CSCE5430: Software Engineering

Project Workbook

Spring 2023

Revision History

Date	Issue	Description	Author
January 27, 2023	Project1-Phase1	Adding Glossary (Question 4)	Abdelnasser Ouda
January 27, 2023	Project1-Phase1	Adding system actors (Question 6.1)	Abdelnasser Ouda
January 27, 2023	Project1-Phase1	Adding system use cases (Question 7.1)	Abdelnasser Ouda
January 27, 2023	Project1-Phase1	Adding use case diagram (Question 8.1)	Abdelnasser Ouda
February 19, 2023	Project1-Phase2	Produce a list of candidate classes	Abdelnasser Ouda
February 19, 2023	Project1-Phase2	Filtering the list of candidate classes	Abdelnasser Ouda
February 19, 2023	Project1-Phase2	Produce a potential class diagrams	Abdelnasser Ouda
February 19, 2023	Project1-Phase2	Identified potential class attributes	Abdelnasser Ouda
February 19, 2023	Project1-Phase2	Revised the diagrams	Abdelnasser Ouda
February 19, 2023	Project1-Phase2	Update the glossary	Abdelnasser Ouda
March 5, 2023	Project1-Phase2	Create the database schema (unrequired part)	Abdelnasser Ouda
March 22, 2023	Project1-Phase3	Potential iFinance Sequence Diagram	Group11/003

Table of Contents

iFINANCE System Glossary

1.1 Introduction

This document is used to define terminology specific to the problem domain, explaining terms, which may be unfamiliar to the reader of the use-case descriptions or other project documents. Often, this document can be used as an informal *data dictionary*, capturing data definitions so that use-case descriptions and other project documents can focus on what the system must do with the information.

1.2 Glossary

The glossary contains the working definitions for the key concepts in the iFINANCE System.

Term	definition
Authentication	Authentication is the process of determining whether someone or something is, in fact, who or what it is declared to be.
User account	A user is a person who uses iFINANCE system. Each user should have an account in order to be identified by iFINANCE. To login to an account, a user is typically required to authenticate himself/herself with a password or other credentials for the purposes of accounting, security, logging, and resource management.
Double-entry bookkeeping	It is a system of accounting in which every transaction has a corresponding positive and negative entry (debits and credits).
Assets	Tangible and intangible items that the company owns that have value (e.g. cash, computer systems, patents).
Liabilities	The money that the company owes to others (e.g. mortgages, vehicle loans).
Income	The money the company earns from its sales of products or services, and interest and dividends earned from marketable securities.
Expenses	The money the company spends to produce the goods or services that it sells (e.g. office supplies, utilities, advertising).
Chart of Accounts	A chart of accounts is a listing of the names of the accounts that a company has identified and made available for recording transactions in its general ledger. A company has the flexibility to tailor its chart of accounts to best suit its needs, including adding accounts as needed.
Master Accounts	The Master Accounts are a super-set of all allowable accounts in the chart of accounts. Transactions are never posted to the Master Accounts.
UI component	UI stands for User Interface. It is a junction between a user and a computer program. An interface is a set of commands or menus through which a user communicates with a program.
PDF Document	Portable Document Format (PDF) is a file format used to present and exchange documents reliably, independent of software, hardware, or operating system.
Windows-based computers	A personal computer powered by Microsoft Windows operating system in a form intended for regular use at a single location desk/table due to its size and power requirements.
Database Schema	The term schema refers to the organization of data as a blueprint of how the database is constructed (divided into database tables in the case of relational databases).
Sequence Diagram	This shows the interaction between user and a use case of the system and describes how the process works, in what order objects are working together to satisfy the use case.

DataGridView	A Windows forms control that provides a customizable table for displaying data.
Context Menu Strip	A component in windows forms that provides a shortcut menu for a control. It appears when user right-clicks the control.
Windows Forms	A graphical user interface library in .net framework.
Event Handler	A delegate that represents a method that will handle an event generated by UI component.
Data Source	An object that provides data to a control such as DataGridView, ComboBox.
CRUD	An acronym for Create, Read, Update, and Delete. It refers to four basic functions that are required for managing data in database.
MVC	An acronym for Model-View-Controller, which is a software architectural pattern for implementing user interfaces. It separates the application into three interconnected components data, view, and logic.

iFINANCE System Actors

The first useful step to analyze the system functionality is to look in the problem statement at the things that interact with the system. In UML use case analysis, these external things are called **actors**. Actors are identified based on the following:

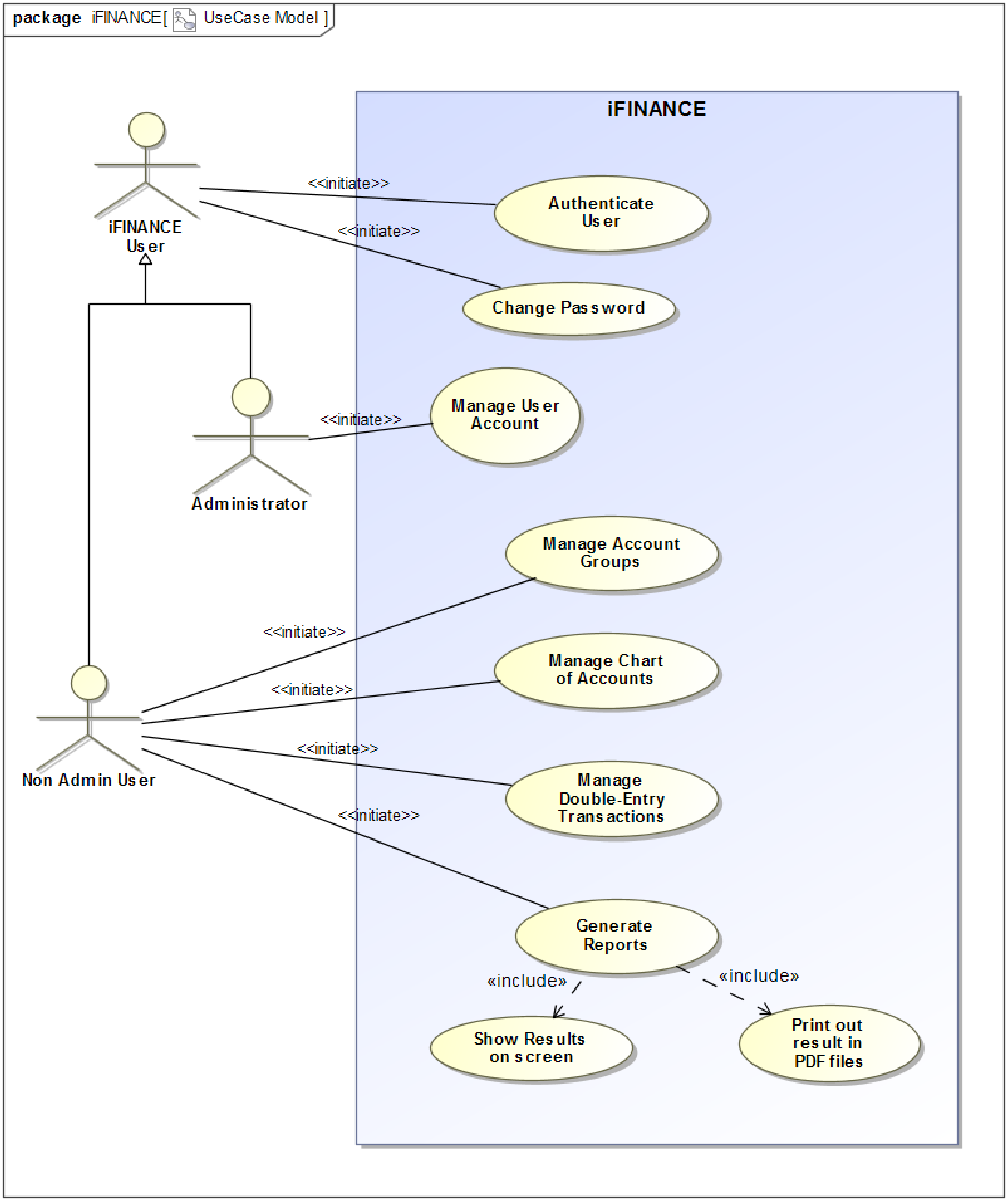
- Actors are always external to the system – they are therefore outside our control.
- Actors interact directly with the system.
- Actors represent roles that people and things play in relation to the system, not specific people or specific things.
- Each actor has a unique name and description.

