Assignment 1

Analysis and Design Document

Student: Sorin Urcan

**Group: 30442**

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 objective of this assignment is to allow students to become familiar with architectural patterns.

# Functional Requirements

Use JAVA/C# API to design and implement an application for the front desk employees of a bank. The application should have two types of users (a regular user represented by the front desk employee and an administrator user) which have to provide a username and a password in order to use the application.

The regular user can perform the following operations:

* Add/update/view client information (name, identity card number, personal numerical code, address, etc.).
* Create/update/delete/view client account (account information: identification number, type, amount of money, date of creation).
* Transfer money between accounts.
* Process utilities bills.

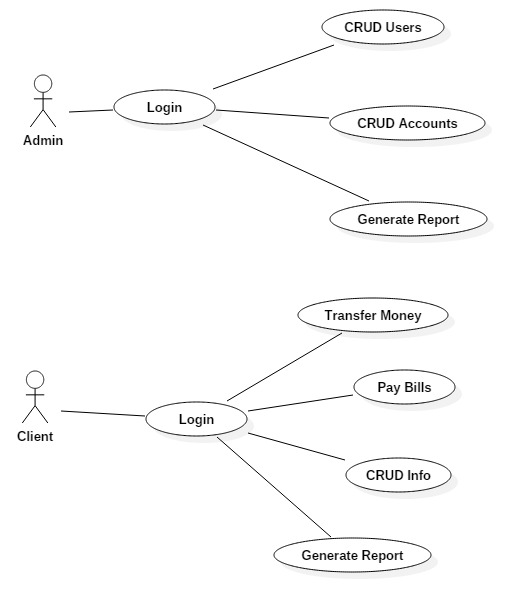
The administrator user can perform the following operations:

* CRUD on employees’ information.
* Generate reports for a particular period containing the activities performed by an employee.

# Non-functional Requirements

1. The data will be stored in a database. Use the Layers architectural pattern to organize your application. Use a domain logic pattern (transaction script or domain model) / a data source hybrid pattern (table module, active record) and a data source pure pattern (table data gateway, row data gateway, data mapper) most suitable for the application
2. All the inputs of the application will be validated against invalid data before submitting the data and saving it in the database.

2. Use-Case Model



*Use case: Add User*

*Level: user-goal level*

*Primary actor: Admin*

*Main success scenario:*

- Complete Username and Password fields

- Successfully log in into the Assignment 1 application

- A table containing user fields to complete will appear

- Click the Add User button

- User is created and displayed

*Extensions:*

- Wrong Username or Password will not allow you to enter the application

- Empty user fields will not allow you to add new user

*Use case: Remove User*

*Level: user-goal level*

*Primary actor: Admin*

*Main success scenario:*

- Complete Username and Password fields

- Successfully log in into the Assignment 2 application

- A table containing user fields to complete will appear

- Click the Remove User button

- User is deleted and no longer displayed

*Extensions:*

- Wrong Username or Password will not allow you to enter the application

- Empty user fields will not allow you to delete user

*Use case:* *Generate Report*

*Level: user-goal level*

*Primary actor: Admin*

*Main success scenario:*

- Complete Username and Password fields

- Successfully log in into the Assignment 2 application

- A table containing user fields to complete will appear

- Click the *Generate Report* button

- Report is generated and saved into the project folder

*Extensions:*

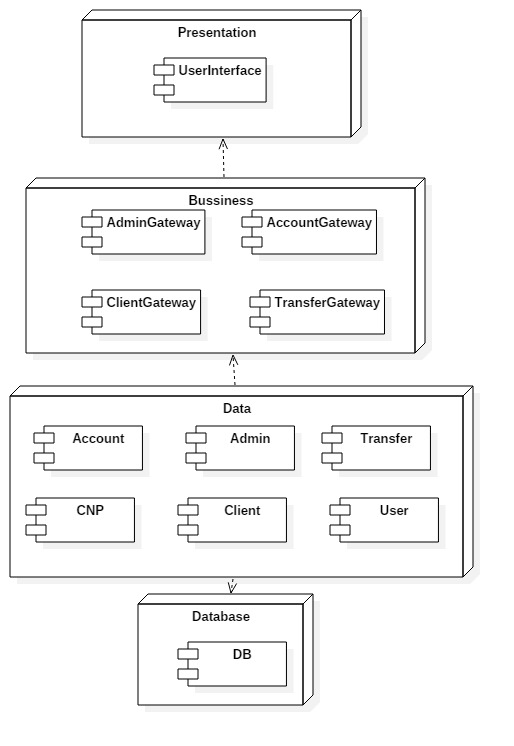
- Wrong Username or Password will not allow you to enter the application

