Requirements Engineering

Polaris Bank

Submitted By: Jason Hanafin (T00207878)

Computing with Software Development

Date Submitted: 06/12/2019

**File of Contents**

[1. Introduction/overview 3](#_Toc25309610)

[2. Functional Components 4](#_Toc25309611)

[3. User Requirements 5](#_Toc25309612)

[3.1. BankSYS will manage Accounts 5](#_Toc25309613)

[3.2. BankSYS will manage staff 5](#_Toc25309614)

[3.3. BankSYS will manage Transactions 5](#_Toc25309615)

[3.4. BankSYS will Generate reports 5](#_Toc25309616)

[4. System Requirements 6](#_Toc25309617)

[4.1. System Level Use Case Diagram 7](#_Toc25309618)

[4.2. Manage Accounts 8](#_Toc25309619)

[4.2.1. Add Accounts 8](#_Toc25309620)

[4.2.2. Update Accounts 10](#_Toc25309621)

[4.2.3. Close Accounts 12](#_Toc25309622)

[5.1. Manage ??? 14](#_Toc25309623)

[5.1.1. ??? 14](#_Toc25309624)

[5.1.2. ??? 16](#_Toc25309625)

[5.1.3. ??? 18](#_Toc25309626)

[5.2. Manage Transaction 20](#_Toc25309627)

[5.2.1. Deposit Transaction 20](#_Toc25309628)

[5.2.2. Withdraw Transaction 23](#_Toc25309629)

[5.2.3. Transfer Transaction 23](#_Toc25309629)

[5.3. Generate reports 25](#_Toc25309630)

[5.3.1. View Customer Signup 25](#_Toc25309631)

[5.3.2. Monthly Transaction Report 27](#_Toc25309632)

[5. System Model 29](#_Toc25309633)

[5.1. Level-0 DFD 29](#_Toc25309634)

[5.2. Level-1 DFD 30](#_Toc25309635)

[5.3. Level-2 DFD (Process P1: Manage Accounts) 31](#_Toc25309636)

[5.4. Level-2 DFD (Process P2: Manage Staff) 32](#_Toc25309637)

[5.5. Level-2 DFD (Process P3: Manager Transactions) 34](#_Toc25309638)

[5.5. Level-2 DFD (Process P4: Generate Reports) 34](#_Toc25309638)

[6. Data Model (Class Diagram) 35](#_Toc25309639)

[6.1. Class Diagram 3](#_Toc25309640)8

[6.2. Relational Schema 3](#_Toc25309641)9

[6.3. Database Schema 40](#_Toc25309642)

[7. Conclusion 4](#_Toc25309643)2

# Introduction/overview

This document provides guidelines to build a banking system that provides functions for staff management, transaction management, account management and Reporting. The system will have four processes: Manage Transactions, Manage Account, Manage staff and Reports. Each will have its own functions.

Manage staff will have three functions: Add Staff, Update Staff and Remove Staff. Add Staff will allow the Admin to add a new staff member to the system. Update Staff will allow the Admin to update an existing staff member. Remove Staff will set the permissions of an existing staff member to ‘0’ so that they can no longer provide services.

Manage Transactions will have three functions: Deposit Transaction, Transfer Transaction and Withdraw Transaction. Deposit Transaction will allow the user to deposit money into an account on the system. Transfer Transaction will allow the user to Transfer money between accounts on the system. Withdraw Transaction will allow the user to withdraw money into an account on the system.

Manage Account will have three functions: Add Account, Modify Account information and Close Account. Add Account will allow the user to add an account into to the system. Modify Account information will allow a user to modify their account information into the system and Close Account will lock an account and prevent modification and money transfer.

Administrative Reporting has two functions: View Total Transactions and View Customer signup. View Total Transactions shows the Total Transactions for a specific month in a specific year. View Customer signup shows the number of signups for a specific month in a specific year.

This document will go through how each function will work using a Level 0 DFD, Level 1 DFD and three Level 2 DFD’s, one for each process.

There will be a class diagram for the data models.

There is a relational schema and database schema for the creation of a database for the system.

# Functional Components

This section outlines the functional components of the propsoed software system.

There are 4 main componets in the proposed system. The Customer can make a Withdrawl, Deposit and a Transfer Transaction. The manager can add staff, update staff and remove staff. The Staff can also add an Account, remove an Account and update an Account.