Actor	Description
iFINANCE User	A general user interacting with iFINANCE system. This general user can be the day-to-day regular user (we called non-admin user) or the system administrator. In order to this general user to use iFINANCE, he/she needs to successfully login to the system.
Administrator	A special type of iFINANCE User who has the privileges to add user accounts into the system, edit and delete their profiles. The administrator user account will be shipped with the system.
Non admin user	A day-to-day iFINANCE user who wish to use iFINANCE system to control the his/her personal finances, keeping track of bank accounts, cash, credit cards, and investment accounts.

Use Case	Description
Authenticate User	iFINANCE shall provide a functionality to authenticate its user by using a username and encrypted password. Once the user has logged in, the set of controls dynamically changes to match the permissions of this account. Note that, each user should be successfully authenticated before using any of iFINANCE functionalities.
Manage User Account	iFINANCE shall provide the functionality for the system administrator to create user accounts for the regular iFINANCE users according to a set of access controls predefined for each user type. The administrator account itself is shipped with the system.
Change Password	iFINANCE shall provide the functionality for its users to change their secret passwords.
Manage Account Group	iFINANCE shall provide the ability for its non-admin users to add, update, or delete a custom sub-categories (called Groups). This includes groupID, groupName, groupElement, i.e., Assets, Liabilities, Income, and Expenses, and groupParent. Each group can be decomposed to further level of sub-groups (the groupParent field is responsible to maintain the parent-children link).
Manage Chart of Accounts	Manage Chart of Accounts UC shall provide a graphical user interface GUI to help non-admin users to add, update, or delete Master Account (e.g., cash, bank, and credit card). Each Master Account in the Chart of Accounts record will include accountID, accountName, openingAmount, closingAmount, and accountGroup (to specify a group/sub-group to which the account belongs).
Manage Double-Entry Transactions	iFINANCE shall provide a tool for its non-admin users to manage accounts' transactions (called double entry bookkeeping or double entry accounting). Each transaction will always have two effects: a debit entry and a credit entry. This tool will facilitate the non-admin user to add, update, or delete a transaction's entry in a Master-Detail style form. The Master part of the form includes the transaction number, date, the total amount of debit, the total amount of credit in addition to a comments/notes field. The Detail part of this form is a grid that has in each line the account number and name, the debit amount, the credit amount and a field of comments.
Generate Reports	iFINANCE shall allow the non-admin users to generate valuable financial reports including the Trial Balance, Balance Sheet, a Profit and Loss statement, or a cash flow statement from the user financial data. This UC includes two UCs Show Results on Screen and Print out results in PDF files.
Show Results on Screen	This is a supplier UC for the Generate Reports UC that shall allow the non-admin users to see the generated reports in an easy to see in the computer screen.
Print out results in PDF files	This is a supplier UC for the Generate Reports UC that shall allow the non-admin users to export the generated reports in a formatted style to PDF file type. These PDF files can be printed out later on.

