Test Cases | BDD & Junit Test Case | Developer | Want to check the functionalities | Bank can perform its online operations |

Use Cases

Login

Use Case Diagram:

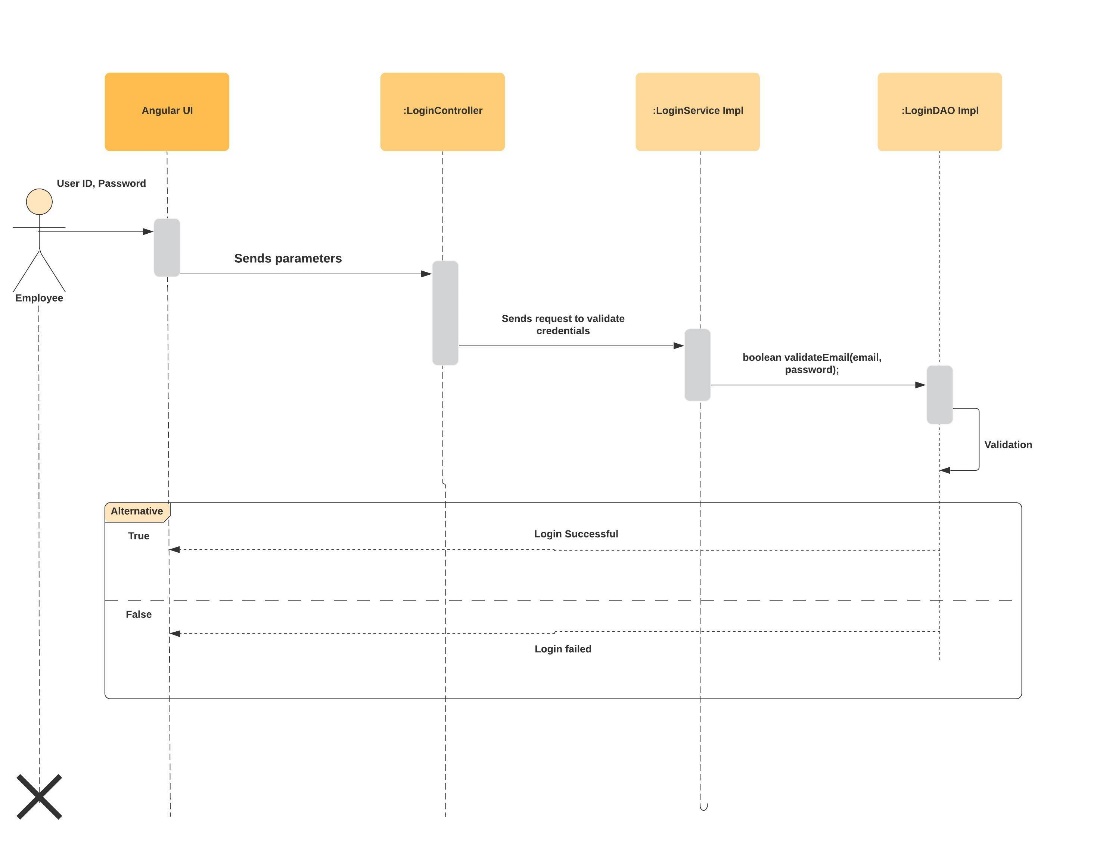
A) Log-in to the account

This module is designed for the employee to log-in. The form requires login credentials as username and password. If the entered credentials match with the one saved in the database, the user is allowed to log in successfully and perform different operations.

Validation:

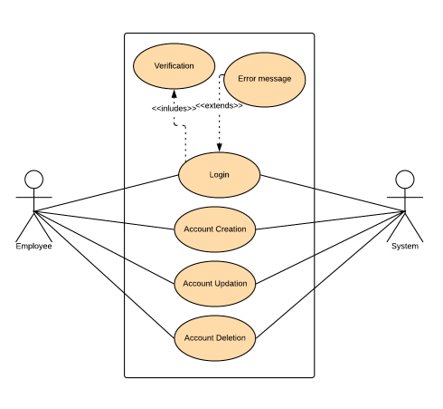
All fields are required.

Sequence Diagram:



Account Management

Use Case Diagram:



A) Add an Account

This module is designed to create a new bank account in the database. The form has proper validation.

Validation:

All fields are required.

Address fields: Address Lines 1 and 2 can be alphanumeric and special characters like /-\( ‘ ), while the city, state and country fields only allow alphabets. The zip code field only allows 6 digit numbers.

Customer Name: Only alphabets allowed.

Customer Contact: Only 10-digit numbers allowed.

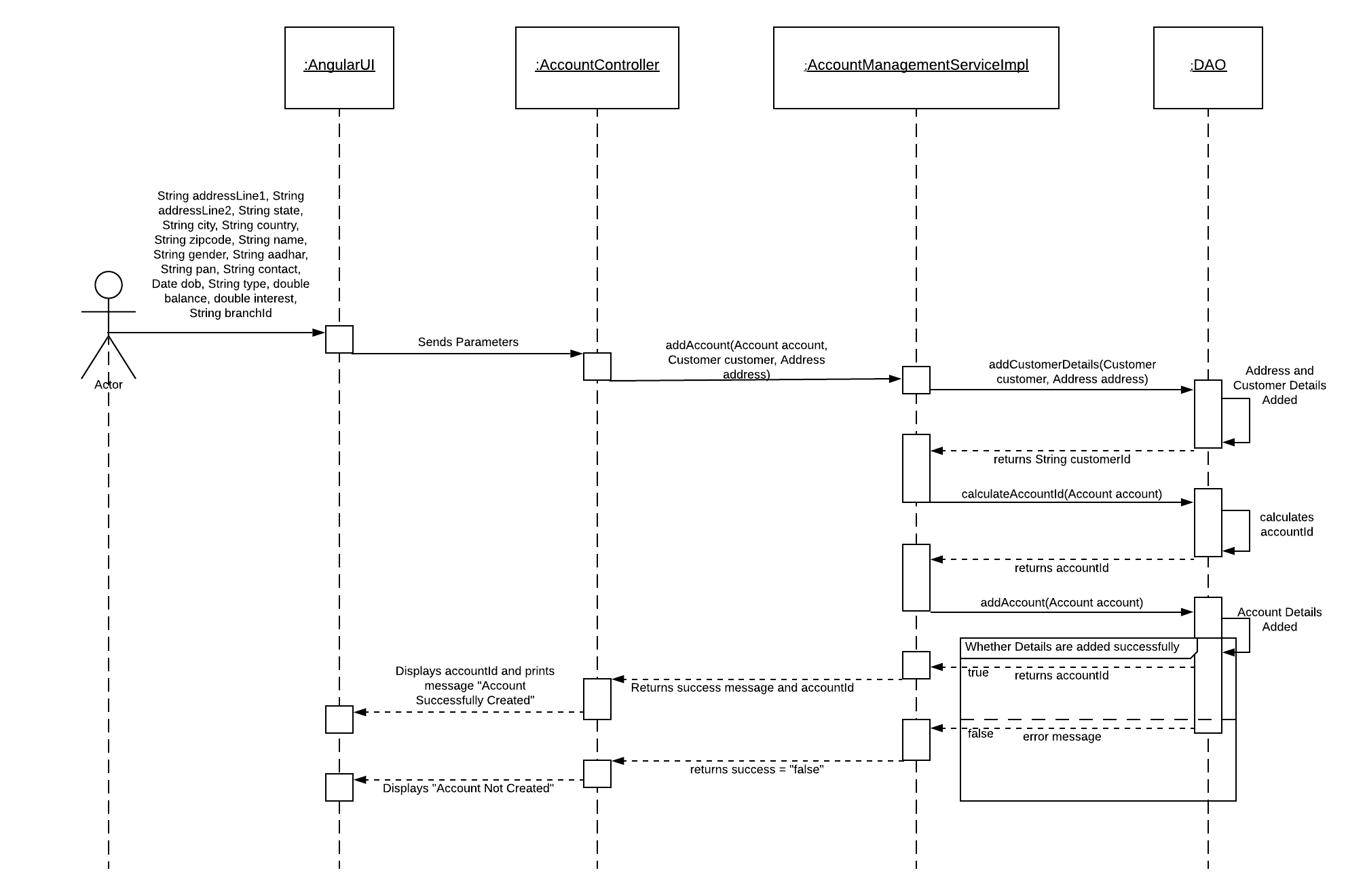
Customer Aadhar: Only 12-digit numbers allowed.

Customer PAN: First 5 characters should be alphabets, next 4 should be digits and the last character should be an alphabet.

Customer DOB: Only valid dates are accepted.

Customer Gender: Male/Female fields are accepted.

Sequence Diagram:



B) Update Customer Name

This module is given so that if any customer wants to update his/her name in the bank database, he/she can do so. In this, only the account ID and the new name is taken from the customer. The new name is updated in the database. The form has all the appropriate validations. A warning message is shown in case of invalid entries.

