SOFTWARE PROJECT REPORT

CESS Junior

Online Banking System

To Be Submitted To:

Dr. Gamal Abdel Shafy

Eng. Sally Edward

Made By:

Moataz Khalid Zakaria	16P8244
Karim Walid Elhammady	16P3090
Youssef Assem Gomaa	16P6064
Mohammed Ehab Elsaeed	16P8160
Ahmed Sameh Shahin	16P6063
Ahmed Bahgat Elsayed	16P6058

TABLE OF CONTENTS

TABLE OF FIGURES AND TABLES

ABSTRACT

The purpose of this report is to identify the makings of an adequate and operational online banking system. Online banking, also known as internet banking, is an electronic payment system that enables customers of a bank or other financial institution to conduct a range of financial transactions through the financial institution's website. The project is divided into several phases in which we determine a banking system's functional requirements and derive from that the needed diagrams and findings to finally establish the project's cost estimation.

Software Project Report

ONLINE BANKING SYSTEM

REPORT INTRODUCTION

A Brief History

The concept of online banking as we know it today dates to the early 1980s, when it was first envisioned and experimented with which was only used by each company's staff. However, it was only in 1995 that Presidential Savings Bank first announced a new facility for regular client use. The idea was quickly snapped up by other banks like Wells Fargo, Chase Manhattan and Security First Network Bank. The speed with which this process happens online, as well as the other services possible by these means, has translated into a literal reverberation in the banking industry over the last decade.

Project's Intention

This Project's intention is to integrate the use of software engineered facilities into the design of a modern banking system to create a globally accessed program or banking software. The banking software is an enterprise software that is used by the banking industry. Typically banking software refers to core banking software and its interfaces that allows commercial banks to connect to other modular software and to the interbank networks. It can also refer to the trading software used by investment banks to access capital markets.

FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

Functional Requirements

Login/Logout:

Customer will be able to login/logout to his/her account if said customer has a valid account. System is then tasked to authenticate said user. Administrators and bank employees will also be able to login to the system. Upon user's logout, progress made by user should be saved, as a point for the user to return to when user logs in again. The User's last session's history also is to be saved.

View Account:

Customer can request details of the last 'n' number of transactions he has performed on any account. As well as check account's balance, additional fees that may be issued by the bank or other companies, as well as the customer's current and/or the savings account if available.

Transfer Funds:

Customer can make a funds transfer to another account in the same bank. Customer can also specify wanted amount, date to be transferred and the receiving account or the destination. Currency can also be specified if needed.

Pay bills:

Customer can view his monthly statement. She/he can also take print it out to view a list of due bills. From there they can pay registered payments, open new payments and pay or delete registration bills. Customer can also view payment history

Cheque Services:

Customer can request for cheque book, enquire about a certain cheque status, and stop a certain cheque payment if desired.

Utilities:

Utilities available for both customers and staff are going to be:

1. Change accounts password.
2. Change account personal information (user name, id, etc.…).
3. View account logs.

Non-Functional Requirements

Those requirements which are not the functionalities of a system but are the characteristics of a system are called the non-functionalities.

Secure access of confidential data. SSL (Secure Socket Layer) can be used.

24 hours a day, seven days a week availability.

Better component design to get better performance at peak time.

Flexible service-based architecture will be highly desirable for future extensions.