iFINANCE System Use cases

iFINANCE System Use Case Diagram



List of Candidate Classes

bank accounts

cash

credit cards

investment accounts

users

deposits

expenses

reports

financial health

double-entry

people

accounting knowledge

financial accounting categories

assets

liabilities

income

expenses

debits type

credits type

financial transactions

user-defined sub-categories

balance sheet

profit and Loss statement

cash flow statement

financial data

master accounts

money

desktop computers

tablet PCs

clubs

self-employed

small businesses

home

user name

encrypted password

access controls

permissions

system administrator

privileges

user accounts

profiles

administrator account

custom group

main form

groupID

groupName

groupElement

groupParent

level of sub-groups

parent-children link

Chart of Accounts

UI component

accountID

accountName

openingAmount

closingAmount

accountGroup

report

master-detail style form

grid

line

account number

account name

the debit amount

the credit amount

field of comments

trial balance

cash flow statement

tabular layout

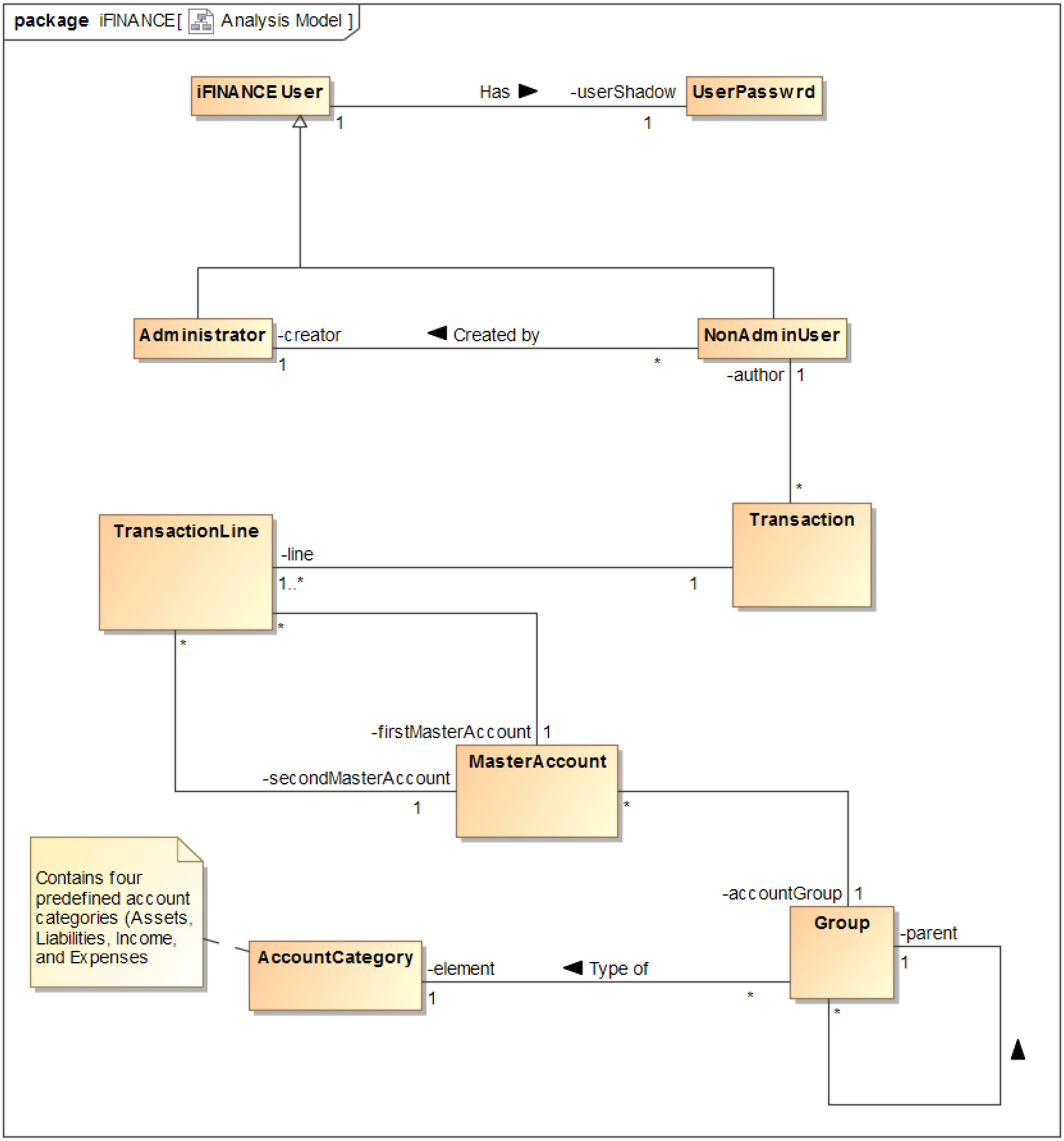
Potential iFINANCE Classes

Class name	Type	Brief Description
UserAuthenticationForm	Boundary	The UserAuthenticationForm will be displayed by the iFINANCE system asking the user to enter its username and password to be authenticated.
iFINANCE User	Entity	The common information of the iFINANCE non admin users and the iFINANCE admin such as User ID and User name will be maintained by this entity class.
Non admin user	Entity	Non admin users is a specification class inherits the iFINANCE User class and add specific information about the user such as name, date of birth, address and email.
Administrator	Entity	iFINANCEAdmin is a specification class inherits the iFINANCE User class and add specific information about the system administrator such as the starting and ending date for the admin responsibility.
UserPassword	Entity	The UserPassword class store the iFINANCE user account information that include a password related information like an encrypted password of a user, the day the password expires, whether or not the password has to be changed from time to time, the minimum and maximum time between password changes, etc.
UserAuthenticationController	Control	A control class to accept and validate the iFINANCE username and its corresponding password. It takes the decision whether the iFINANCE user is allowed to access the iFINANCE services or not.
ManageUserAccountsForm	Boundary	iFINANCE displays the ManageUserAccountsForm when the system admin initiates the process for adding/modifying iFINANCE user account. This form will help the admin to enter the user general information like name and addresses and the secret information like password and the associated access roles.
UserRole	Entity	UserRole class maintains a list of all available iFINANCE access roles and the related permissions. For example, System administrator who has a privilege to create/modify user accounts, Non admin user who has a privilege for all iFINANCE financial functionalities.
ManageUserAccountsController	Control	A control class to create and store the new iFINANCE users' information. It also determines whether the entered username has been used by other user or not, it encrypts the password before storing, and finally assign a specific system access role to the user.
ChangePasswordForm	Boundary	The UserAuthenticationForm will be displayed by the iFINANCE system when the non admin user ask to change his/her password. This form will ask the user to enter the old password and the new password two times.

ChangePasswordController	Control	A control class to accept and validate the non admin user old and new password. The old password needs to be related to this user and the two new passwords should match each other. This control will update the UserPassword entity class.
CustomGroupForm	Boundary	CustomGroupForm is a boundary class that show all exiting financial group names in a Tree View style control, allowing the user to edit, add, or delete a groups and subgroups.
Group	Entity	Group class defines the basic information of the system financial group names and its parent-child relationship. It also determines the type of the group whether it is an Assets, Liabilities, Income, or Expenses.
AccountCategory	Entity	An entity class that defines the account types such as Income, Expense, Asset.
CustomGroupController	Control	This class defines the attributes and the methods needed to retrieve the information of all account groups and subgroups names from the Group objects and display it in a TreeView control in CustomGroupForm organized by the main account type categories. It also allows for edit, add, or delete functionalities using the same form.
ChartOfAccountForm	Boundary	A boundary class to define a graphical user interface GUI element that display a list of all Master accounts at iFINANCE information such as name, group, type, closing and opening amount. The form allows the user to add new master account, edit or delete exiting ones.
MasterAccount	Entity	This entity class defines the basic information of the system financial Masters Accounts such as name, group, type, closing and opening amount.
ChartOfAccountController	Control	This class defines the attributes and the methods needed to retrieve the existing master accounts data and display it in a grid control in ChartOfAccountForm. It allows the user to add new master account, edit or delete exiting ones using the same form.
iFINANCETransactionForm	Boundary	iFINANCE provides a that facilitate the user to add, update, or delete a transaction's entry in a Master-Detail style form. The iFINANCETransactionForm consists of a master part that includes the transaction number, date, the total amount of debit, the total amount of credit in addition to a comments/notes field. The Detail part of this form is a grid that has in each line the account number and name, the debit amount, the credit amount and a field of comments.
Transaction	Entity	This entity class defines the data involved in master part of the iFINANCETransactionForm. This data includes: the transaction number, date, the total amount of debit, the total amount of credit in addition to a comments/notes field.
TransactionLine	Entity	This entity class defines the data involved in detail part of the iFINANCETransactionForm. This data includes: the account number and name, the debit amount, the credit amount and a field