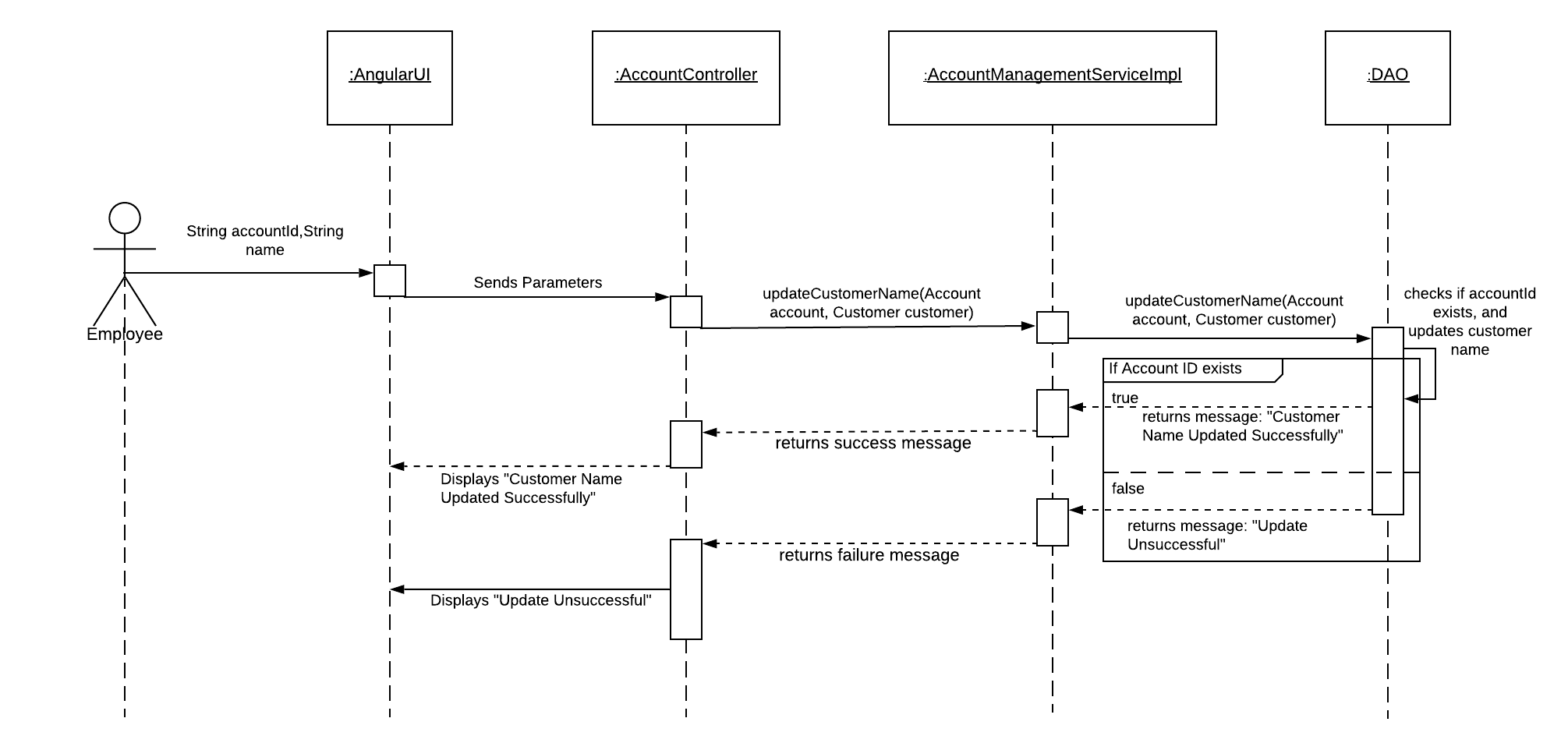
Validation:

All fields are required.

Account ID: Only 12-digit numbers are accepted.

Name: Only alphabets are allowed.

Sequence Diagram:



C) Update Customer Contact

This module is given so that if any customer wants to update his/her contact in the bank database, he/she can do so. In this, only the account ID and the new contact is taken from the customer. The new contact is updated in the database. The form has all the appropriate validations. A warning message is shown in case of invalid entries.

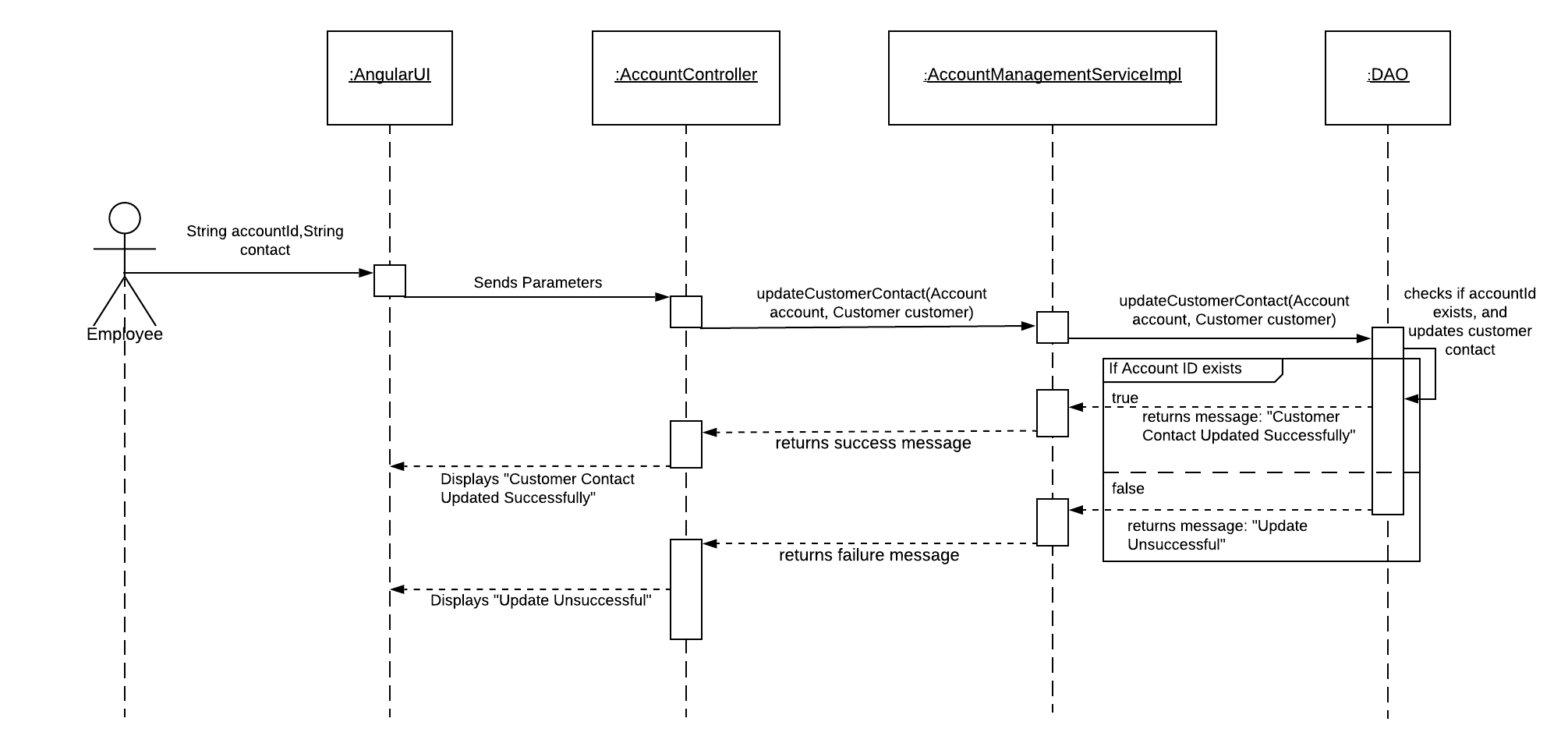
Validation:

All fields are required.

Account ID: Only 12-digit numbers are accepted.

Contact: Only 10-digit numbers allowed.

Sequence Diagram:



D) Update Customer Address

This module is given so that if any customer wants to update his/her contact in the bank database, he/she can do so. In this, only the account ID and the new contact is taken from the customer. The new contact is updated in the database. The form has all the appropriate validations. A warning message is shown in case of invalid entries.

