Student:Noja Alex

**Group:30233**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

The application is designed for the front-desk employee`s of a bank. In this way, they will manage the clients and their accounts. Also there will be an administrator which will manage the employees.

# Functional Requirements

The regular user can do the following: CRUD on client info, CRUD on client account, transfer money between accounts, process utilities bills and the administrator can do CRUD on employee`s.

# Non-functional Requirements

2. Use-Case Model

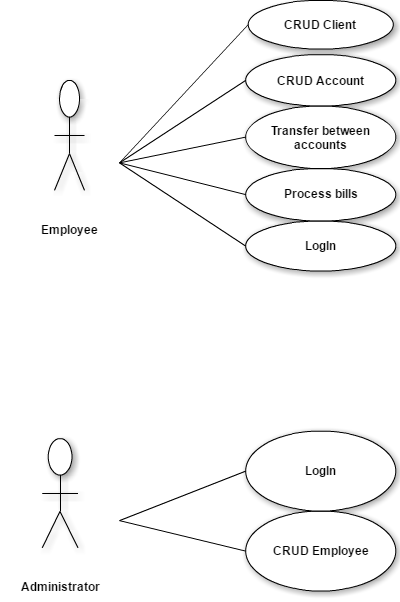
Use case: <transfer money between accounts>

Level: <user-goal level>

Primary actor: <regular user (employee)>

Main success scenario: <the user successfully logins, and then transfers the money between accounts>

Extensions: <the user wrongly introduce his credentials, after that he finally login, but the amount he wants to transfer is smaller the the amount of the account. An error message is displayed>



3. System Architectural Design

**3.1 Architectural Pattern Description**

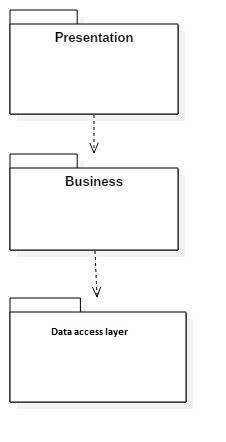
Three-tier architecture is a client-server software architectural pattern in which the user interface,functional process logic,data access and computer data storage are developed and maintained as independent modules.

Presentation logic is about how to handle the interaction between the user and the software. This can be as simple as a command-line or text-based menu

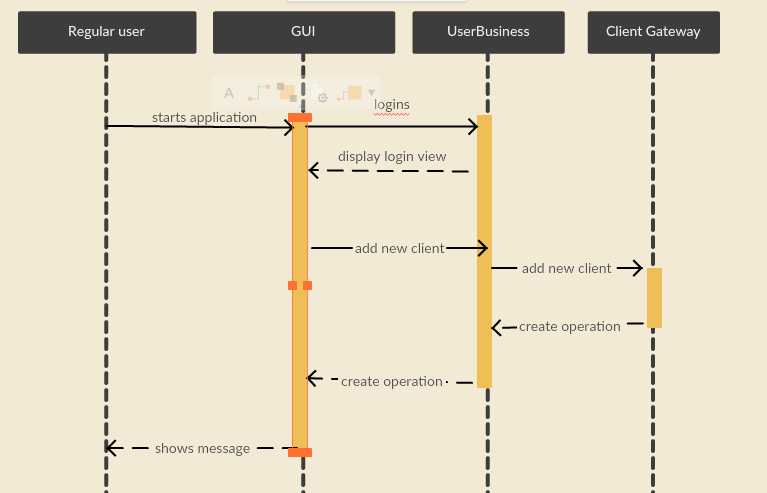
Data source logic is about communicating with other systems that carry out tasks on behalf of the application. These can be transaction monitors, other applications, messaging systems, and so forth. For most enterprise applications the biggest piece of data source logic is a database that is primarily responsible for storing persistent data.

The remaining piece is the domain logic, also referred to as business logic. This is the work that this application needs to do for the domain you’re working with.

**3.2 Diagrams**

****

4. UML Sequence Diagrams



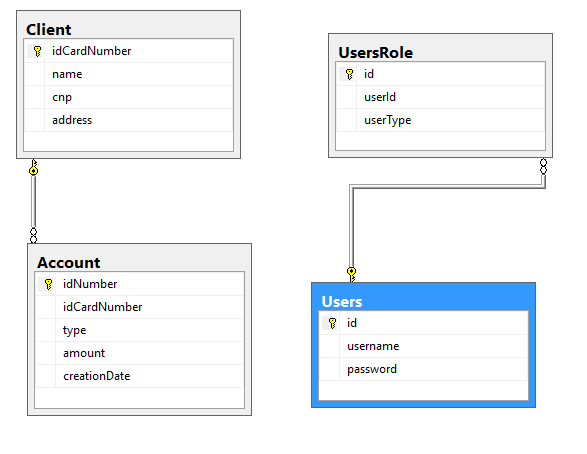
5. Class Design

**5.1 Design Patterns Description**

In “Active data record”, an object carries both data and behavior. Much of this data is persistent and needs to be stored in a database. Active Record uses the most obvious approach, putting data access logic in the domain object. This way all people know how to read and write their data to and from the database.

Table Data Gateway holds all the SQL for accessing a single table or view: selects, inserts, updates, and deletes.

6. Data Model

**

7. System Testing

For the main operations the system supports tests: delete, update, create, retrieve etc. If something is going wrong the application send an error message to inform the user. In particular, I tested the outcome of the following operations:

1. on login button click, from login view – this takes the user to admin view or user view depending on the introduced credentials
2. on create, list, update, logout, delete, from admin vie
3. on create, update, view, logout, delete, transfer buttons click, from user view

8. Bibliography

http://tutorialspoint.com/

https://www.w3schools.com/

<https://stackoverflow.com/>