		of comments in a grid format.
iFINANCETransactionController	Control	A control class to define the application logic that add, edit, or delete accounts’ transactions (called double entry bookkeeping or double entry accounting). In each transaction it will recognize one of two processes: a debit entry and a credit entry, such that for every Debit entry, there will always be an equal Credit entry. In the Debit entries we do one of the following: increase assets account, increase expense account, decrease liability account, or decrease income account. However, In the Credit entries we do one of the following: decrease assets account, decrease expense account, increase liability account, or increase income account. iFINANCE deals with two basic transactions—deposits and withdrawals.
GenerateReportForm	Boundary	iFINANCE provides their non admin users with a form represented by the GenerateReportForm boundary class by which the user can enter specific criteria and option to generate reports such as Trial Balance, Balance Sheet, a Profit and Loss statement, or a cash flow statement.
TrialBalanceReport	Boundary	TrialBalanceReport class presents the closing balance of all master accounts in a tabular layout with the debit accounts on one side and the credit accounts on the other. The sum of all credit balances should always match the sum of all debit balances. The Trial Balance is the basis of preparing the Profit and Loss account and the Balance Sheet.
BalanceSheetReport	Boundary	BalanceSheetReport class shows Assets plus Inventory on one side and Liabilities + Profit or Loss (as derived above) on the other. These two sides should be equal.
ProfitLossStatement	Boundary	ProfitLossStatement class is all about subtracting all Expenses from Income to derive a Profit or Loss figure. Thus: Profit or (Loss) = Income – Expenses.
CashFlowStatement	Boundary	CashFlowStatement is a boundary class that shows how changes in balance sheet accounts and income affect cash and cash equivalents, and breaks the analysis down to operating, investing and financing activities.
GenerateReportController	Control	This control class defines the methods that read the user options in which report they would like to generate and to read the specific criteria that is related to the chosen report, such as date range, account types, or account categories. Then calculate and generate the desire report. It also determines whether the output will be displayed on the screen or saved as PDF.

Potential iFINANCE class diagrams (entity classes only)



Revised iFINANCE UML class diagram

1. iFINANCEUser Class

Attribute name	Type	Brief Description
ID	String	This is the primary key of the iFINANCE user info class. Every registered iFINANCE user is assigned a unique user ID.
name	String	The name attribute stores the full name of the iFINANCE user.

2. NonAdminUser Class

Attribute name	Type	Brief Description
address	String	The address attribute stores the address of the iFINANCE non admin user.
email	String	The email attribute stores the email address of birth of the iFINANCE non admin user.

3. Administrator Class

Attribute name	Type	Brief Description
dateHired	Date	The date when the system administrator is hired by iFINANCE.
dateFinished	Date	The date when the system administrator left the iFINANCE.

4. UserPassword Class

Attribute name	Type	Brief Description
ID	String	This is the primary key of the UserPassword class. The value of this attribute need to be the same as the value of the iFINANCE user ID.
userName	String	The name attribute stores the use account name of the non admin user.
encryptedPassword	String	The userEncryptedPassword attribute stores the encrypted version of the non admin user password. A salted hash will be used in order to encrypt the password.
passwordExpiryTime	Integer	From time to time the system requires the user to change the password. The passwordExpiryTime attribute stores this period of time.
userAccountExpiryDate	Date	The userAccountExpiryDate attribute stores the expiry date of the user account if any.

5. AccountCategory Class

Attribute name	Type	Brief Description
ID	String	This is the primary key of the AccountCategory class. The value of this attribute distinguishes one category from the other.

name	String	The name attribute stores the iFINANCE pre-defined account category name. Example, Assets, Liabilities, Income, and Expenses.
type	String	The type attribute stores the type of account category, this could be Debit or Credit.

6. Group Class

Attribute name	Type	Brief Description
ID	String	This is the primary key of the Group class. The value of this attribute distinguishes one account group from the other.
name	String	The name attribute stores the name of the group or its subgroups.

7. MasterAccount Class

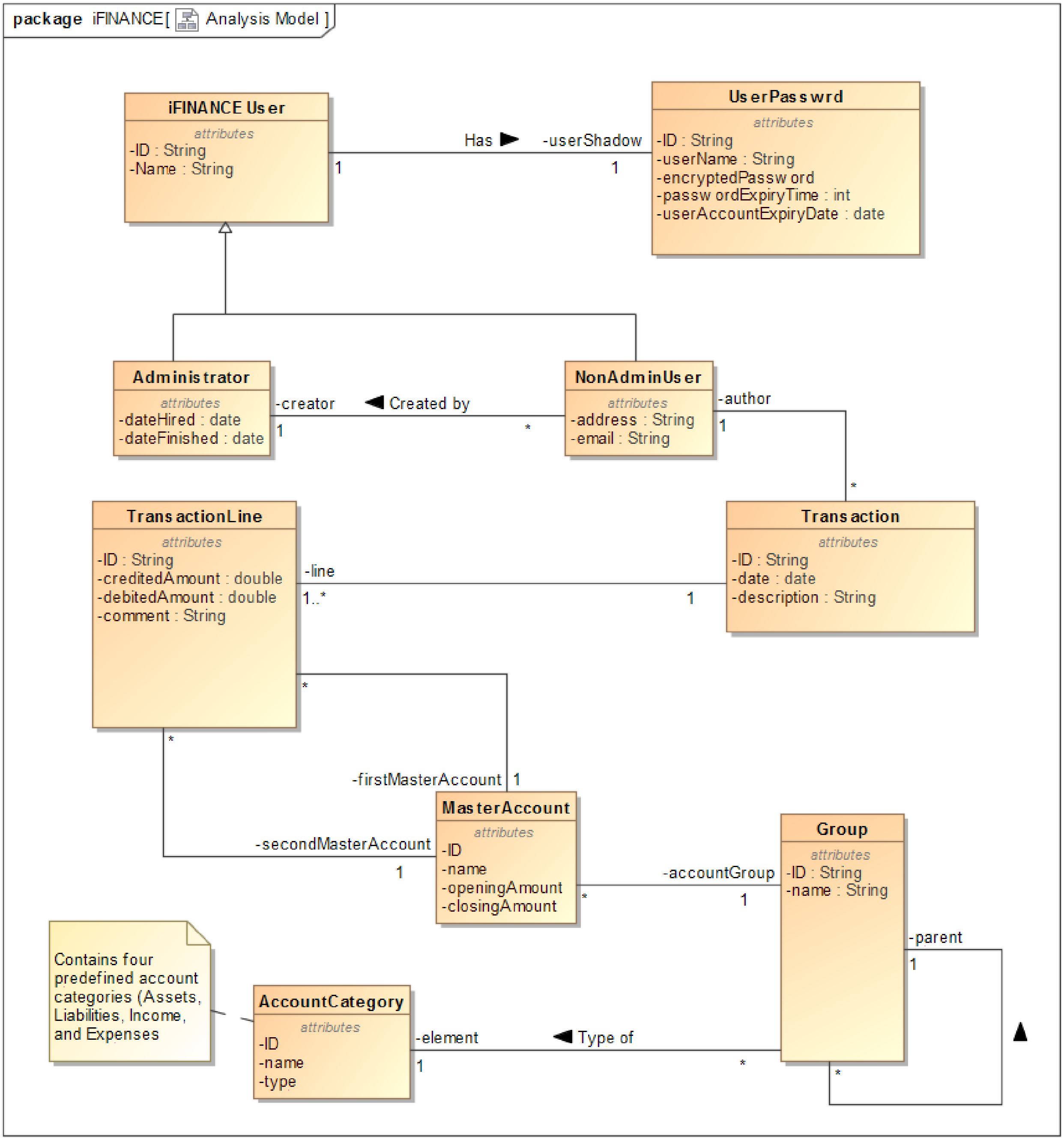
Attribute name	Type	Brief Description
ID	String	This is the primary key of the master account class.
name	String	The name attribute stores the full name of the master account.
openingAmount	double	The openingAmount attribute stores the balance brought forward at the beginning of an accounting period.
closingAmount	double	The openingAmount attribute stores the amount remaining in an account within the chart of accounts, positive or negative, at the end of an accounting period or year end.