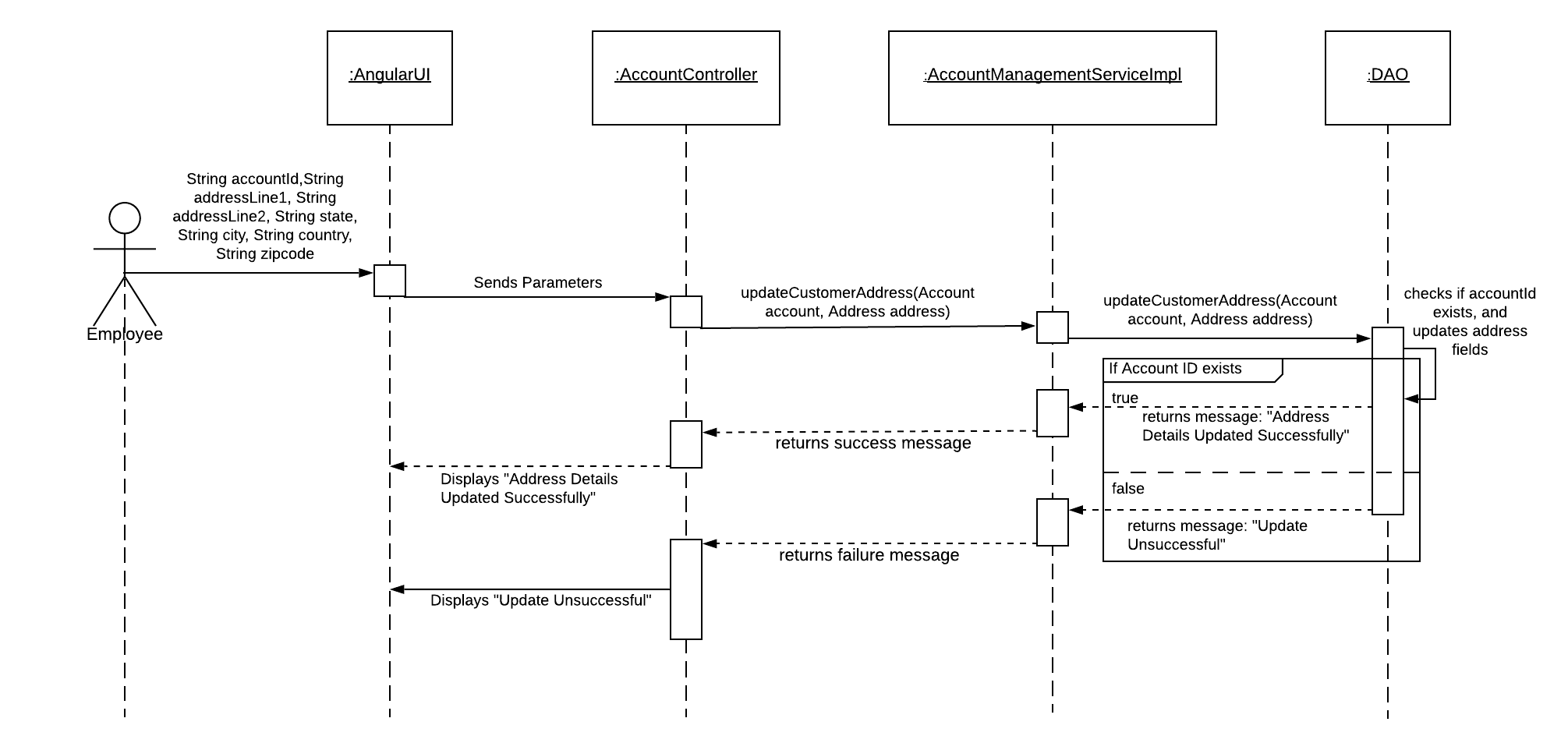
Validation:

All fields are required.

Account ID: Only 12-digit numbers are accepted.

Address fields: Address Lines 1 and 2 can be alphanumeric and special characters like /-\( ‘ ) are also allowed, while the city, state and country fields only allow alphabets. The zip code field only allows 6- digit numbers.

Sequence Diagram:



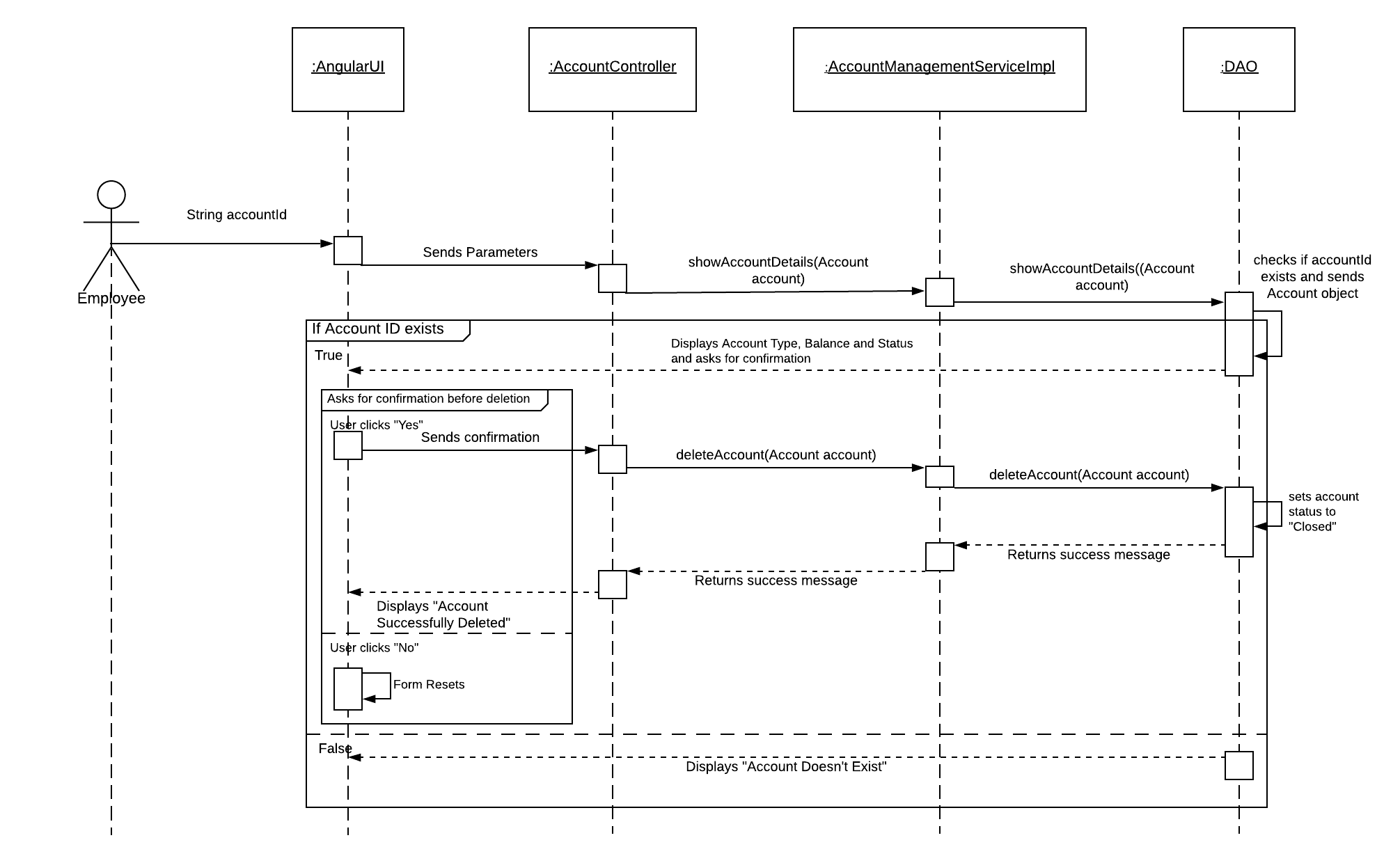
E) Delete Account

This module is given so that if any customer wants to close their account, he/she can do so. In this, only the account ID is taken from the customer. First, the account details like, balance, current account status and account ID is displayed, and a confirmation message is shown. If the user clicks on “yes” button, the account status is set to “closed”, otherwise just the form will reset. The form has all the appropriate validations. A warning message is shown in case of invalid entries.

Validation:

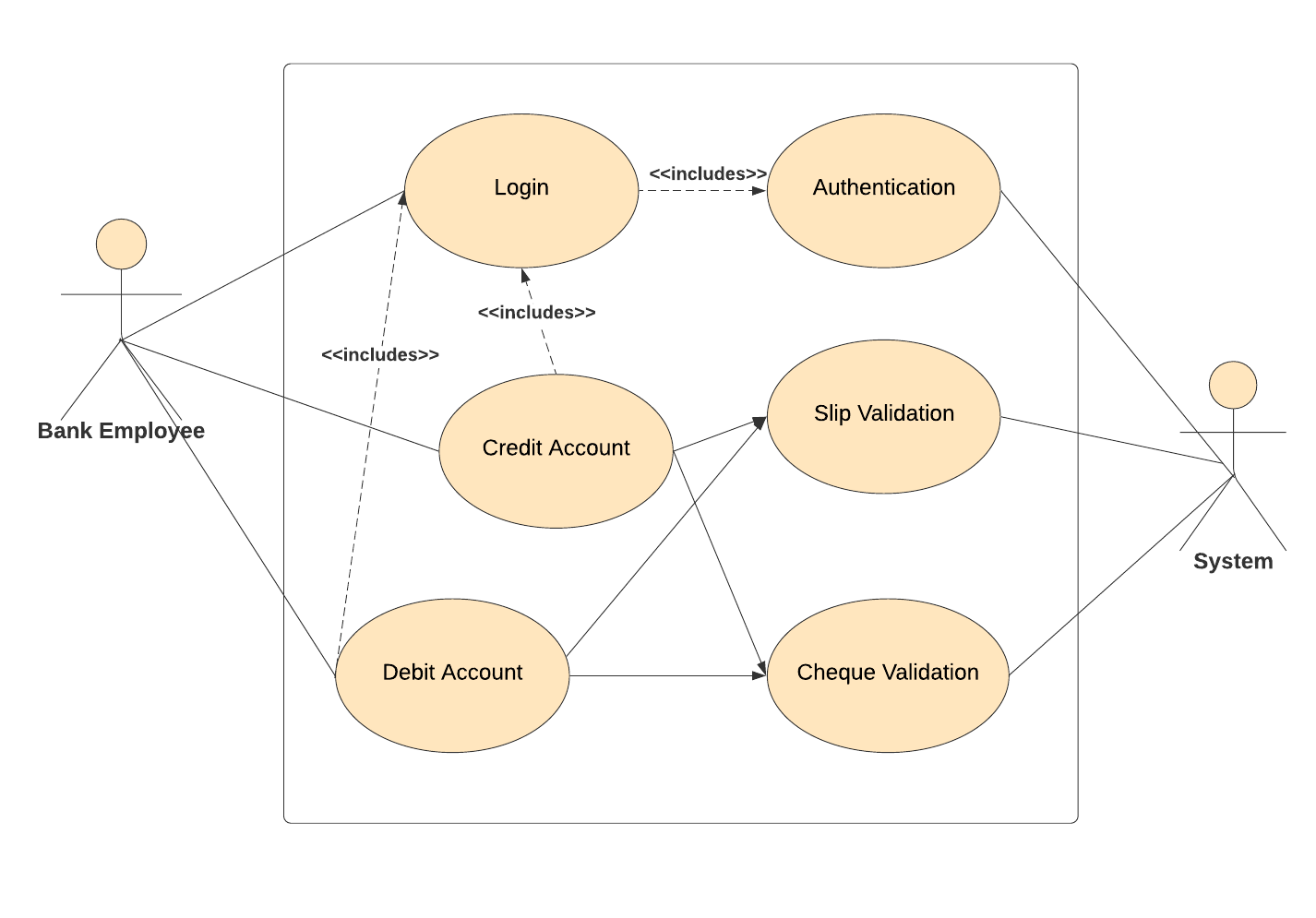
Account ID: Only 12-digit numbers are accepted. Required field.

Sequence Diagram:



Transaction

Use Case Diagram:



A) Credit Using Slip

This module is designed to credit amount to a bank account in the database. The form has proper validation.

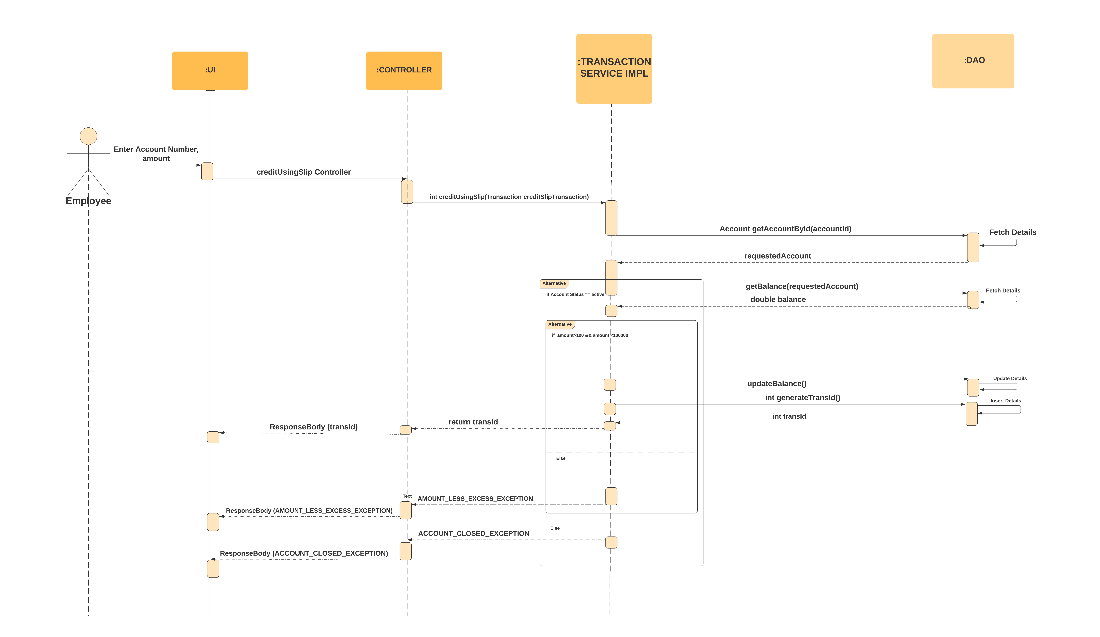
Validation:

All fields are required.

Account Number: Should be of 12 digits only.

Amount: Should only contain Numeric & amount must be between 100-100000

Sequence Diagram:



B) Debit Using Slip

This module is designed to debit amount to a bank account in the database. The form has proper validation.

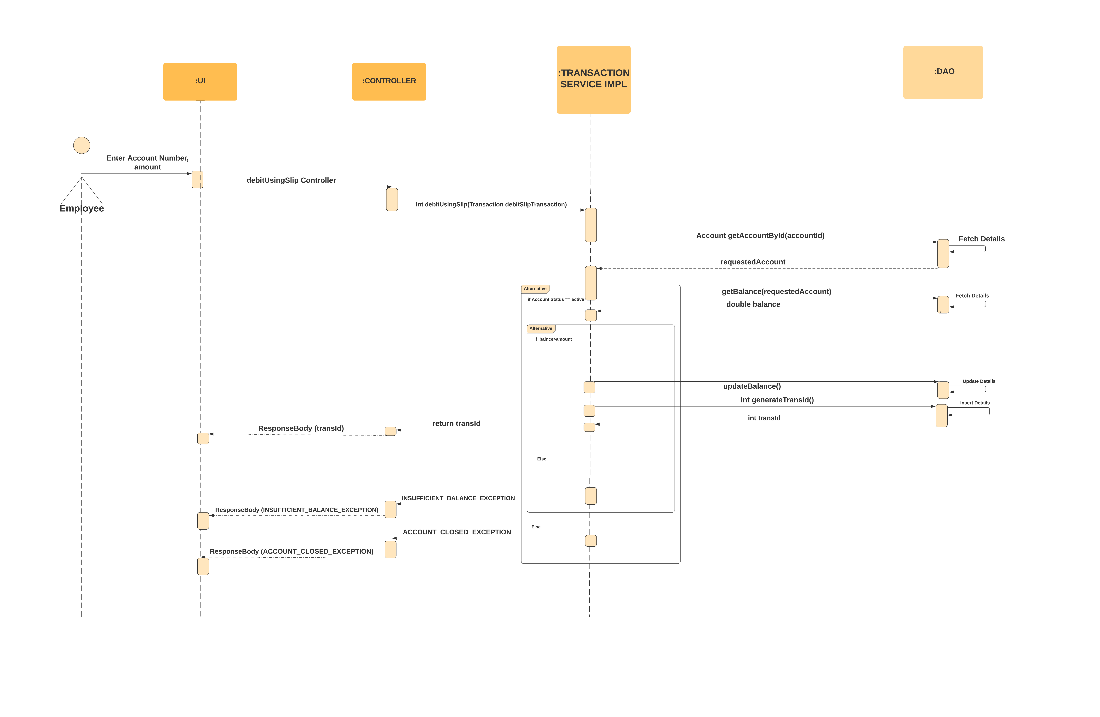
Validation:

All fields are required.

Account Number: Should be of 12 digits only.

Amount: Should only contain Numeric.

Sequence Diagram:



C) Debit Using Cheque

This module is designed to debit amount from a bank account in the database. The form has proper validation.

Validation:

All fields are required.

Account Number: Should be of 12 digits only.

