**M31885 SOFTWARE ENGINEERING THEORY AND PRACTICE (SETP)**

**Project Proposal for SETP Main Assignment**

**Group Members: -**

- Nidhin (Leader)  
- Nixon  
- Savio  
- Jun Rong  
- Vincent

The information contained herein is the property of SETP Group 6 and may not be copied, used or disclosed in whole or in part to any third party except with written approval of SETP Group 6 or, if it has been authorised under a contract.

# Project Proposal

## Purpose

The purpose of this Project Proposal is to provide precise requirements definition of the Bank Application software (hereinafter known as “App”). The design and development of the App will be based on the functional specifications defined here.

## Key Objectives



The key objective of this project is to develop a platform 2 stakeholders, the Bank Employees, and the Bank Customer to: -

1. Create and manage accounts for both Bank Employees and Bank Customers.
2. Allow Bank Employees to track Customer’s transactions history.
3. Allow Bank Customers to check their bank balance, transfer money between accounts.
4. Allow Bank Customers to apply for a bank loan.
5. Allow Bank Employee to manage Customer’s loan requests.
6. Allow Bank Customers to pay back their bank loans and view their loan status.

## Main Features/Services



The App shall have these key features/modules: -

* User Management
* Loan Management (Approval/Rejection)
* Loan Due Notification
* Customer home page
* Transaction History
* Request Loan
* Transfer Money to another account
* Payback Loan
* Loan Due Notification

## Assumptions and Dependencies



One Bank Employee must first create the accounts for other Bank Employees as well as Customers with the system before the system can be used.

## User Roles and Access Permissions

|  |  |  |  |
| --- | --- | --- | --- |
| **Module/Feature** | **Remarks** | **App Role** | |
| Bank Employee | Customer |
| User Management | Manage / Register | ✓ | ✖ |
| Loan Management (Approval/Rejection) | Manage | ✓ | ✖ |
| Loan Due Notification | View | ✖ | ✓ |
| Customer Home Page | View | ✖ | ✓ |
| Transaction History | View (according to permission role) | ✓ | ✓ |
| Request Loan | Manage | ✖ | ✓ |
| Transfer Money to another account | Manage | ✓ | ✓ |
| Payback loan | Manage | ✖ | ✓ |

## Resources Required

# PROJECT PLAN: WBS, PERT and Gantt Charts

# SYSTEM REQUIREMENTS SPECIFICATION

## Account Registration (For Bank Employee / Customer)



### There are two (2) types of users available for creation: -

* Bank Employee
* Bank Customer

### Bank Employee can create new Bank Employee accounts that have identical user roles and Bank Customers account.

### Only Bank Employee account users can create other Bank Employees and Bank Customer account.

### The User Management will display the appropriate registration form to the user for registration. Please refer to Appendix A for more details on the registration workflow.

### Bank Customer account is assigned by the Bank Employee.

### System to allow Bank Employee to add new / edit / delete Bank Customers account.

### Please refer to SETP-001: Add User Details for Bank Employee user account creation.

### Please refer to SETP-001: Add User Details for Bank Customer user account creation.

## Bank Employee

### 

## Bank Customer



### 

## Business Rules for User Registration

|  |  |  |
| --- | --- | --- |
| **No.** | Scenario | Business Rule |
| **1** |  |  |
| **2** |  |  |
| **3** |  |  |
| **4** |  |  |

# PROCESS MODEL

# DESIGN OF THE SYSTEM

## Bank Application design from a Use Case Diagram

Diagram

Description automatically generated

### Use Case Specification

|  |  |
| --- | --- |
| **Use Case:** | **Account Management** |
| Actors | Bank Employee |
| Description | An admin user attempts to create a new Bank Employee or Customer account |
| Pre-condition | There must already be 1 existing Bank Employee account |
| Post-condition | A Bank Employee / Customer account is successfully created |
| Data | **Input**: Provide user’s basic credentials and password  **Output**: System assign userID and success/error notification |
| Valid Case | User loads a registration form provided by the web server for the user to input his/her basic credentials and password. User will then submit the form which the server receives. The form’s data is then sent to the account management system and successfully validate the data according to the systems rules. A user account then is created on the database. A success message will be sent back to the user’s interface and displayed to the user. The user interface displayed will depend on if the user is either or Bank Employee or Customer account with the list of functions available. |
| Error Case | **Account already exists**. User loads a registration form provided by the web server for the user to input his/her basic credentials and password. User will then submit the form which the server receives. The form’s data is then sent to the account management system and unsuccessfully validate the data according to the systems rules. A unsuccess message will be sent back to the user’s interface and displayed to the user |