DECISIONS

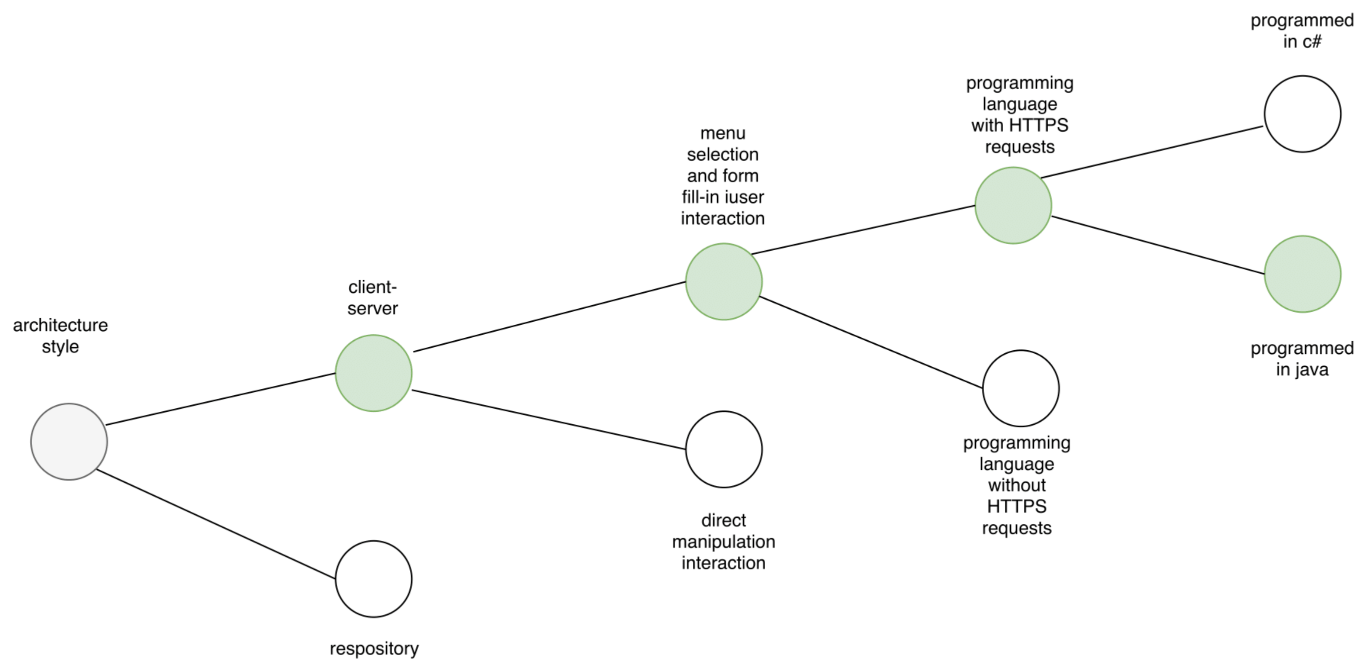

programmed in c#
programming language with HTTPS requests
menu selection and form fill-in iuser interaction
client-server
architecture style
programmed in java
programming language without HTTPS requests
direct manipulation interaction
respository

Figure 1 Decisions Diagram

Architectural style:

Why client server?

a. We need tasks and workloads to be partitioned between the providers of a resource/service (servers) and the service requesters (clients).
b. We need clients and servers to communicate over a computer network on separate hardware, yet both clients and server can reside on same system

Why Not Repository?

a. A repository architecture consists of a central data structure (often a database) and a collection of independent components which operate on the central data structure
b. Repositories are used for data integration; thus is can't be used for an online banking system software because centralized control isn't feasible.

Interface:

Why two interfaces?

> One interface wouldn't suffice. We need the interface that displays options as menu items or icons for the user to choose from, which is found in the menu selection interaction, and a lot of information has to be gathered from the user which is inaugurated using the form fill-in interaction.

Why menu and form fill-in instead of direct manipulation?

> Because operations performed by direct manipulation would be rapid, incremental, irreversible and immediately visible; thus would give the customer undesired privileges.

Programming Language:

We used a programming language with https requests for the security benefits and data privacy, to protect the users' data.

We also preferred java over c# because of java's advantages in speed, memory, programming efficiency and the available libraries.

ARCHITECTURAL STYLE

System Architecture used:

Client-Server

Alternatives Considered:

Repository Style; Rejected because centralized control wasn't feasible.

Problem:

Service should be accessible through mobile/web application over the world wide web and interaction with the DBMS servers shouldn't be through the client directly.

Context:

Languages with HTTPS requests capabilities.

Solution:

System model: modules in hierarchy, may be weak or strong, coupling/cohesion arguments.

Components: clients, main, DBMS and service-providing servers; clients interact only with main servers.

Control Structure: Multi-threaded, centralized control: each client-server connection will have a thread of its own.

Connector: procedure call.