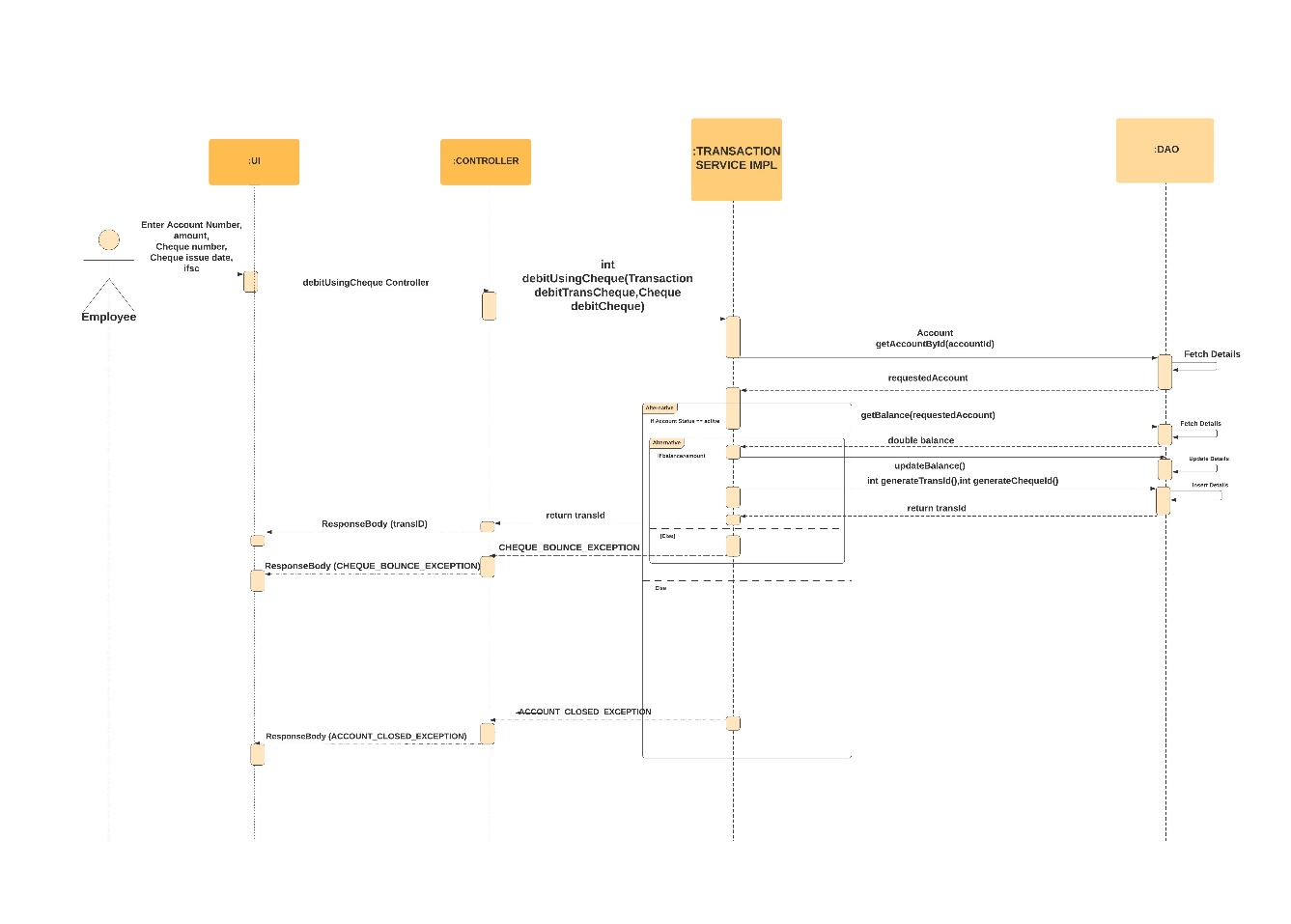
Amount: Should only contain Numerics & amount must be between 100-200000

Cheque number: Should contain 6 digits only.

IFSC: Should be alphanumeric and 10 characters only.

Cheque issue date: Should be within last 3 months up to present date.

Sequence Diagram:



D) Credit Using Cheque

This module is designed to credit amount from a bank account in the database. The form has proper validation.

Validation:

All fields are required.

Payee Account Number: Should be of 12 digits only.

Beneficiary Account Number: Should be of 12 digits only.

Amount: Should only contain Numerics & amount must be between 100-200000

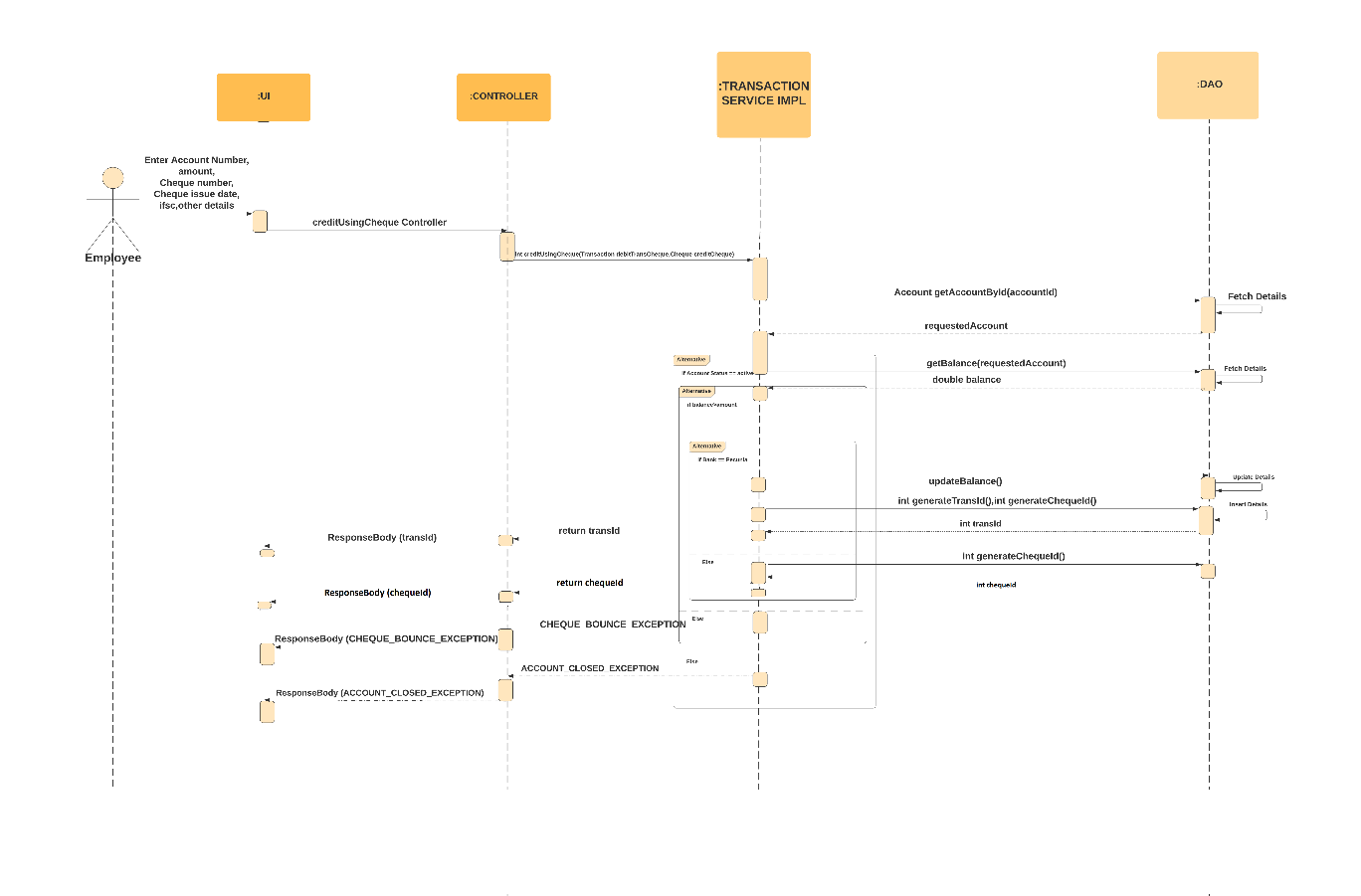
Cheque number: Should contain 6 digits only.

Bank name: It is a dropdown list containing bank names.

IFSC: Should be alphanumeric and 10 characters only.

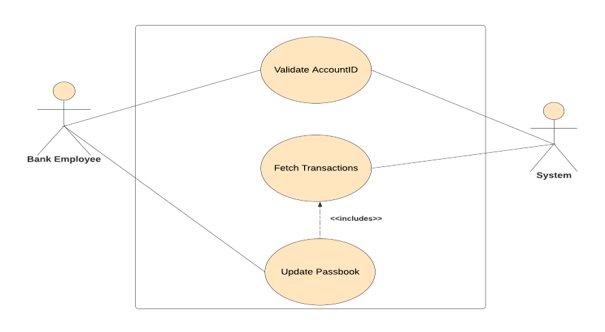
Cheque issue date: Should be within last 3 months up to present date.

