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

**Project Proposal for SETP Main Assignment**

**Group Members: -**

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

**Coding Team**- Nidhin (Leader)  
- Nixon  
- Jun Rong

**Documentation Team**  
- Aiju  
- Savio  
- 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, 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 transaction history.
3. Allow Bank Customers to check their bank balance, and 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



## 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 | ✖ | ✓ |

# PROJECT PLAN: WBS, PERT and Gantt Charts

**Work Breakdown Structure (WBS)**

The work break structure of the project is shown below. WBS is a tool that helps to take a step-by-step approach by breaking down the project into several steps.

Graphical user interface

Description automatically generated with medium confidence

WBS can be created phase-based or product based. Here a product-oriented WBS is created. The bank application has been broken down into four major tasks and those tasks are further divided into some functions.

**Gantt Charts**

**Program Evaluation Review Technique** (PERT)

A screenshot of a computer

Description automatically generated with medium confidence

# SYSTEM REQUIREMENTS SPECIFICATION

## Introduction

The software requirement specification (SRS) introduction is the overview of the entire SRS with purpose, scope, definition, and references. The objective of the SRS is to address the problem statement which is to develop a banking web application. An in-depth analysis of the features included and requirements from all the stakeholders are discussed in the SRS. Web application specifications and requirements are gathered and listed down in the document.

## Project Propose

The document presents valuable thoughts and inputs from the team members during the team meeting regarding the web application and its requirements. Information gathered through team discussion on the requirements of the banking application and specifics of the requirement are discussed in the document. SRS documents illustrate a detailed overview of the product, specifications, and desired outcome.

As we moved away from physical money into the realm of virtual money, ATM machines are less convenient when we can use internet banking for the same features at our convenience. This document addresses the specifications of the features required for the web application. Important priorities will be: -

* Build a banking system that is easily accessible to the customer from any location.
* Minimize the traffic flow towards the bank.
* Easy to use interface for the application to improve customer relationships.

## Project Scope

The scope of the project is to implement a banking web application. The focus of the project will be on the company, stakeholders, and web application which enables the online user to register an account, check balances make online transactions, and loan applications.

* This application intends to provide better service to the customer.
* The application can be used by any bank to improve their service.
* This application helps customers to check all the information about their account from anywhere.
* Bank employees can check the account present in the bank.

## Glossary and Reference

## Intended Audience and Suggestions

The different types of readers are: -

1. Customers.

* Project Scope
* Security available

1. Employees.
2. Developers.

* Project Scope
* Use Case Module

## Overall Description

The web application interface enables the client to interact with the banking system. The application system will be the web page of the banking application. The client will have a login page to input the login particulars of the user. If the log-in details are correct, then the user will be redirected to the home page. Homepages consist of features such as check balance, transfer money, request loan, and transaction history. The client can access the features from the home page and will be able to use the features accordingly. The employee login interface will be different as bank employees can register or open a new account from the home page. Bank Employees will be able to access the features clients expect for the transfer of money and pay back their loan.

## User Features

The internet banking application includes the following modules: -

* + Log in page - Module permits validated customers access to the bank services.
  + Balance enquiry - Module maintains the accounts balance details.
  + Update profile - The module allows the customer to update their particulars
  + Fund Transfer - The module allows the customer to make fund transfer within the same bank
  + Transaction History - The module helps to view the history of transactions made.
  + Apply for Loan - The module allows to apply for a loan

## ****User Classes and Characteristics****

**Customers**: The customer will have a general account. Account features permit customers to view account balance, and transaction history. Other features such as fund transfer and apply for loan options will be accessible to the customer.

**Employee**: Employee will have additional features such as account creation, approve or reject the loan.

## ****Operating environment****

**Server-side**

* Hard drive - More than 150 GB
* RAM – More than 1GB
* Processor – Pentium 4 or higher

**Client-side**

* + Hard Drive – More than 150GB
  + RAM – More than 1GB
  + Processor – Pentium 4 or higher

## ****Design and Implementation Constraints****

Only one server will be used for the system.

Server is assigned at ….

The language used for the application is Java..

System is limited to HTTP/HTTPS protocols

Need to add in …

## Assumptions and Dependencies

**Assumption**

* Customer details such as username, password, account type should be entered by the employee into the system. Customer should be familiar using computer and internet browsing.
* User should have to have basic knowledge in English to use the banking application.
* Customer should be familiar of banking system. The web page link is available to customer.
* 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.

**Dependencies**

## User Documentation

The registered user will have following features:

* Bank account
* Check balance
* Fund transfer
* Click apply for loan
* Transaction history records.

## Features and Requirements

This segment documents the entire spectrum of functional and non-functional requirements identified in the project. Functional requirements will be listed in according to the relationship software application, bank employees and customers perspective. The non-functional requirements that are related to the accessibility, confidentiality, integrity, safety, usability, security, availability, efficiency, reliability, and suitability will be presented.

## Functional Requirements

This sub segment presents the basic functional requirements identified for this Bank Application project. Initially very generic requirements that contains to the entire system is provided.

The appended table **XX** below shows the identified functional basic requirements that pertain to the entire Bank Application Project.

