FastPay

Analysis and Design Document

Student: Jurj Alexandru

**Group: 30233**

Table of Contents

1. Requirements Analysis

1.1 Assignment Specification

1.2 Functional Requirements

1.3 Non-Functional Requirements

2. Use-Case Model

3. System Architectural Design

4. UML Sequence Diagrams

5. Class Design

6. Data Model

7. System Testing

8. Bibliography

1. Requirements Analysis

# Assignment Specification

The application is used for managing the bank account of the National Bank clients.

# Functional Requirements

The regular user ( client ) can perform the following operations:

* + Add, update, view client information ( name, id number, personal numeric code, address, pin code, etc.).
  + Create an account ( the client can open a new account using the application)
  + Deposit/Withdraw money ( the client can deposit or withdraw money)
  + Transfer money to other user/personal account

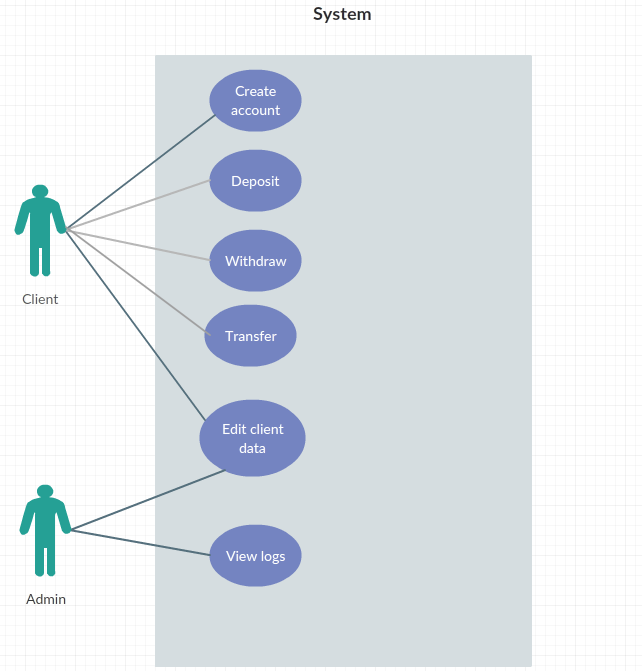
The administrator user can perform the following operations:

* + CRUD on client information.
  + View logs of client operations in details
  + Notify clients if the account balance has been changed

# Non-functional Requirements

Some of the non-functional requirements of the applications are the performance ( respond and utilization time ) , availability, security, data integrity. All data that are used for the implementation of the application are introduced using a data base. The difference between functional and non-functional requirements is that non-functional requirements describe how the system works, while functional requirements describe what the system should do.

2. Use-Case Model



Use-Case description :

* **Use case** :selection of user operation
* **Level** : use-goal level
* **Primary actor** : student or teacher
* **Main success scenario** :After the user login, he choose the type of user he is, then he choose what kind of operation he want to perform. The system returns the result to the user.
* **Extensions** : A good scenario can be the next one: the user correctly introduce he’s data into the login platform, the he choose the type of user. A fail scenario can be like this: the user does not introduce correctly he’s data and then he do not have access to the operation he want to perform, also the system is showing a fail message (“ User name or password are wrong!”).

**­­**

3. System Architectural Design

**3.1 Architectural Pattern Description**

The following applications design is based on layered architecture pattern, otherwise known as the n-tier architecture pattern.  The layered architecture pattern closely matches the traditional IT communication and organizational structures found in most companies, making it a natural choice for most business application development efforts.

Each layer of the layered architecture pattern has a specific role and responsibility within the application. For example, a presentation layer would be responsible for handling all user interface and browser communication logic, whereas a business layer would be responsible for executing specific business rules associated with the request..Those are the most common layers used :

- Presentation layer ( UI layer)

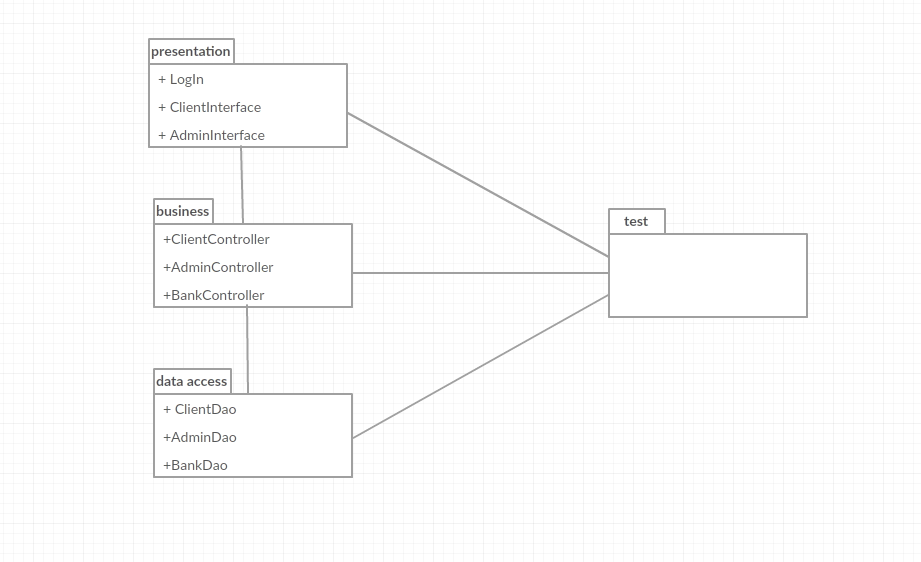
- Application layer ( service layer)