# User Requirements

This section describes the user requirements as high-level abstract statements.

## BankSYS will manage Customer

* + 1. BankSYS Register Customer
    2. BankSYS Update Customer.

## BankSYS will manage Accounts

* + 1. BankSYS will Create Account.
    2. BankSYS will Update Account.
    3. BankSYS will Close Account.

## BankSYS will manage Transactions

* + 1. BankSYS will Deposit money
    2. BankSYS will Withdraw money
    3. BankSYS will Transfer money

## BankSYS will Generate reports

* + 1. BankSYS will produce a Bank Statement report.
    2. BankSYS will Create a Transaction report.

# System Requirements

BankSYS consists of four modules, Account, Transactions, Staff and Reports.

The user requirements specified in section 3 are presented as high-level system requirements in this section of the document.

## System Level Use Case Diagram

The following system level use case diagram illustrates the high-level system requirements.

Customer

## Manage Customer

### Register Customer

This component lets a Customer Register with the bank

Customer

<<Include>>

<<Extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Register Customer** | |
| **Use Case Id** | BS001 | |
| **Priority** | 1 | |
| **Source** | Manager | |
| **Primary Business Actor** | Customer | |
| **Other Participating Actors** |  | |
| **Description** | This function registers a Customer to the System. | |
| **Preconditions** | The PPS Number must not be registered to another user | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** Customer invokes the Register Customer function.  **Step 3:** The Customer enters the required data:   * Personal Access Code(PAC) * PPS Number   **Step 7:** The User enters the required data:   * FirstName * LastName * DateOfBirth * CountryCode * PhoneNo * AddressLine1 * AddressLine2 * AddressLine3 * Town * County * Eircode | **Step 2:** The system displays the RegisterLoginData UI  **Step 4:** The system validates the data entered:   * All fields are mandatory * PAC must be numeric * PPS Number must be formatted correctly   **Step 5:** The system checks if the PPS Number is already registered to a Customer  **Step 6:** The system displays the RegisterUserData UI  **Step 8:** The system validates the data entered:   * All fields except AddressLine2 and AddressLine3 are mandatory * FirstName, LastName must be Alphabetic * DateOfBirth must be a Date * CountryCode and County must be selected * Phone number must be numeric * AddressLine1, AddressLine2, AddressLine3 must be alphanumeric * Town must be Alphabetic * Eircode must be in the correct format   **Step 9:** The system assigns and formats the next CustomerId to 8 digits  **Step 10:** Sets status to Active (“A”)  **Step 11:** The system gets the current assigns the Current date as DateCreated  **Step 12:** The System saves the Customers details in the Customer File:   * CustomerId * FirstName * LastName * PPSNo * DateOfBirth * AddressLine1 * AddressLine2 * AddressLine3 * Town * County * Eircode * DateCreated   **Step 13:** The system assigns and formats the next LoginId to 8 digits  **Step 14:** The System saves the Login details in the Login File:   * LoginId * CustomerId * PAC   **Step 15:** The system displays a confirmation message  **Step 16:** The system Displays the Start Screen UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Entered** |  | **Step 4:** The customers Data doesn’t validate  **Step 5:** The system displays an appropriate error message.  **Step 6:** Return to Step 3  **Step 5:** The customers PPS Number is already in use  **Step 6:** The system displays an appropriate error message.  **Step 7:** Return to Step 3  **Step 8:** The customers Data doesn’t validate  **Step 9:** The system displays an appropriate error message.  **Step:10:** Return to Step 7 |
| **Conclusions** | The Customer gets recorded in the database. | |
| **Post conditions** | The Customer can Create an Account | |
| **Business Rules** | The PPS Number must not already exist in the database. | |
| **Implementation Constraints** |  | |