Sequence Diagram:



Passbook Maintenance

Use Case Diagram:



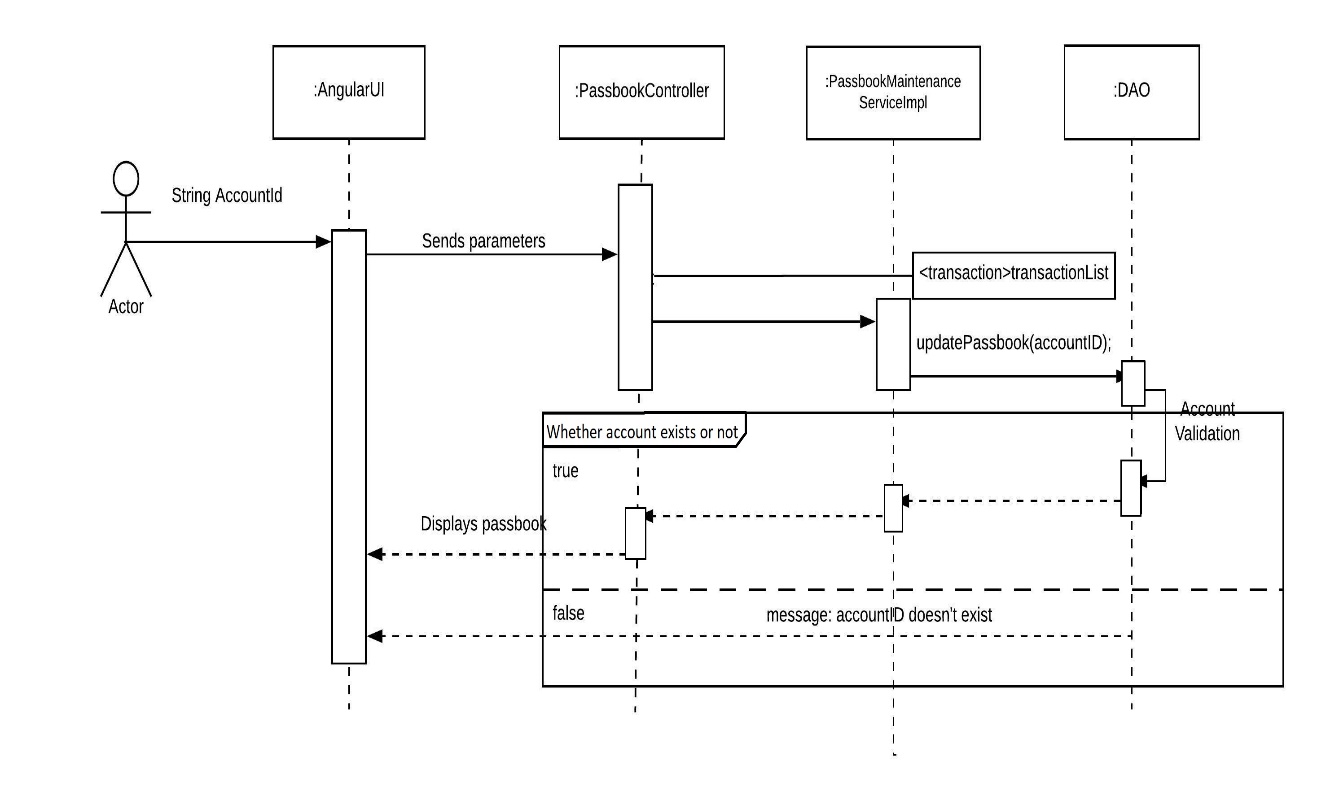
A) Passbook Update

This module is designed to show and print the recent transaction that happened since the passbook was updated the last. It accepts the account Id and corresponding to that it shows the list of transaction made since the last date of passbook updation. The form has proper validation.

Validation

Account Id – Must be a 12 digit number. Accepts only positive numeric value.

Sequence Diagram:



B) Account Summary

This module is designed to display the list of transactions made between two given dates. It accepts account Id, start date and end date for the range of dates one would like to see the transaction list. The form has proper validation.

Validation

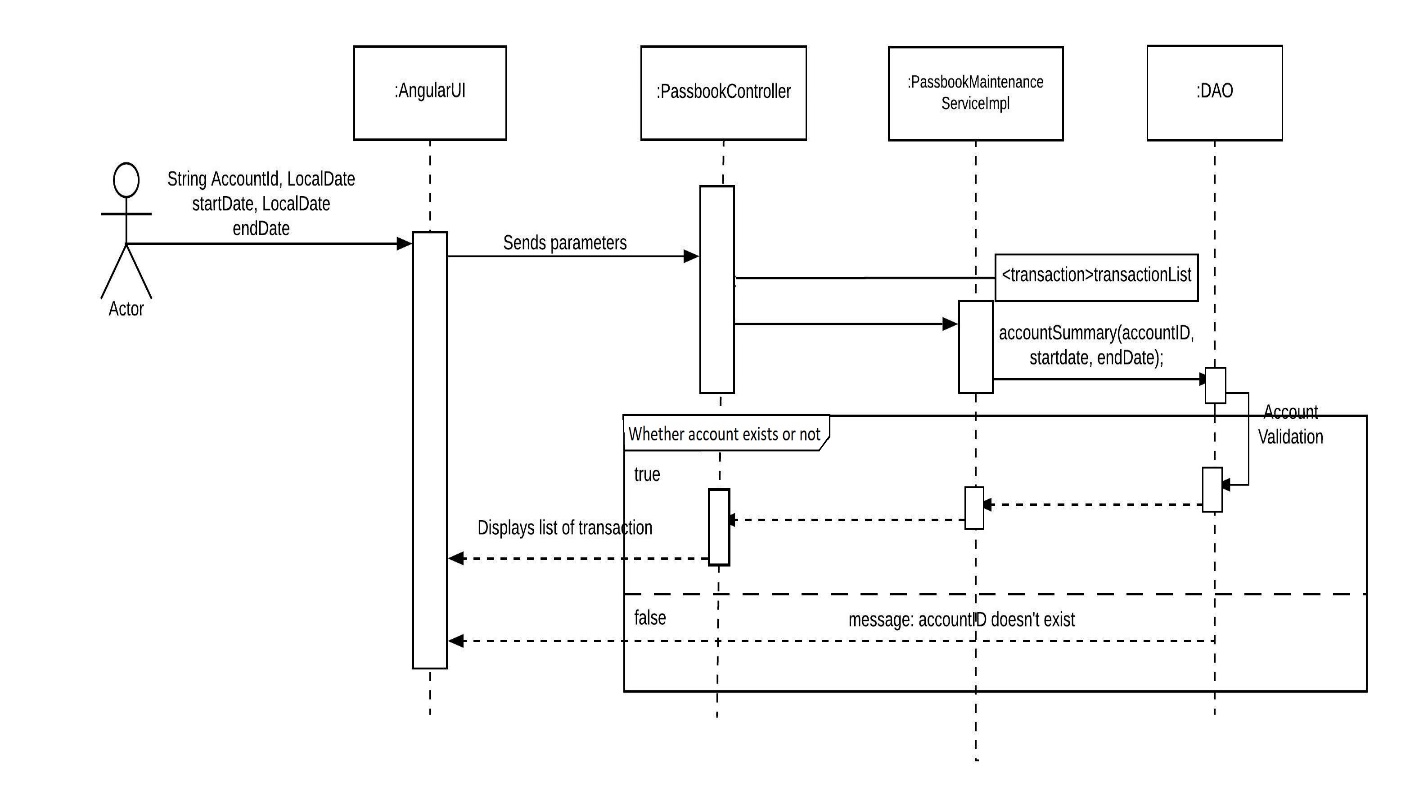
All fields are required.

Account Id – Must be a 12 digit number. Accepts only positive numeric value.

Start date – must be written in given format or chosen from the calendar.

End date- must be written in given format or chosen from the calendar. End date cannot be greater the current date.

Sequence Diagram:



Loan

A) Loan Request

This module is designed to create an entry for Loan Request in Database by adding details like Loan Amount, Tenure, Rate of Interest, Credit Score etc. The form has proper validation.

Validation

All fields are required.

Account Id – Must be a 12 digit number. Accepts only positive numeric value.

Amount - Min amount is 1000 and Max Amount is 100,00,000. Accepts only positive numeric value.

Tenure - Accepts only positive numeric value and must be between 12 and 240. (Months)

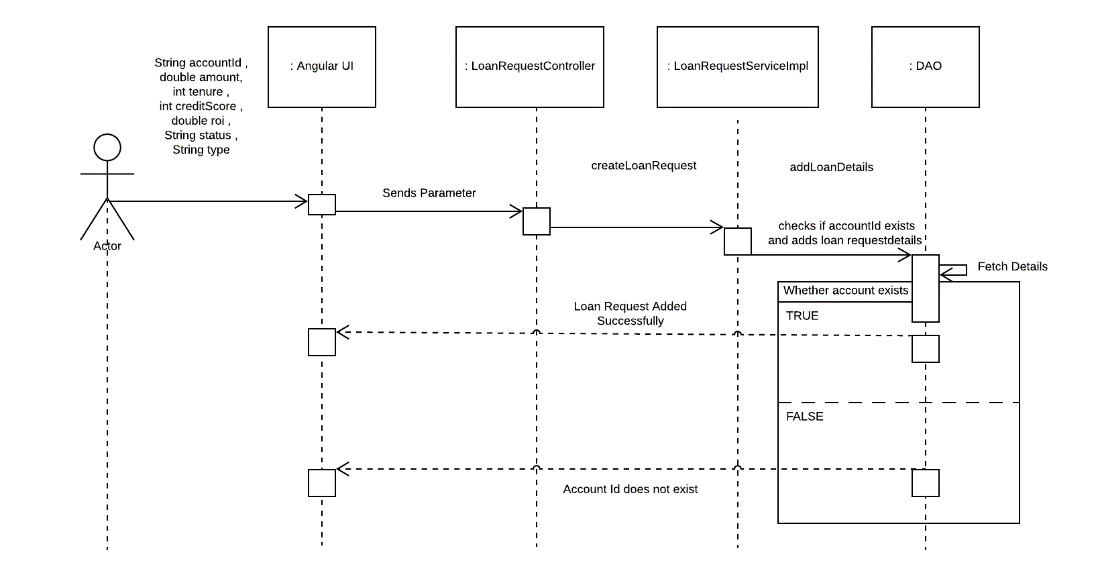
Credit Score – Accepts only positive numeric value must be between 100 & 999

Rate of Interest - Accepts only positive integer value and must be between 4 and 15.