8. Transaction Class

Attribute name	Type	Brief Description
ID	String	This is the primary key of the transaction header class. The value of this attribute should at least one value in the transaction line class objects.
date	Date	The date attribute stores the date and time of the transaction.
description	String	The description attribute stores the detail comments about the transaction.

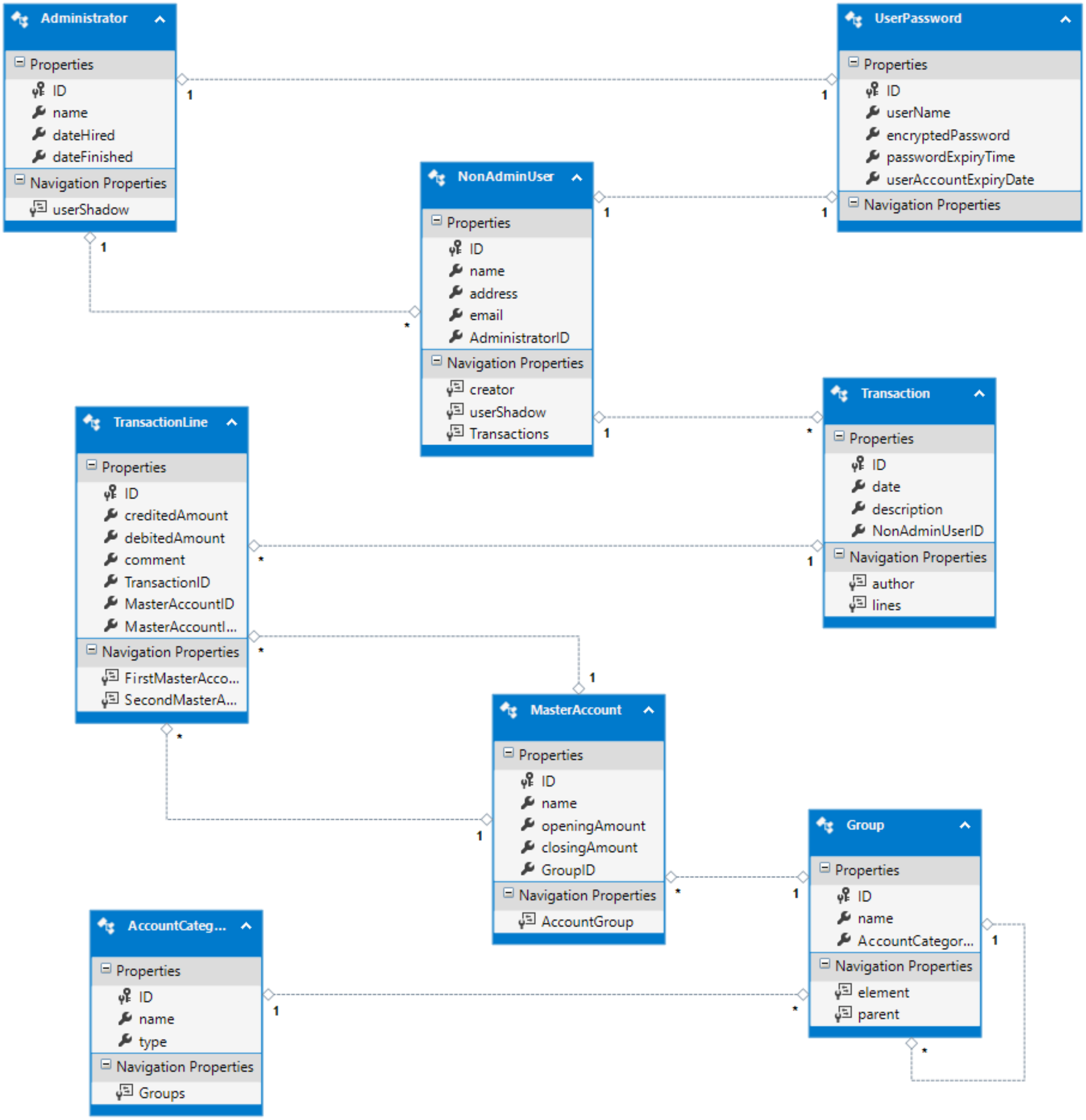
9. TransactionLine Class

Attribute name	Type	Brief Description
ID	String	This is the primary key of the TransactionLine class.
creditedAmount	double	The creditedAmount attribute stores the credited amount of the transaction.
debitedAmount	double	The debitedAmount attribute stores the debited amount of the transaction.
comments	String	The comments attribute stores the detail comments about the transaction.