### Update Customer Data

This component lets the Customer update their Data

Customer

<<Include>>

<<Extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Update Account** | |
| **Use Case Id** | BS002 | |
| **Priority** | 9 | |
| **Source** | Customer | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function updates Customer Data. | |
| **Preconditions** | The Customer must exist on the system. | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** Customer invokes the Update Customer Data function.  **Step 3:** The Staff selects the Account they wish to update.  **Step 6:** The Staff enters the details of the Account they wish to edit. | **Step 2:** The system retrieves the ID, Full name and DateOfBirth for all Accounts from the Accounts File.  **Step 2:** The system displays the Select Account UI.  **Step 4:** The system retrieves the full details of that Account from the Account File.  **Step 5:** The system displays the full details using the Edit Details UI.  **Step 7:** The system validates the Modified data:  **Step 8:** The system displays a confirmation message  **Step 9:** The system resets the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid data entered** |  | **Step 7:** The system fails to update the Account details in the File  **Step 8:** The system displays an appropriate error message  **Step 9:** Return to Step 6 |
| **Conclusions** | The Account details that have been modified update in the system. | |
| **Post conditions** | The updated Account details can now be viewed from the File. | |
| **Business Rules** | The Account must exist and not be “Closed” in the File. | |
| **Implementation Constraints** |  | |

## Manage Accounts

### Create Account

This component lets the Customer create an Account.

Customer

<<Include>>

<<Extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Create Account** | |
| **Use Case Id** | BS003 | |
| **Priority** | 2 | |
| **Source** | Customer | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function Creates an account for the Customer | |
| **Preconditions** | The staff member must not already exist on the system. | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** Manager invokes the add staff function.  **Step 4:** The manager enters the required data:   * FirstName (varchar 20) * LastName (varchar 20) * MobileNo (varchar 10) * Email (varchar 50) * Iban (varchar 22) * PPSNo (varchar 9) * PayRate (Decimal 5) * AddressLine1 (varchar 20) * AddressLine2 (varchar 20) * AddressLine3 (varchar 20) * Town (varchar 20) * County (varchar 20) * Eircode (varchar 7) * DateStarted (Date) * DateLeft (Date) | **Step 2:** The system assigns the next StaffID  **Step 3**: The system displays the Add Staff UI  **Step 5:** The system validates the data entered:   * All fields must be entered except DateLeft * FirstName and LastName must be alphabetical * MobileNo must be numeric * PPSNo must be alphanumeric * PPSNo must not already exist * PayRate must be numeric and greater than zero * AddressLine1, AddressLine2, AddressLine3 must be alphanumeric * Town ,County must be Alphabetic * Eircode must be alphanumeric * DateStarted must be a Date * DateLeft must be a Date   **Step 6:** The systems sets staff status to (“E”) “Employed”  **Step 7:** The System adds the staff details in the Staff File:   * StaffID * FirstName * LastName * MobileNo * Email * Iban * PPSNo * PayRate * AddressLine1 * AddressLine2 * AddressLine3 * Town * County * Eircode * DateStarted * Status (“E”) * DateLeft   **Step 8:** The system displays a confirmation message  **Step 9:** The system resets the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Data Entered** |  | **Step 5:** Invalid data detected  **Step 6:** The system displays an appropriate error message  **Step 7:** Return to Step 4 |
| **PPSNo already exists** |  | **Step 5:** PPSNo is in staff File already  **Step 6:** The system displays “A staff with this PPS number already exists”  **Step 7:** Return to Step 4 |
| **Conclusions** | The new staff members data is stored in the database. | |
| **Post conditions** | This staff member can now register customers | |
| **Business Rules** | PPSNo must not be duplicated in the database. | |
| **Implementation Constraints** |  | |

### Update Staff

This component lets the Manager update a staff

Manager

<<Include>>

<<Extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Update Staff** | |
| **Use Case Id** | BS005 | |
| **Priority** | 9 | |
| **Source** | Bank Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function updates a staff members data in the system. | |
| **Preconditions** | The staff member must already exist in the system. | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** Manager invokes the update Staff function.  **Step 3:** The Manager selects the Staff they wish to update.  **Step 6:** The Manager enters the details of the Staff they wish to edit. | **Step 2:** The system retrieves the ID, Full name and PPSNo for all Staff from the Staff File.  **Step 2:** The system displays the Select Staff UI.  **Step 4:** The system retrieves the full details of that Account from the Account File.  **Step 5:** The system displays the full details using the Edit Staff UI.  **Step 7:** The system validates the Modified data:  **Step 8:** The system displays a confirmation message  **Step 9:** The system resets the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid data entered** |  | **Step 4:** The system fails to update the staff details in the database  **Step 5:** The system displays an appropriate error message  **Step 6:** Return to Step 6 |
| **Conclusions** | The modified staff data will be updated in the system. | |
| **Post conditions** | The up to date staff details can now be viewed in the database. | |
| **Business Rules** | The staff member must exist and be “Employed” in the database. | |
| **Implementation Constraints** |  | |