Loan Status - A dropdown is being provided with three entries only out of which one is mandatory to choose.

Loan Type - A dropdown is being provided with three entries only out of which one is mandatory to choose.

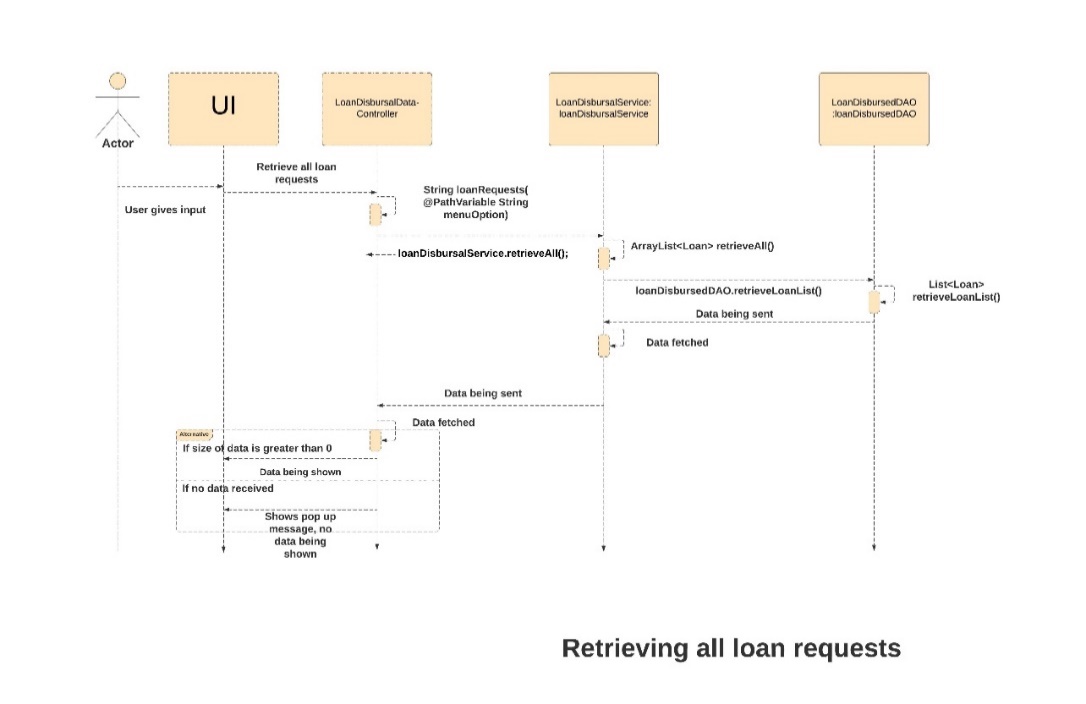
Sequence Diagram:



B) LOAN DISBURSAL

This module is designed to accept and reject loan requests, create a loan disbursed database and update the account balance of the loan customers. Loan is given to those who are the customer of this bank.

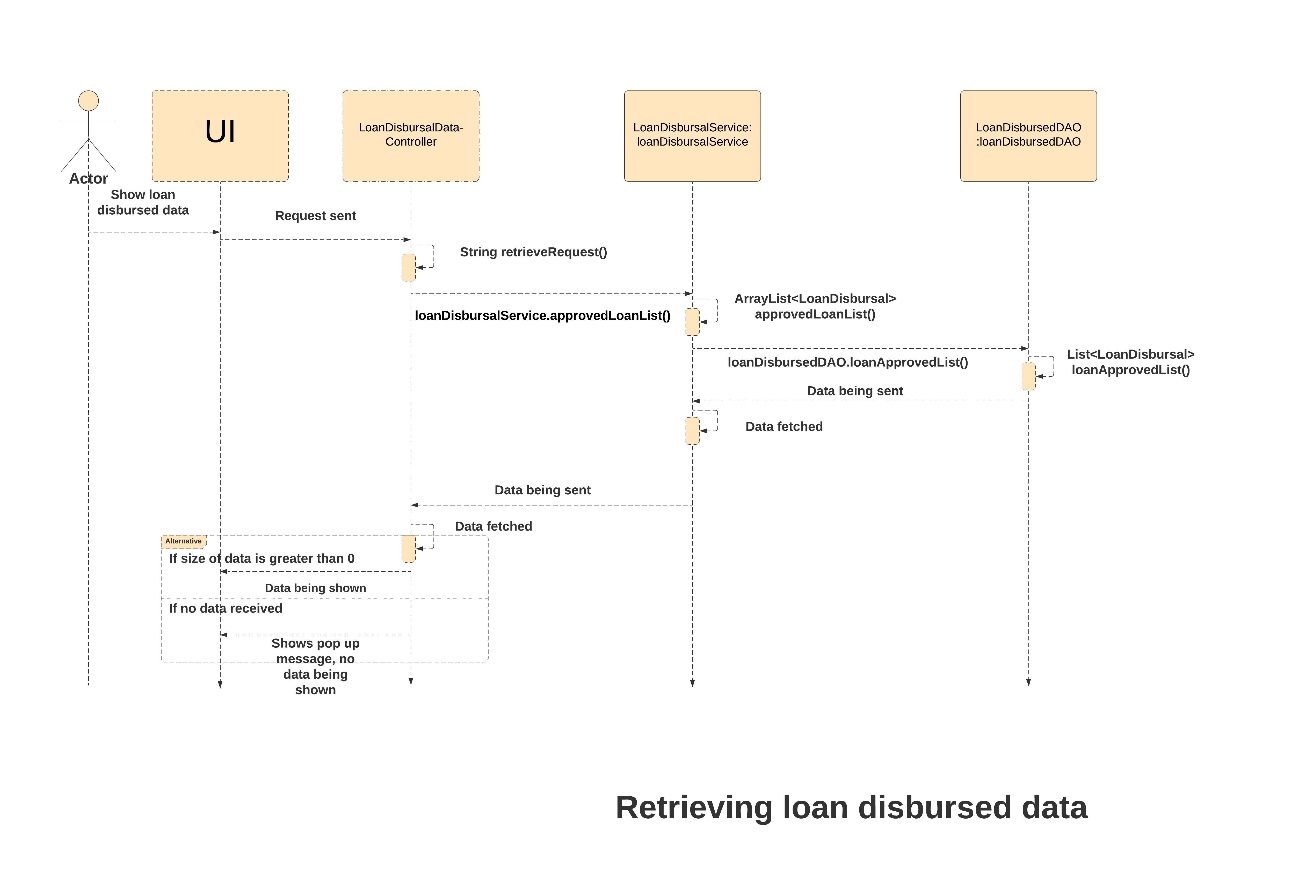
Sequence Diagram:



For accepting and rejecting loan requests.