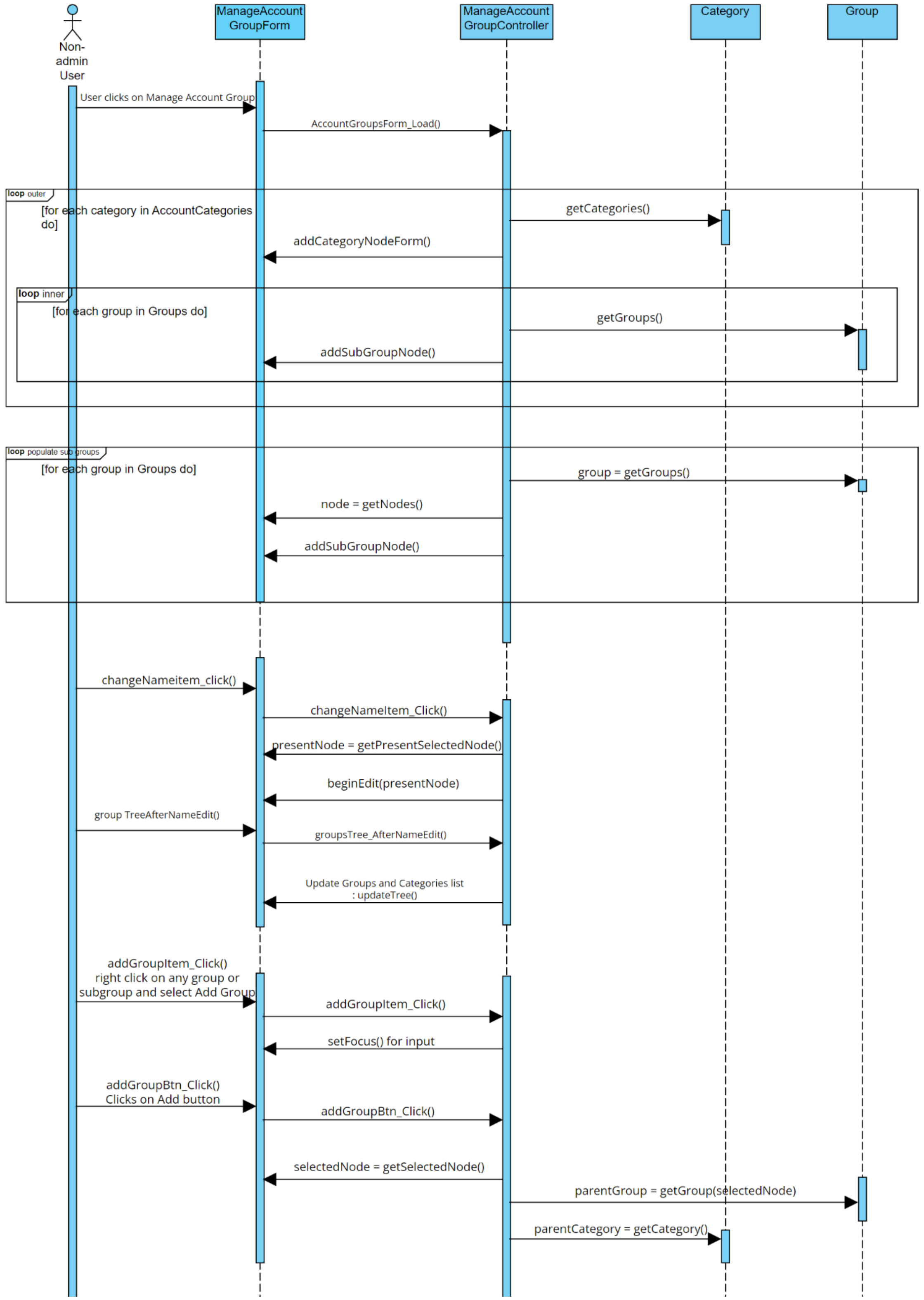
package iFINANCE[Analysis Model]
iFINANCE User
attributes
-ID : String
-Name : String
Has
-userShadow
1
1
UserPasswrd
attributes
-ID : String
-userName : String
-encryptedPassword
-passwordExpiryTime : int
-userAccountExpiryDate : date
Administrator
attributes
-dateHired : date
-dateFinished : date
-creator
1
Created by
*
NonAdminUser
attributes
-address : String
-email : String
-author
1
*
TransactionLine
attributes
-ID : String
-creditedAmount : double
-debitedAmount : double
-comment : String
-line
1..*
1
Transaction
attributes
-ID : String
-date : date
-description : String
*
*
-firstMasterAccount
1
MasterAccount
attributes
-ID
-name
-openingAmount
-closingAmount
-secondMasterAccount
1
-accountGroup
*
1
Group
attributes
-ID : String
-name : String
-parent
1
*
Contains four predefined account categories (Assets, Liabilities, Income, and Expenses
AccountCategory
attributes
-ID
-name
-type
-element
1
Type of
*