### Remove staff

This component lets the Manager remove a staff member.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Remove Staff** | |
| **Use Case Id** | BS006 | |
| **Priority** | 5 | |
| **Source** | Bank Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function changes the staff status to “Unemployed” in the System. | |
| **Preconditions** |  | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** Manager invokes the close account function.  **Step 4:** The manager selects the StaffID of the Staff they wish to remove in the Staff File. | **Step 2:** The system retrieves the ID, Full name and PPSNo for all “Employed” Staff from the Staff File.  **Step 3:** The system displays the Remove Staff UI  **Step 5:** Update status from Employed (“E”) to Unemployed (“U”)  **Step 6:** DateLeft is set to the current date.  **Step 6:** The system displays a confirmation message  **Step 7:** The system resets the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **System error** |  | **Step 5:** The system fails to change the staff status in the database  **Step 5:** The system displays an appropriate error message |
| **Conclusions** | The staff has been removed from the database. | |
| **Post conditions** | The staff can no longer create/modify/remove customers from the database. | |
| **Business Rules** | The staff must already exist in the database. | |
| **Implementation Constraints** |  | |

## Manage Transactions

### Deposit Transaction

This component lets the Customer make a Deposit Transaction.

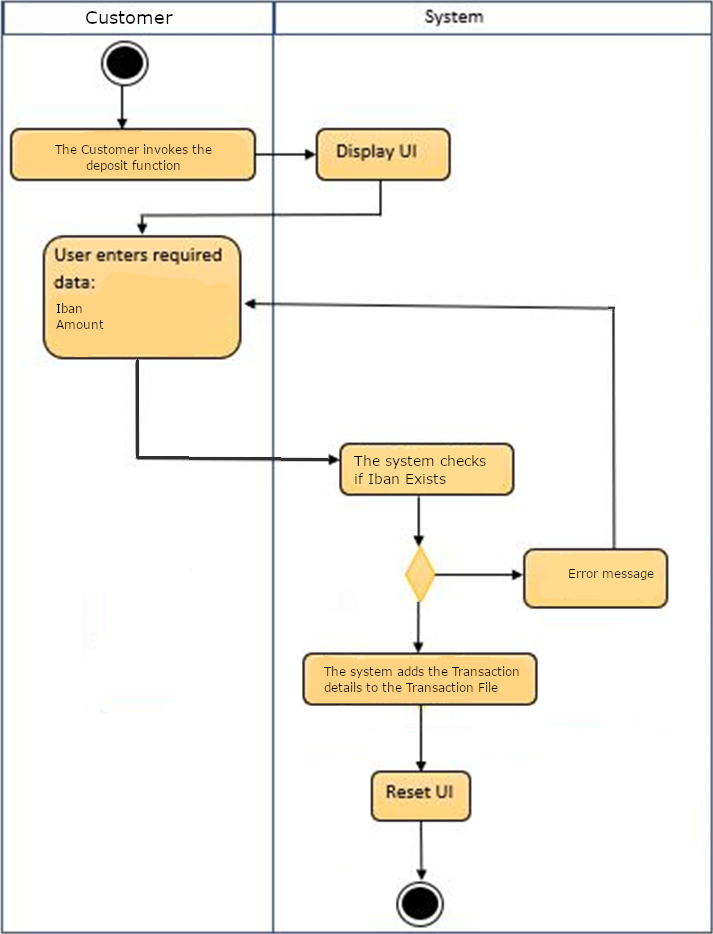
Customer

<<Include>>

<<Extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Deposit Transaction** | |
| **Use Case Id** | BS007 | |
| **Priority** | 3 | |
| **Source** | Bank Customer | |
| **Primary Business Actor** | Customer | |
| **Other Participating Actors** |  | |
| **Description** | This function Deposits money into a Customer’s account | |
| **Preconditions** | The Customer must have an account with the Bank | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** Cusomer invokes the Deposit function.  **Step 3:** The Customer enters the following information:   * Iban (22) * Amount (Decimal 5) | **Step 2:** The system displays the UI  **Step 4:** The system validates the data entered:   * All fields must be entered * Iban must be alphanumerical * Iban must be registered in the system * Deposit must be numerical   **Step 5:** The System produces a TransactionID, Date and time  **Step 6:** The System Sets TransactionType to “W”  **Step 7:** The system adds the Transaction details to the Transaction File:   * TransactionID * Iban * TransactionType * Amount * Date * Time   **Step 8:** The system displays a confirmation message  **Step 9:** The system resets the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid iban** |  | **Step 4:** The system fails to add Deposit  **Step 5:** The system displays “Invalid iban”  **Step 6:** Return to Step 3 |
| **Conclusions** | The Transaction has been added to the database. | |
| **Post conditions** | The Deposit can now be viewed, and money may be withdrawn. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

**Deposit Transaction Activity Diagram**



### Withdraw Transaction

This component lets the Customer make a withdraw Transaction.