|  |  |
| --- | --- |
| **Use Case:** | **Transaction History** |
| Actors | Bank Employee, Customer |
| Description | A logged in user attempts to view the transaction history of a selected userID |
| Pre-condition | 1. User is logged into their account 2. Only Bank Employees have the rights to view Customers transaction history 3. Customers can only view their own transaction history |
| Post-condition | The transaction history of the selected userID is displayed |
| Data | **Input**: transactionID  **Output**: transactionID log description |
| Valid Case | User clicks on the Transaction History button. This sends a route request to the web server which calls on the transactions manager. The manager then queries the transactions record within the database and successfully validates the data from the selected userID. The manager then retrieves and sends all the transactions data for the selected userID and is displayed to the user interface. |
| Error Case | **No transactions found**. User clicks on the Transaction History button. This sends a route request to the web server which calls on the transactions manager. The manager then queries the transactions record within the database and unsuccessfully validates that there is no data from the selected userID. The manager then sends an invalid route message back to user interface. |

|  |  |
| --- | --- |
| **Use Case:** | **Transfer Money** |
| Actors | Customer |
| Description | A logged in user attempts to transfer money to another existing bank account |
| Pre-condition | 1. User is logged into their account 2. At least 2 customer account must already be in the database 3. The same userID cannot transfer money to ownself 4. User must know the other userID to make a transfer 5. Transfer amount cannot be more than current balance |
| Post-condition | The money is transferred from updates the User’s bank balance |
| Data | **Input**: userID, transferAmt  **Output**: Show bank balance |
| Valid Case | User clicks on the Transfer Money button. User loads a form provided by the web server. The user inputs another existing userID and transferAmt. User then submits the form which the web server will receive. This form data is then sent to the users manager and validates successfully that there is existing userID within the database and there is sufficient amount for the transfer.  The manager then queries the user record within the database, updates the bankBalance of both Users. The interface then loads the Customer interface page. |
| Error Case | **No userID found**. User clicks on the Transfer Money button. User loads a form provided by the web server. The user inputs another existing userID and transferAmt. User then submits the form which the web server will receive. This form data is then sent to the users manager and validates unsuccessfully that there is no existing userID within the database. An unsuccessful validation error message is sent back to the customer interface and displayed to the User.  **Insufficient funds for transfer**. User clicks on the Transfer Money button. User loads a form provided by the web server. The user inputs another existing userID and transferAmt. User then submits the form which the web server will receive. This form data is then sent to the users manager and validates unsuccessfully that there is insufficient funds within the user database. An unsuccessful validation error message is sent back to the customer interface and displayed to the User. |

|  |  |
| --- | --- |
| **Use Case:** | **Check Balance** |
| Actors | Bank Employee, Customer |
| Description | A logged in user attempts to view the bank balance of a selected userID |
| Pre-condition | 1. User is logged into their account 2. Only Bank Employees have the rights to view Customers account 3. Customers can only view their own bank balance |
| Post-condition | The bank balance of the selected userID is displayed |
| Data | **Input**: userID, bankBalance  **Output**: Show bank balance |
| Valid Case | User clicks on the Check Balance button. This sends a route request to the web server which calls on the users manager. The manager then queries the users record within the database and successfully validates the data from the selected userID. The manager then retrieves and sends the bank balance data for the selected userID and is displayed to the user interface. |
| Error Case | **No records found**. User clicks on the Check Balance button. This sends a route request to the web server which calls on the users manager. The manager then queries the users record within the database and unsuccessfully validates that there is no data from the selected userID. The manager then sends an invalid route message back to user interface. |

|  |  |
| --- | --- |
| **Use Case:** | **Request Loan** |
| Actors | Customer |
| Description | A logged in Customer account attempts to apply for a loan account |
| Pre-condition | There must be at least 1 Bank Employee and 1 Customer account |
| Post-condition | Request is submitted and awaiting approval/rejection by Bank Employee |
| Data | **Input**: userID, loanAmt and supporting documents  **Output**: Request is successfully submitted |
| Valid Case | User clicks on the Request Loan button. User loads a form provided by the web server. The user inputs another existing userID and all the mandatory supporting documents for the loan request. User then submits the form which the web server will receive. This form data is then sent to the load manager and validates successfully that that all required data are provided for processing. A success message will be sent back to the user’s interface and displayed to the user that their request is pending for approval. |
| Error Case | **Do not meet loan requirements**. User clicks on the Request Loan button. User loads a form provided by the web server. The user inputs another existing userID and all the mandatory supporting documents for the loan request. User then submits the form which the web server will receive. This form data is then sent to the load manager and validates unsuccessfully that that the user does not meet the loan requirement. An unsuccessful message will be sent back to the user’s interface and displayed to the user that their request is pending for approval. |