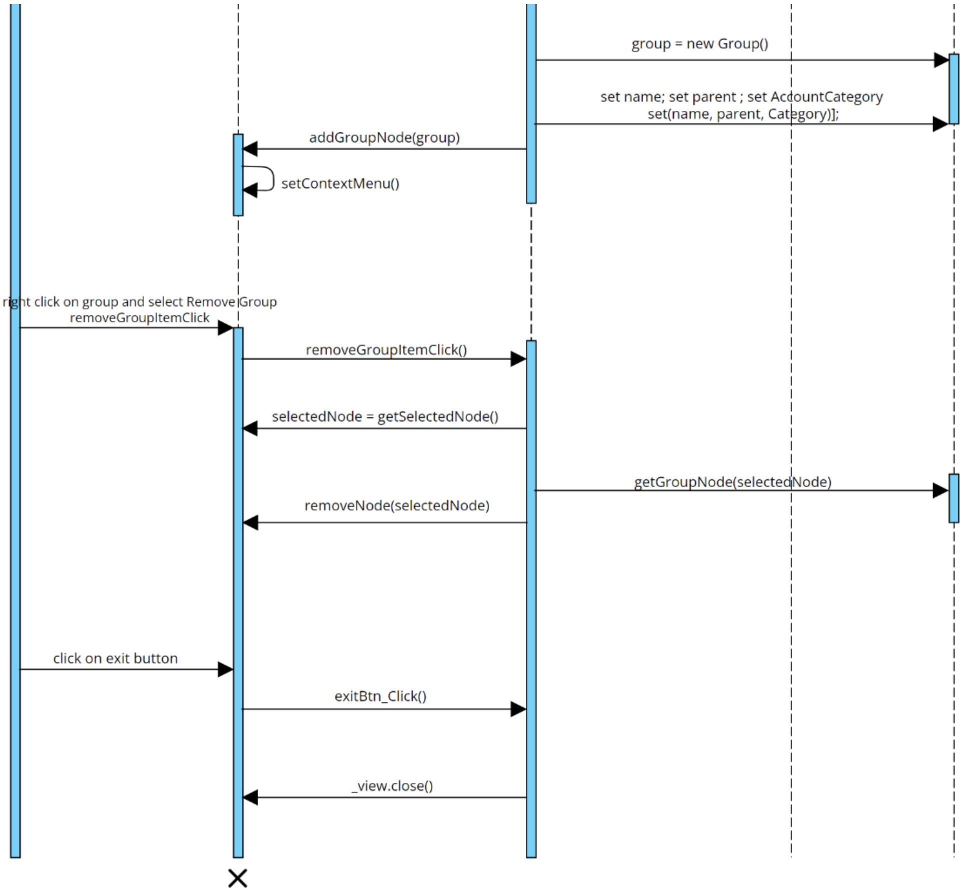
Modeling and Mapping



Potential iFinance Sequence Diagram

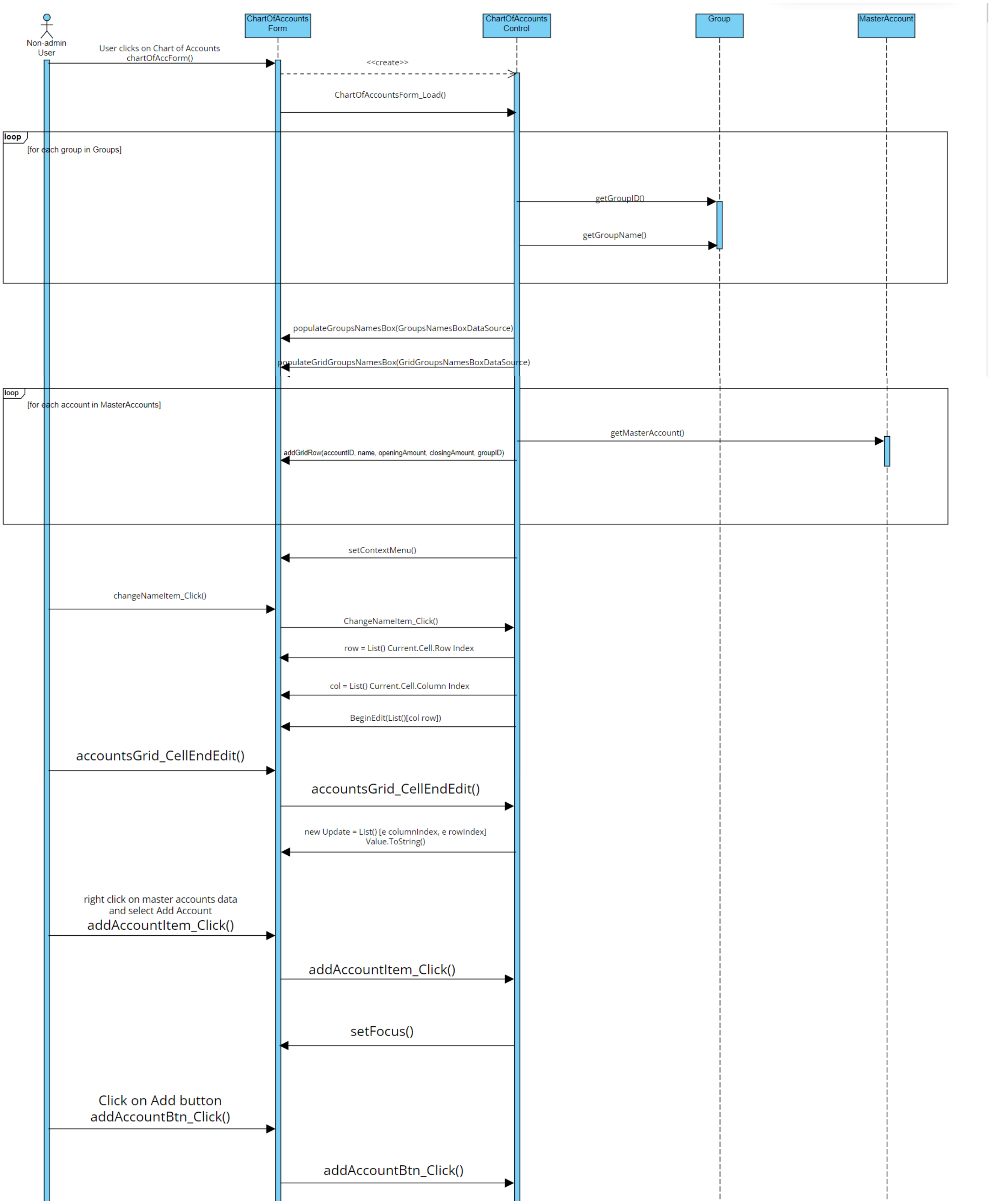
Manage Account Groups

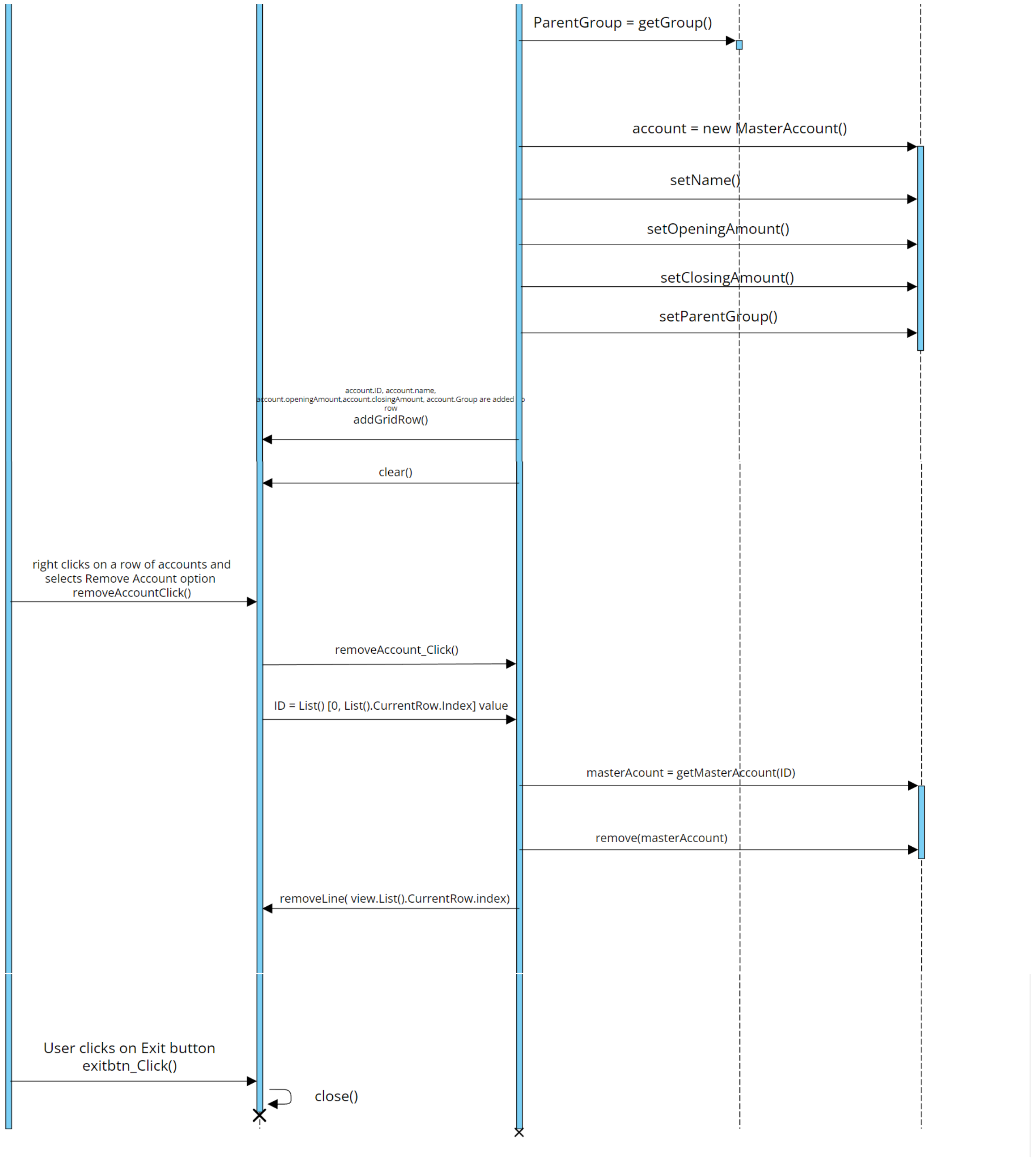

Non-admin User
ManageAccount GroupForm
ManageAccount GroupController
Category
Group
User clicks on Manage Account Group
AccountGroupsForm_Load()
loop outer
[for each category in AccountCategories do]
getCategories()
addCategoryNodeForm()
loop inner
[for each group in Groups do]
getGroups()
addSubGroupNode()
loop populate sub groups
[for each group in Groups do]
group = getGroups()
node = getNodes()
addSubGroupNode()
changeNameitem_click()
changeNameItem_Click()
presentNode = getPresentSelectedNode()
beginEdit(presentNode)
group TreeAfterNameEdit()
groupsTree_AfterNameEdit()
Update Groups and Categories list
: updateTree()
addGroupItem_Click()
right click on any group or
subgroup and select Add Group
addGroupItem_Click()
setFocus() for input
addGroupBtn_Click()
Clicks on Add button
addGroupBtn_Click()
selectedNode = getSelectedNode()
parentGroup = getGroup(selectedNode)
parentCategory = getCategory()



group = new Group()
set name; set parent ; set AccountCategory
set(name, parent, Category)];
addGroupNode(group)
setContextMenu()
right click on group and select Remove Group
removeGroupItemClick
removeGroupItemClick()
selectedNode = getSelectedNode()
getGroupNode(selectedNode)
removeNode(selectedNode)
click on exit button
exitBtn_Click()
_view.close()