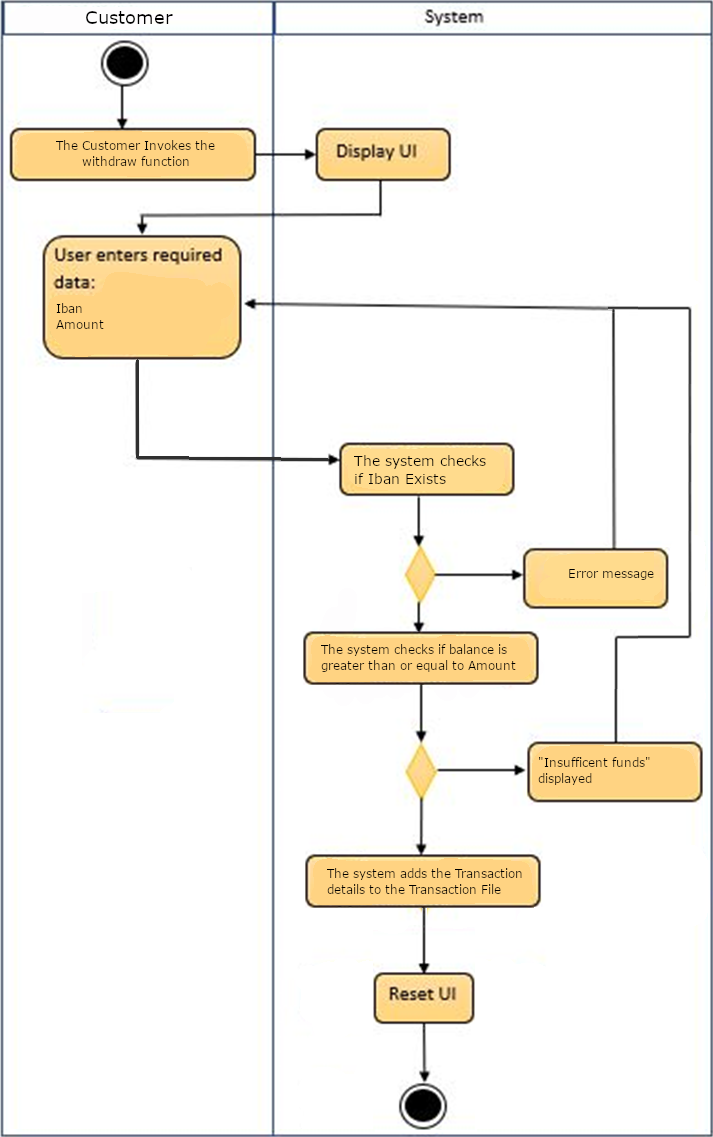
Customer

<<Include>>

<<Extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Withdraw Transaction** | |
| **Use Case Id** | BS009 | |
| **Priority** | 4 | |
| **Source** | Bank Customer | |
| **Primary Business Actor** | Customer | |
| **Other Participating Actors** |  | |
| **Description** | This function withdraws money from an account | |
| **Preconditions** | The Balance must be greater then withdraw amount | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** Customer invokes the Withdraw function.  **Step 3:** The Customer enters the following information:   * Iban (22) * Amount (Decimal 5) | **Step 2:** The system displays the UI  **Step 4:** The system validates the data entered:   * All fields must be entered * Iban must be alphanumerical * Iban must be registered in the system * withdraw must be numerical   **Step 5:** The system calculates balance for this customer  **Step 6:** Balance must be greater or equal to withdraw    **Step 7:** The System produces a TransactionID, Date and time  **Step 8:** The System Sets transaction type to “D”  **Step 7:** The system adds the Transaction details to the Transaction File:   * TransactionID * Iban * Amount * TransactionType * Date * Time   **Step 8:** The system displays a confirmation message  **Step 9:** The system resets the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Insufficient Funds** |  | **Step 6:** Balance is less then withdraw  **Step 5:** The system displays “Insufficient Funds”  **Step 6:** Reset UI |
| **Invalid iban** |  | **Step 4:** The system fails to add withdraw  **Step 5:** The system displays “Invalid iban”  **Step 6:** Return to Step 3 |
| **Conclusions** | The withdraw Transaction has been added to the database. | |
| **Post conditions** | Further withdraw/Transfer Transactions may not be possible | |
| **Business Rules** | The Customer must have the funds to withdraw. | |
| **Implementation Constraints** |  | |

**Withdraw Transaction Activity Diagram**



### Transfer Transaction

This component lets the Customer make a Transfer Transaction.

Customer

<<Include>>

<<Extends>>

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Transfer Transaction** | |
| **Use Case Id** | BS010 | |
| **Priority** | 5 | |
| **Source** | Bank Customer | |
| **Primary Business Actor** | Customer | |
| **Other Participating Actors** |  | |
| **Description** | This function make a Transfer request between two accounts | |