- Business logic layer ( domain layer)

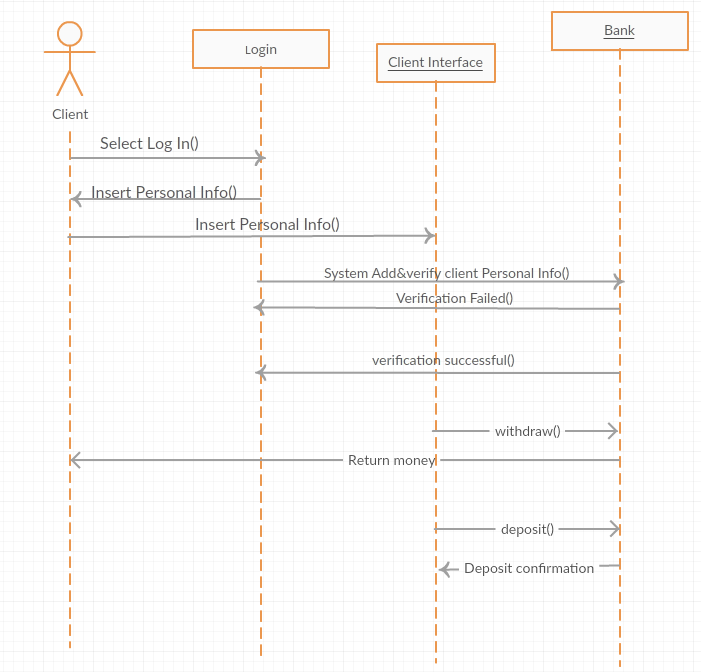
- Data access layer ( persistence layer)

* **Presentation layer**. This layer contains the user oriented functionality responsible for managing user interaction with the system, and generally consists of components that provide a common bridge into the core business logic encapsulated in the business layer.
* **Business layer**. This layer implements the core functionality of the system, and encapsulates the relevant business logic. It generally consists of components, some of which may expose service interfaces that other callers can use**.**
* **Data layer**. This layer provides access to data hosted within the boundaries of the system, and data exposed by other networked systems; perhaps accessed through services. The data layer exposes generic interfaces that the components in the business layer can consume.

**3.2Diagrams**



4. UML Sequence Diagrams



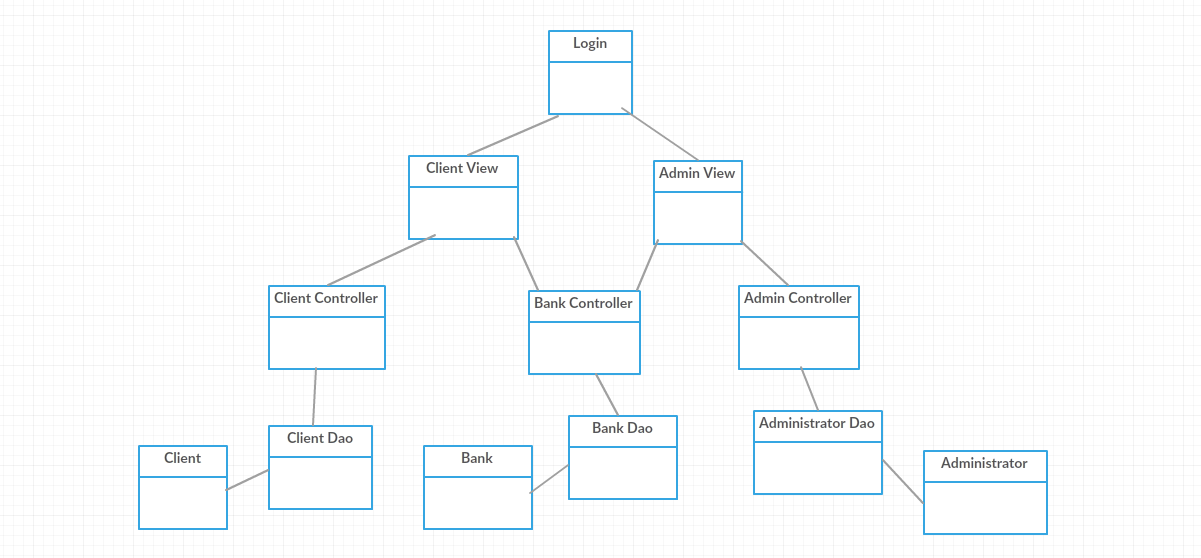
5. Class Design

**­**

**5.1 Design Patterns Description**

* **Singleton -** Ensure a class has only one instance, and provide a global point of access to it.
* **DAO -** Is used to separate the data persistence logic in a separate layer. This way, the service remains completely in dark about how the low-level operations to access the database is done.
* **Observer -** is used when there is one-to-many relationship between objects such as if one object is modified, its depenedent objects are to be notified automatically.

**5.2UML Class Diagram**



6. Data Model

7. System Testing

8. Bibliography

[1] <https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/>

[2] <https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013>

[3] <https://www.journaldev.com/1827/java-design-patterns-example-tutorial>[4] <https://en.wikipedia.org/wiki/Software_design_pattern>

[5] <https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ee658109%28v%3dpandp.10%29>