3. System Architectural Design

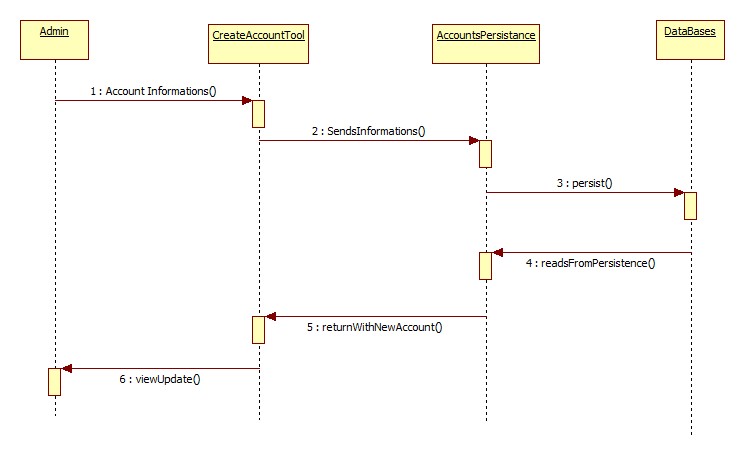
**3.1 Architectural Pattern Description**

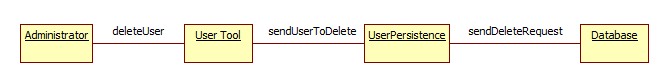
The chosen architectural pattern is the model-view-controller. It fits the requirements very well, since it divides the software into three main parts, the model, the view and the controller. For our system, the model is represented by the Book, BookList, User, UserList classes, the view is represented by the user interface and the controller is the domain logic of the system.

**3.2 Diagrams**



4. UML Sequence Diagrams





5. Class Design

**5.1 Design Patterns Description**

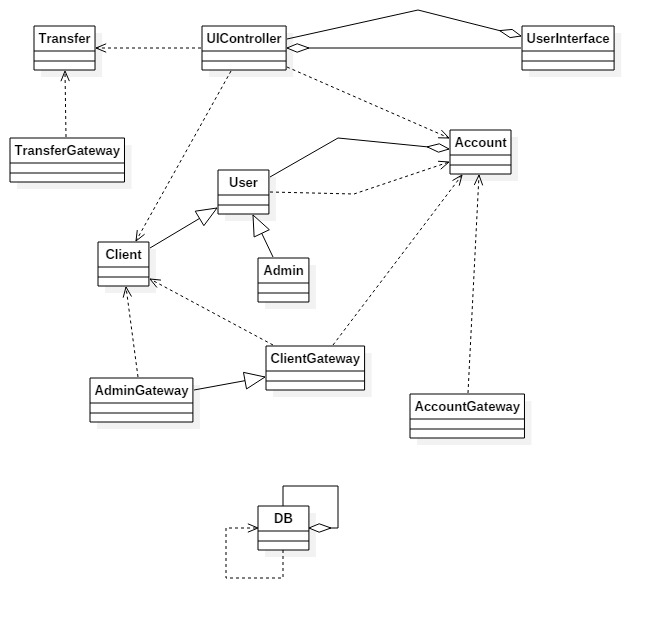
The design model builds on the analysis model by describing, in greater detail, the structure of the system and how the system will be implemented. Classes that were identified in the analysis model are refined to include the implementation constructs. The design model is based on the analysis and architectural requirements of the system. It represents the application components and determines their appropriate placement and use within the overall architecture. In the design model, packages contain the design elements of the system, such as design classes, interfaces, and design subsystems, that evolve from the analysis classes. Each package can contain any number of sub packages that further partition the contained design elements. These architectural layers form the basis for a second-level organization of the elements that describe the specifications and implementation details of the system.

The original view on the system had in mind high cohesion, low coupling, polymorphism and good abstraction of data.

The Gateway in your example is also called a "Service". The service layer is important because it provides a higher abstraction and a more "holistic" way in dealing with a Person entity.

The reason for this "extra" layer is the other objects in the system that are connected to a Person. For example, say there are Car objects and each Person may have a Car. Now, when we sell a car we should update the "owner" field, further you'll want to do the same for the Person objects that are involved (seller/buyer).

**5.2 UML Class Diagram**



6. Data Model

Data modeling is the formalization and documentation of existing processes and events that occur during application software design and development. Data modeling techniques and tools capture and translate complex system designs into easily understood representations of the data flows and processes, creating a blueprint for construction and/or re-engineering.

The system data model is represented by classes that represent the abstraction of the business logic. The model layer contains the data models that are used in the system. The data model is in the lowest level of the system hierarchy. The client is modeled by the Client class that contains all the information this system needs in order to model the interactions of a client. The client account and application user data are modeled in their respective classes. All the classes that model data are simple containers for the data they hold and they do not process in any way the data they receive. These objects are passed between modules which modify and ultimately translate them in an SQL query where database interaction is needed

7. System Testing

Unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for usage. Unit testing was not completed for this project scarce of time and resource limitations

Input validation, module testing.

8. Bibliography

<http://docs.oracle.com/javase/tutorial/uiswing/>

<http://docs.oracle.com/javase/tutorial/jdbc/basics/index.html>

<https://msdn.microsoft.com/en-us/library/54xbah2z(v=vs.110).aspx>

<https://msdn.microsoft.com/en-us/library/e80y5yhx(v=vs.110).aspx>