| **Preconditions** | The Customer must have an Account | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** Cusomer invokes the Withdraw function.  **Step 3:** The Customer enters the following information:   * Iban (22) * Amount (Decimal 5) * Beneficiary Iban (22) | **Step 2:** The system displays the UI  **Step 4:** The system validates the data entered:   * All fields must be entered * Iban must be alphanumerical * Iban must be registered in the system * withdraw must be numerical * Beneficiary Iban must be alphanumerical   **Step 5:** The system calculates balance for this customer  **Step 6:** Balance must be greater or equal to withdraw    **Step 7:** The System produces a TransactionID, Date and time  **Step 8:** The System sets TransactionType to “W”  **Step 9:** The system adds the Withdraw details to the Transaction File:   * TransactionID * Iban * Amount * TransactionType * Date * Time   **Step 10:** The System produces a second TransactionID, Date and time  **Step 11:** The System sets the second TransactionType to “D”  **Step 12:** The system adds the Deposit details to the Transaction File:   * TransactionID * Beneficiary Iban * Amount * TransactionType * Date * Time   **Step 13:** The system displays a confirmation message  **Step 14:** The system resets the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Insufficient Funds** |  | **Step 6:** Balance is less then withdraw  **Step 5:** The system displays “Insufficient Funds”  **Step 6:** Reset UI |
| **Invalid iban** |  | **Step 4:** The system fails to add withdraw  **Step 5:** The system displays “Invalid iban”  **Step 6:** Return to Step 3 |
| **Conclusions** | The withdraw Transaction has been added to the database. | |
| **Post conditions** | Further withdraw/Transfer Transactions may not be possible | |
| **Business Rules** | The Customer must have the funds to withdraw. | |
| **Implementation Constraints** |  | |

## Generate Reports

### View Monthly Transaction Report

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **View Monthly Transactions** | |
| **Use Case Id** | BS010 | |
| **Priority** | 7 | |
| **Source** | Bank Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function allows the Monthly Transactions to be viewed | |
| **Preconditions** | Selections can only be made on months that have past | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** Manager invokes the View Monthly Transaction function.  **Step 3:** The manager selects which month and year They wish to view. | **Step 2:** The system displays the UI  **Step 4:** The system displays sum of all transactions based on the information in the Transactions File  **Step 5:** The system displays a Graph for the selected period.  **Step 6:** The system displays a confirmation message  **Step 7:** The system resets the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid data entered** |  | **Step 4:** The system fails to gather necessary information to form a revenue report  **Step 5:** The system displays an appropriate error message |
| **Conclusions** | The Monthly Transactions gets displayed. | |
| **Post conditions** | The Monthly Transactions are exported for later viewing. | |
| **Business Rules** | The function can only be run on previous months. | |
| **Implementation Constraints** |  | |