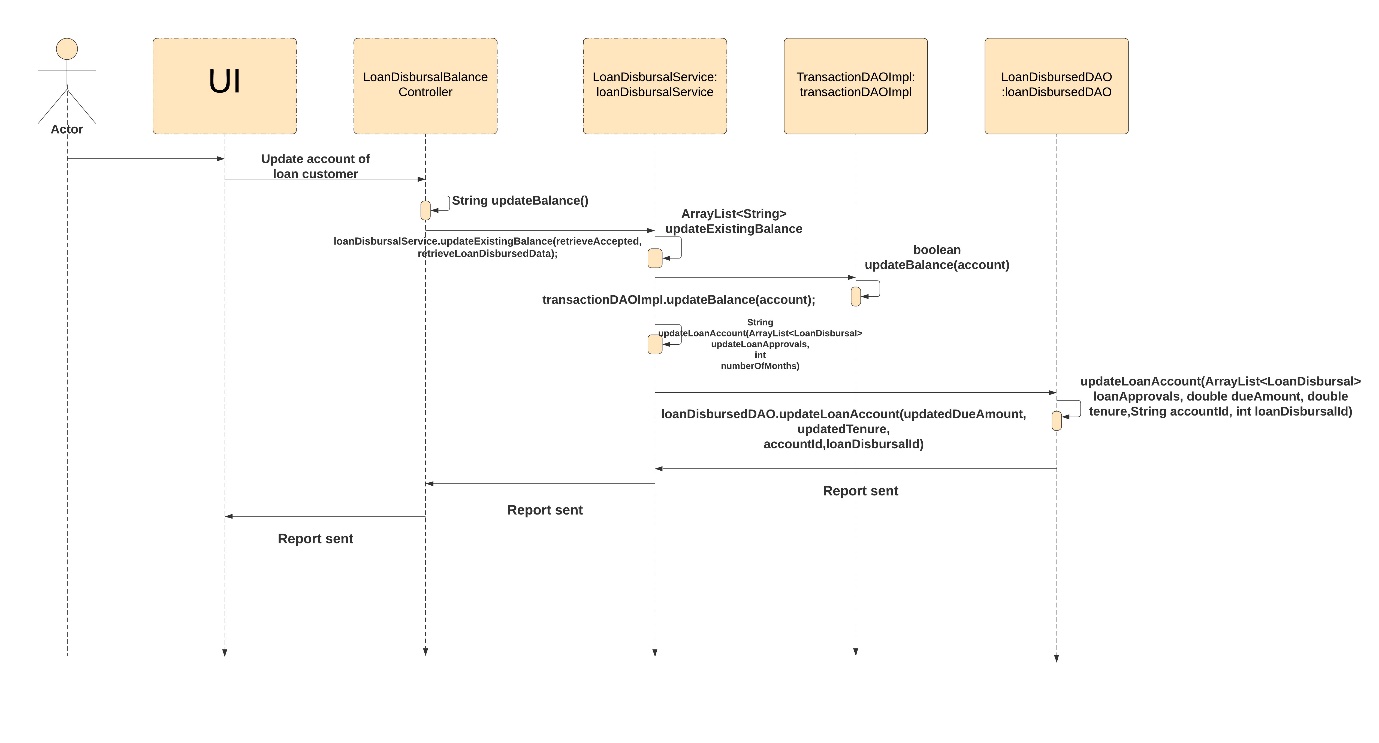
The user can see all the incoming loan requests. Also, the user can accept and reject loan requests by choosing appropriate option in the UI. The above sequence is when the employee chooses the option retrieve all the loan requests. When the user chooses retrieve the accepted loan requests, then first the loan status is changed from pending to accepted. Only those loan requests gets accepted in which the credit score is greater than 670. And then those loan requests are shown. Similarly, when the user chooses show loan requests to be rejected, then first the loan status is changed from pending to rejected. Only those loan requests gets accepted in which the credit score is less than 670.

Sequence Diagram:



Whose loan requests have been accepted, a loan disbursed database is created. When user clicks on Show loan disbursed data, the database is shown.

Sequence Diagram:

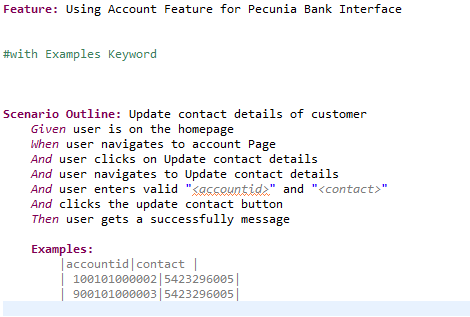


**Update Account Balance**

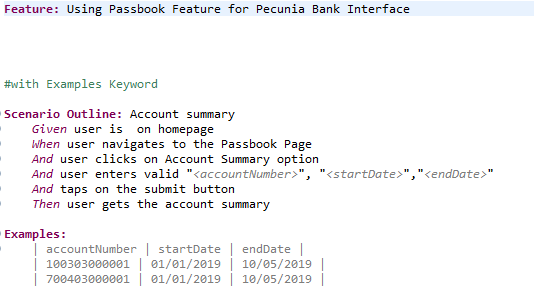
When the user clicks on the third submit button, Update account balance, EMI is debited from the main account balance.

BDD Feature Files

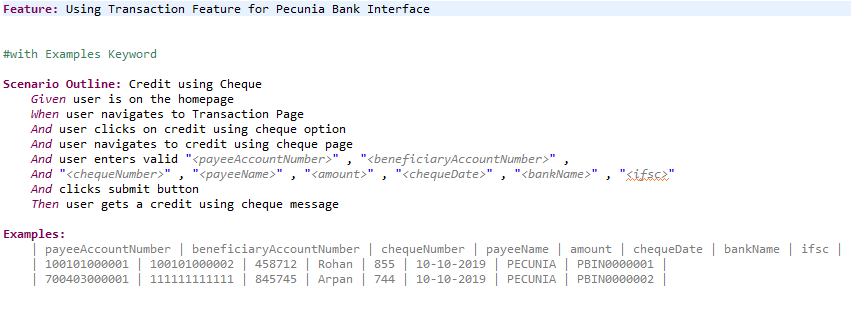
1. Feature file of update account contact details:



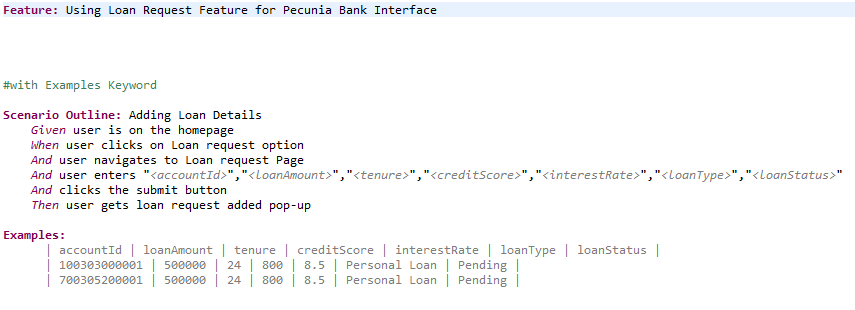
1. Feature file of account summary:



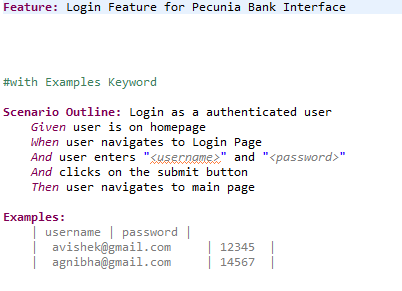
1. Feature file of Credit cheque transaction:



1. Feature file of Loan request:



1. Feature file of login feature:



Conclusion

Project Progress Status (Product Management System)

|  |  |  |  |
| --- | --- | --- | --- |
| Sprint# | Task Assigned | Status | Remark |
| Sprint 1 | Creating UML Diagrams, Defining Test Cases and Sequence Diagrams | Epic and Stories written. Use Case Diagram and Sequence Diagram defined | Update the sequence diagram, include the life block in sequence diagram. Develop Knowledge about micro services architecture |
| Sprint 2 | Implement the test cases using Junit, Implement the modules with core java implementation. Use Java Collection API for data storage (non-persistence) | Modules Implemented using Java. Junit test cases are written and successfully tested.  Sequence Diagrams are modified according to the previous sprint feedback | 3-layer architecture is not properly designed. Write more test case scenario. Properly comment the code. Code Convention is not upto industry standard. Presentation layer is not implemented |
| Sprint 3 | Implement 3-tier architecture. Link the business logic with Database using JDBC connection | Database is designed as per ER  Diagram.  DAO, Service and Presentation layer is properly implanted. All Validations are done in presentation layer.  Code is properly commented. Inline comments and redundant codes are removed. JDBC connections are done.  Logger is being implemented. | All ok |
| Sprint 4 | Design the front end with Html , CSS, Bootstrap | Front end pages are designed with HTML, CSS and Bootstrap. JSP and servlet are being used to connect the front end with backend Java modules. | Pages are not made responsive for mobile. Proper use of bootstrap is missing. Unity in colour scheme is missing.  Additional advise : To make the software more dynamic. Toaster message should be implemented |
| Sprint 5 | Replace the presentation layer with Angular client App and write the BDD test Cases using cucumber. | Angular app is designed. Pages are made mobile responsive according to the previous sprint feedback.  Colour scheme is done uniform across all pages.  Entire project is made dynamic. Toastr messege have been implemented  Jersey as well as Servlet Technology is being explored to link the front-end with backend | Proper Scrum Model is not followed. Every Individual is being asked to assigned a single module. JDBC connections are not closed properly. Proper Documentation is missing. |
| Sprint 6 & Sprint 7 | Replace JDBC connection with JPA Hibernate. Implement Sprint MVC | JDBC is being replaced with JPA Hibernate API. Connection open and close is being managed by session and transaction management of Hibernate. Singleton design is being done for session and transaction management is done using Spring bean. Dependency injection is implemented using Auto wiring. Started working on updated documentation.  We have properly divided the modules. | Proper Documentation required. Add proper validation for product id and product image. Write Spring test cases.  Additional Advise:  To implement audit trail |
| Sprint 8 | Spring Boot Implementation | Documentation Started | NA |