|  |  |
| --- | --- |
| **Use Case:** | **Approve/Reject Loan** |
| Actors | Bank Employee |
| Description | A logged in Bank Employee account attempts to approve/reject a submitted loan request by a Customer. |
| Pre-condition | 1. Only Bank Employee can approve/reject loan requests 2. There must be at least 1 Bank Employee and 1 Customer account 3. There must be a submitted loan request |
| Post-condition | Request is either approved or rejected by Bank Employee |
| Data | **Input**: userID, loanID, and supporting documents  **Output**: Approval / Rejection |
| Valid Case | User clicks on the View Loan button. This sends a route request to the web server which calls on the loans manager. The manager then queries the loans record within the database and successfully validates the data from the selected loanID. The manager then retrieves and sends the loan balance data for the selected loanID and is displayed to the user interface with an Approve and Reject selection.   **Approve**: User clicks on the Approve button. A loan account then is created on the loan database. A success message will be sent back to the Customers user’s interface and displayed to the user. |
| Error Case | **Reject**: User clicks on the Reject button. User clicks on the Approve button. A reject message will be sent back to the Customers user’s interface and displayed to the user. |

|  |  |
| --- | --- |
| **Use Case:** | **Check Loan** |
| Actors | Bank Employee, Customer |
| Description | A logged in user attempts to view the loan balance of a selected userID |
| Pre-condition | 1. Customer is logged into their approved loan account 2. Only Bank Employees have the rights to view Customers account 3. Customers can only view their own loan balance |
| Post-condition | The loan balance of the selected userID is displayed |
| Data | **Input**: userID, loanBalance  **Output**: Show outstanding loan balance |
| Valid Case | User clicks on the Check Loan button. This sends a route request to the web server which calls on the loans manager. The manager then queries the loans record within the database and successfully validates the data from the selected userID. The manager then retrieves and sends the loan balance data for the selected userID and is displayed to the user interface. |
| Error Case | **No loan records found**. User clicks on the Check Loan button. This sends a route request to the web server which calls on the loans manager. The manager then queries the loans record within the database and unsuccessfully validates that there is no data from the selected userID. The manager then sends an invalid route message back to user interface. |

|  |  |
| --- | --- |
| **Use Case:** | **Payback Loan** |
| Actors | Customer |
| Description | A logged in user attempts to payback their outstanding loan balance. |
| Pre-condition | 1. Customer is logged into their approved loan account 2. Customer’s must at least have $0.01 in their bank account 3. Customer cannot pay back any amount more than their current bank balance in their account. 4. Only Customer can pay back their loan |
| Post-condition | The loanBalance is updated on the user interface. |
| Data | **Input**: userID, loanID, payLoan  **Output**: Update and display outstanding loan balance amount. |
| Valid Case | User clicks on the Payback Loan button. This sends a route request to the web server which calls on the users manager. The manager then queries the users record within the database and successfully validates that the User at least have $0.01 to pay the loan.  A form is loaded, provided by the web server. The user inputs the desired amount to pay back no more than the User’s existing bank balance. User then submits the form which the web server receives. This form data is then sent to the loans manager and validates that there is an existing loanID for the User. The manager then queries the loan record within the database, adds and updates loanBalance based on the submitted amount for the User.  The manager retrieves updated loanBalance and is displayed to the user interface. |
| Error Case | **Insufficient amount selected**. User clicks on the Payback Loan button. This sends a route request to the web server which calls on the users manager. The manager then queries the users record within the database and unsuccessfully validates that the User do not have any money to pay the loan. The manager then sends an invalid route message back to user interface.  **Insufficient amount selected**. User clicks on the Payback Loan button. This sends a route request to the web server which calls on the users manager. The manager then queries the users record within the database and unsuccessfully validates that the input amount is lesser than the User’s bank balance amount. The manager then sends an invalid route message back to user interface. |

## Bank Application design from a Class Diagram

## Bank Application design from a MVC

# QUALITY ASSURANCE PLAN

# IMPLEMENTATION AND TESTING