### View Customer Signup

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **View Customer Signup** | |
| **Use Case Id** | BS011 | |
| **Priority** | 8 | |
| **Source** | Bank Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function allows the Customer Signups to be viewed | |
| **Preconditions** | Selections can only be made on months that have past | |
| **Trigger** |  | |
| **Expected Scenario** | **Actor Action** | **System Response** |
|  | **Step 1:** Manager invokes the View Customer Signup function.  **Step 3:** The manager selects which month and year They wish to view. | **Step 2:** The system displays the UI  **Step 4:** The system displays number of new signups based on the information in the Accounts File  **Step 5:** The system displays a Graph for the selected period.  **Step 6:** The system displays a confirmation message  **Step 7:** The system resets the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Invalid Date entered** |  | **Step 4:** A future month or a month in progress selected.  **Step 5:** The system displays an appropriate error message |
| **Conclusions** | The Customer signup analysis gets displayed. | |
| **Post conditions** | The Customer signup analysis is exported for later use. | |
| **Business Rules** | The function can only be run on previous months. | |
| **Implementation Constraints** |  | |

# System Model

The following dataflow diagrams have been produced for the system:

## Level-0 DFD

BankSYS

Transaction Details

Transaction Confirmation

Customer

## Level-1 DFD

Transaction Details

Customer

Staff Details

Staff Details

Transaction Details

Transaction Details

Account

Details

Account Details

Account Details

Transaction Confirmation

P4

Generate

Reports

D3

Staff File

D3

Transaction File

P3

Manage

Transactions

P2

Manage

Steff

D3

Account File

P1

Manage

Customer

## Level-2 DFD (Process P1: Manage Accounts)

Account

Details

P1.2

Close

Account

P1.1

Add

Account

Account

Details

Account

Details

D1

Accounts File

## Level-2 DFD (Process P2: Manage Staff)

Staff

Details

P2.3

Remove

Staff

P2.1

Add

Staff

Staff

Details

Staff

Details

Staff

Details

Staff

Details

D2

Staff File

P2.2

Update

Staff

**Level-2 DFD** (Process P3 Manage Transactions)

Transaction

Confirmation

Transaction

Confirmation

Transaction

Confirmation

Transaction

Details

Transaction

Details

Transaction

Details

Transaction

Details

Transaction

Details

Transaction

Details

Transaction

Details

Customer

Details

Customer

Details

Customer

Details

Customer

Details

D3

Account File

D3

Transaction File

P2.2

Deposit

Transaction

P2.2

Withdraw

Transaction

P2.2

Transfer Transaction

Customer

## Level-2 DFD (Process P4: Generate Reports)

Customer

Details

D3

Accounts File

P4.2

Customer

Singup

P4.1

Transaction

Report

Transaction

Details

D3

Transaction File

# Data Model (Class Diagram)

These are my class diagrams for my data model.

## Class Diagram

Object Model – UML Class Diagram

Account

* AccountID
* Status
* AccountName
* TypeId
* Balance
* DateCreated
* DateClosed

Customer

* CustomerID
* Status
* FirstName
* LastName
* PPSNo
* DateOfBirth
* CountryCode
* PhoneNumber
* AddressLine1
* AddressLine2
* AddressLine3
* Town
* County
* Eircode
* DateCreated

Can have

0…\*

1

1

Can have

0…\*

Transaction

* TransactionID
* AccountID
* TransactionType
* Amount
* Note
* Debtor
* Timestamp

## Relational Schema

Customer (CustomerID, Status, FirstName, LastName, PPSNo, DateOfBirth, CountryCode,PhoneNumber,AddressLine1, AddressLine2, AddressLine3, Town, County, Eircode, AccountType, Iban, DateCreated)

Transaction (TransactionId, AccountID, Amount, Type, Date, Time)

Staff (StaffID, FirstName, LastName, MobileNo, Email, Iban, PPSNo, PayRate, AddressLine1, AddressLine2, AddressLine3, Town, County, Eircode, DateStarted, Status, DateLeft)

## Database Schema

A definition of the database to be implemented.

This includes primary key, foreign key and other constraints to be implemented.