Chart of Accounts


Non-admin User
ChartOfAccounts Form
ChartOfAccounts Control
Group
MasterAccount
User clicks on Chart of Accounts
chartOfAccForm()
<<create>>
ChartOfAccountsForm_Load()
loop
[for each group in Groups]
getGroupID()
getGroupName()
populateGroupsNamesBox(GroupsNamesBoxDataSource)
populateGridGroupsNamesBox(GridGroupsNamesBoxDataSource)
loop
[for each account in MasterAccounts]
getMasterAccount()
addGridRow(accountID, name, openingAmount, closingAmount, groupID)
setContextMenu()
changeNameItem_Click()
ChangeNameItem_Click()
row = List() Current.Cell.Row Index
col = List() Current.Cell.Column Index
BeginEdit(List()[col row])
accountsGrid_CellEndEdit()
accountsGrid_CellEndEdit()
new Update = List() [e columnIndex, e rowIndex]
Value.ToString()
right click on master accounts data
and select Add Account
addAccountItem_Click()
addAccountItem_Click()
setFocus()
Click on Add button
addAccountBtn_Click()
addAccountBtn_Click()



ParentGroup = getGroup()
account = new MasterAccount()
setName()
setOpeningAmount()
setClosingAmount()
setParentGroup()
account.ID, account.name,
account.openingAmount,account.closingAmount, account.Group are added to
row
addGridRow()
clear()
right clicks on a row of accounts and
selects Remove Account option
removeAccountClick()
removeAccount_Click()
ID = List() [0, List().CurrentRow.Index] value
masterAcount = getMasterAccount(ID)
remove(masterAccount)
removeLine(view.List().CurrentRow.index)
User clicks on Exit button
exitbtn_Click()
close()

iFinance Sequence Diagrams Methods

1. Manage Account Groups

Class Name	Receiving Message (operation)	Brief Description
ManageAccountGroupsForm	AccountGroupsForm_Load()	Used to create AccountGroupsForm object and load the data onto _view
	addCategoryNode()	Creates a node in form using the key and data sent.
	addSubGroupNode()	Creates a node and adds it to a group node using key, data and parent group id.
	getNodes(groupID)	Retrieves group based on group ID.
	changeNameItem.Click()	Called when change name option is selected in right click menu
	groupsTree.AfterLabelEdit()	After edit is done,
	getselectedNode()	Retrieves selected group based on where user clicked in UI
	BeginEdit(currentNode)	Moves focus to input box
	updateTree()	This function is called after a label edit to update it in the list view.
	addGroupItem.Click()	Moves focus to input box to enter group name.
	addGroupBtn.Click	Called when user clicks on Add button to create and save the group.
	addGroupNode(g)	Uses data and key to create node and add context menu strip to it.
	SetContextMenu()	A recursive method that sets the context menu for each node in the tree view (UI)
	removeGroupItem.Click	Invoked when Remove Group option is selected in the menu to delete the group.
	exitBtn.Click	This closes the created AccountGroupsForm class object using this.Close() method.
	Close()	Closes the form object after exit button is clicked.
	removeNode(node)	Removes group from list
	setFocus	This method is called to move focus to group name input box
ManageAccountGroupsController	changeNameItem_Click(object sender)	Selects the current node and initiates the editing of the node label.
	groupsTree_AFterLabelEdit(object sender)	Updates tree view after label change
	addGroupItem_Click(object sender)	Moves focus to add group name input box.
	addGroupBtn_Click(object sender)	Called when add group button is clicked. Checks whether provided group name is valid using regex and then creates Group object to store

		information and save changes and update Tree view in form.
	removeGroupItem_Click(object sender)	Called when Remove Group option is selected. Checks if the item is from level 0, if yes then iteam is not removed else, item is removed and changes are saved to database.
	exitBtn_Click(object sender)	Called when exit button is clicked by user. This closes the view we created using AccountGroupsForm_Load function.
Category	getCategories()	Lists all the categories stored in database
Group	setGroup()	Sets group name with user input "name" for group created
	getGroup()	Gets group name when controller object requests group data

2. Chart of Accounts

Class Name	**Receiving Message (operation)**	**Brief Description**
ChartofAccountsForm	ShowLoginForm()	The user will be displayed an authentication form, where login details need to enter and submitted.
	populateGroupsNamesBox(GroupNamesBoxDataSource)	Populates acGroupNameBox with list of Group objects
	populateGridgroupsNamesBox(GridGroupsNamesBoxDataSource)	Populates acGridGroupName with list of group objects
	addGridRow()	Creates a new row to the DataGridView with specified values and sets its context menu
	changeNameItem.Click()	Called when change name option is selected in right click menu to edit account name, group, openingAmount etc., enabling editing on the cell.
	setContextMenu()	Sets context menu for the each row in the 'DataGridView'
	List().CurrentCell.RowIndex	Gets the row index for the selected row in the accounts UI
	List().CurrentCell.ColumnIndex	Gets the column index for the selected column in the accounts UI
	BeginEdit(List()[col,row])	Based on the column and row, BeginEdit sets focus for input.
	addAccountItem_Click()	
	setFocus()	Moves focus to the 'acAccountNameBox1' input box.
	addAccountBtn_Click()	
	Clear()	Clears acOAmount and acName inputs in UI after Account is added using addAccountBtn_Click()

	removeAccountItem_Click()	Removes a selected account from the iFinanceModel and the Chart of Accounts form.
	removeLine(_view.List().CurrentRow.Index)	Removes the row at the specified index from the ‘DataGridView’ i.e., deleted account data.
	exitBtn_Click()	Is called when user clicks the exit button to close the form
	Close()	This is inturn called by exitBtn_Click() to remove the ChartOfAccounts form object from memory.
ChartOfAccountsControl	ChartOfAccountsForm_Load()	Loads the chart of accounts form and populates it with rows of accounts stored in database using iFinanceModel. It also sets up the context menu for the form
	accountsGrid_CellEndEdit()	Handles the end of editing for a cell in the accounts grid and updates the corresponding data in the iFinanceModel.
	addAccountItem_Click()	Adds a new account to the Chart of Accounts form
	addAccountBtn_Click()	Handles the user clicking the “Add account” button in the form and adds a new account to the iFinanceModel and the Chart of Accounts form.
	removeAccount_Click()	Removes a selected account from the iFinance Model and chart of accounts form.
MasterAccount	getAccount()	Gets the details of the selected account
	setAccount()	Sets the details for the account to be created using users inputs like account name, openingAmount and group.