**Basic Functions**

|  |  |
| --- | --- |
| **Requirements** | **Description** |
| A1 | A server is used to host the Bank Application and provide the system processing and storage capabilities. |
| A2 | A computer to provide a bank employee with all the different functionalities in the application. |
| A3 | A computer to provide a customer with all the different functionalities in the application. |

**Bank Employee Functions**

|  |  |
| --- | --- |
| **Requirements** | **Description** |
| B1 | Creating a new account for another bank employee |
| B2 | Opening a Customer bank account. |
| B3 | Allow Bank Employees to track Customer’s transactions history. |
| B4 | Allow Bank Employee to manage Customer’s loan requests. |
| B5 | Allow Bank Employee to manage Customer’s loan status |

**Customer Functions**

|  |  |
| --- | --- |
| **Requirements** | **Description** |
| C1 | Allow Bank Customer to check their bank balance, transfer money between accounts. |
| C2 | Allow Bank Customer to apply for a bank loan. |
| C3 | A Bank Customer is not allowed to get a loan unless gotten approval from the Bank Employee |
| C4 | Allow Bank Customer to pay back their bank loans and view their loan status. |

## Non- Functional Requirements

This sub segment presents the non-functional requirements identified for this Bank Application project. The different sub classes of the non-functional requirements are **Reliability**, **Capacity, Security, Manageability, Data Integrity, and Usability** respectively.

**Reliability**

|  |  |
| --- | --- |
| **Requirements** | **Description** |
| D1 | The system should be easily accessible by both Bank Employees and Customers to access the Bank Application’s features 24/7 |
| D2 | The users can consistently perform the intended functions without failure |

**Capacity**

|  |  |
| --- | --- |
| **Requirements** | **Description** |
| E1 | Up to a 100 users accounts can be stored in the system |

**Security**

|  |  |
| --- | --- |
| **Requirements** | **Description** |
| F1 | A user must have a created account to access the Bank Application. |
| F2 | The account is required to input a id and password to access the Bank Application’s functionalities. |
| F3 | Only one user shall into one device at any given point of time. |

**Manageability**

|  |  |
| --- | --- |
| **Requirements** | **Description** |
| G1 | During the creation of new accounts or the use other service features, the the Bank Application stays up and running. |

**Data Integrity**

|  |  |
| --- | --- |
| **Requirements** | **Description** |
| H1 | The Bank Application maintains will keep periodic backups to the database for every record transaction once every day. |

## Budget

|  |  |  |
| --- | --- | --- |
| **S/N** | **Activities** | **No. of Man-Days** |
| **Project Planning** | | |
| 1 | * Requirements Gathering * Analysis & Design * Impact Assessment | 2.0 |
| **Development** | | |
| 2 | * Database creation * Linking of tables and logic * Developing the User Interface | 3.0 |
| **System Integrated Testing** | | |
| 3 | * Perform functional and non-functional requirements testing * Perform Quality Assurance (QA) testing | 2.0 |
| **User Acceptance Testing** | | |
| 4 | * Preparation of User Acceptance Testing (UAT) environment and data * Handle queries from user on UAT related issues * UAT to be scheduled accordingly upon completion of System Integration Testing (SIT) | 2.0 |
| **Production Implementation** | | |
| 5 | * Packing of code for deployment * Migration of package to Production Stage | 2.5 |
| **Documentation** | | |
| 6 | * Functional specification * Create Test cases | 2.0 |
| **Total** | | **13.5** |
| **Estimated Effort Required: 13.5 days**  **Man-days Rate: $700.00**  **Total Cost: S$ 9450.00** | | |

## 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

The software process model illustrates the order in which the activities of the software development activities will be performed. The process model aims to give guidance for controlling and coordinating the task to achieve the product. Different types of process models are there such as: -

* V model
* Waterfall
* Incremental model
* RAD model
* Agile model
* Iterative model
* Spiral model.
* Prototype model

Each of the above-mentioned process models has its advantages and disadvantages. The decision of choosing the process model is incredibly important and crucial to the development process. Important factors to consider while choosing the process model are the complexity of the project, project size, delay cost, involvement of customers, knowledge of the technology, and availability of resources. Many other variables need to be considered while choosing a process model but above mentioned are the top priority list of items we need to consider.

**Agile Model**

Agile simply means swift or versatile. It is an approach based on delivering requirements incrementally throughout the product cycle and delivering requirements iteratively. The agile model prioritizes central values and behaviours of trust, flexibility, collaboration, and empowerment. With the agile method tasks are divided into short phases of work and adaptations of plans and frequent reassessment. Iteration/tasks are considered as the short time frame.

Widely used project management methodologies are compact and tight while Agile is flexible. If we use agile methodology, it permits the requirement changes even after the execution of project which is not possible if we choose waterfall method. Agile models have the highest customer involvement in development life cycle, but other models don’t have that amount of customer involvement in development cycle. Agile model involves the participation of customer phase in each development phase. Generally used project management methodology such as waterfall model, management of scope of project, personal, quality assurance, status reporting to stakeholders, risk management and all the responsibilities will be rest on the project manager’s shoulder. In the Agile project management model divides these big responsibilities into three agile roles:

* Product owner

The responsibility of product owner is to set up project goals, manage schedule against the scope, project requirement changes and prioritizing the product feature.

* Scrum

Main scope for Scrum master to help manage the team and resolution of problems while executing the task.

* Team member

Team members work on activities assigned to them and report to Scrum master.

Some of the benefits of the agile are higher customer satisfaction, improved product quality, morality of the team will be high, collaboration and ownership will be high, improved visibility of performance, reduced risk, and better project control. Agile method allows us to make frequent changes for the banking application development. We do have the resources and team available to split the task and execute accordingly. Our size of the project is smaller and frequent discussions and communications can be made to complete project. Hence, we decided to use Agile as the 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