**Schema: BankSYS Schema**

**Relation:** County

Attributes:

CountyId NUMBER(2) NOT NULL UNIQUE,

County VARCHAR2(30) NOT NULL UNIQUE,

**Primary Key:** CountyId

**Relation:** CountryCode

Attributes:

CountryId NUMBER(3) NOT NULL UNIQUE,

Country VARCHAR2(30) NOT NULL UNIQUE,

CountryCode VARCHAR2(6) NOT NULL UNIQUE,

**Primary Key:** CustomerId

**Relation:** AccountType

Attributes:

TypeId NUMBER(1) NOT NULL UNIQUE,

TypeName VARCHAR2(30) NOT NULL UNIQUE,

TypeCode CHAR(1) NOT NULL UNIQUE,

**Primary Key:** TypeId

**Relation:** TransactionType

Attributes:

TypeId NUMBER(1) NOT NULL UNIQUE,

TypeName VARCHAR2(30) NOT NULL UNIQUE,

Identifier CHAR(1) NOT NULL UNIQUE,

**Primary Key:** TypeId

**Relation:** Customer

Attributes:

CustomerId CHAR(8) NOT NULL UNIQUE,

Status CHAR(1) DEFAULT 'A' NOT NULL,

FirstName VARCHAR2(40 CHAR) NOT NULL,

LastName VARCHAR2(40 CHAR) NOT NULL,

PPSNumber VARCHAR2(9 CHAR) NOT NULL,

CountryCode NUMBER(3) NOT NULL,

PhoneNumber NUMBER(12) NOT NULL,

DateOfBirth DATE NOT NULL,

AddressLine1 VARCHAR2(50 CHAR) NOT NULL,

AddressLine2 VARCHAR2(50 CHAR),

AddressLine3 VARCHAR2(50 CHAR),

Town VARCHAR2(22 CHAR) NOT NULL,

CountyId NUMBER(2) NOT NULL,

Eircode CHAR(8 CHAR) NOT NULL,

DateCreated DATE NOT NULL,

**Primary Key:** CustomerId

**Foreign Key:**

**Relation:** Login

Attributes:

LoginId CHAR(8) NOT NULL,

CustomerId CHAR(8) NOT NULL,

PAC CHAR(5) NOT NULL,

**Primary Key:** LoginId

**Relation:** Account

Attributes:

AccountId Number(9) NOT NULL,

CustomerId CHAR(8) NOT NULL,

Status CHAR(1) DEFAULT 'A' NOT NULL,

AccountName VARCHAR2(40 CHAR) NOT NULL,

AccountType VARCHAR2(40 CHAR) NOT NULL,

Balance NUMBER(7) DEFAULT 0 NOT NULL,

DateCreated DATE NOT NULL,

DateClosed DATE NOT NULL,

**Primary Key:** AccountId

**Relation:** Transaction

Attributes:

TransactionId NUMBER(14) NOT NULL,

AccountId NUMBER(8) NOT NULL,

Type CHAR(1) NOT NULL,

Amount VARCHAR2(40 CHAR) NOT NULL,

Note VARCHAR2(30 CHAR),

Debtor NUMBER(8),

Timestamp TIMESTAMP(0) NOT NULL,

**Primary Key:** TransactionId

# Conclusion

This document provides guidelines to build a banking system that provides functions for staff management, transaction management, account management and Reporting. The system will have four processes: Manage Transactions, Manage Account, Manage staff and Reports. Each will have its own functions.

Manage staff will have three functions: Add Staff, Update Staff and Remove Staff. Add Staff will allow the Admin to add a new staff member to the system. Update Staff will allow the Admin to update an existing staff member. Remove Staff will set the permissions of an existing staff member to ‘0’ so that they can no longer provide services.

Manage Transactions will have three functions: Deposit Transaction, Transfer Transaction and Withdraw Transaction. Deposit Transaction will allow the user to deposit money into an account on the system. Transfer Transaction will allow the user to Transfer money between accounts on the system. Withdraw Transaction will allow the user to withdraw money into an account on the system.

Manage Account will have three functions: Add Account, Modify Account information and Close Account. Add Account will allow the user to add an account into to the system. Modify Account information will allow a user to modify their account information into the system and Close Account will lock an account and prevent modification and money transfer.

Administrative Reporting has two functions: View Total Transactions and View Customer signup. View Total Transactions shows the Total Transactions for a specific month in a specific year. View Customer signup shows the number of signups for a specific month in a specific year.

This document will go through how each function will work using a Level 0 DFD, Level 1 DFD and three Level 2 DFD’s, one for each process.

There will be a class diagram for the data models.

There is a relational schema and database schema for the creation of a database for the system.