MAIN SYSTEM MODELS

Semantic Data Model:

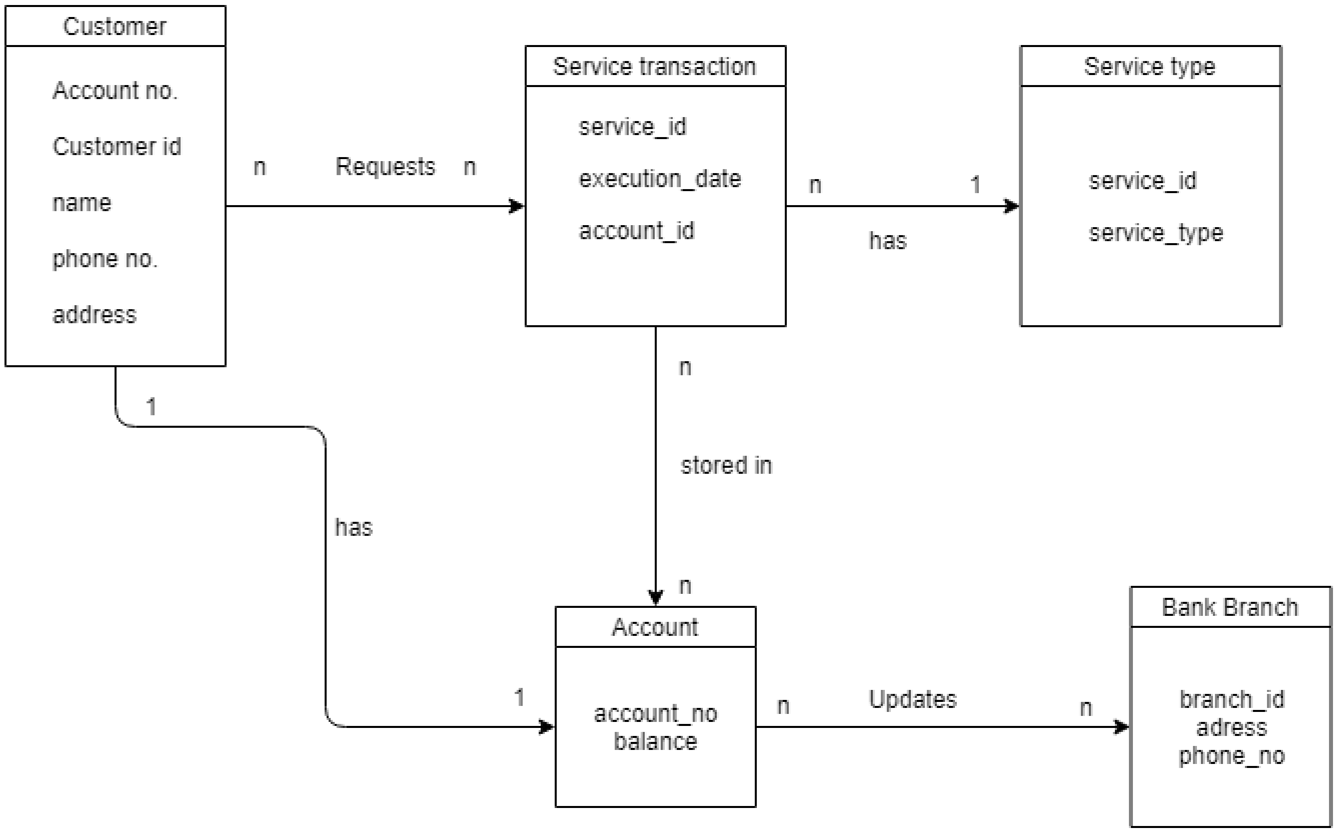


Figure 2 Semantic Data Model

The process: Customer requests n number of services whether it would be to request a loan, issue or cash a cheque, or any other transactions. Then bank verifies the customer’s request to do said service then the bank updates the customer’s account with the latest balance.

Data Dictionary Table:

Name	***Description***	***Type***
Customer	The person that requests a service from the bank	Entity
Service	Details of the requested services by the customer	Entity
Bank	The organisation responsible for verifying and granting the request made by customer	Entity
Account	Where the customer info and transaction history is saved	Entity
Verify request	A 1:n relation between bank and services meaning that the bank validates the request before granting it	Relation
Service type	Indicates the type of the service whether it is a loan, money transfer, etc..	Attribute

Figure 3 Data Dictionary Table

Object Diagram:

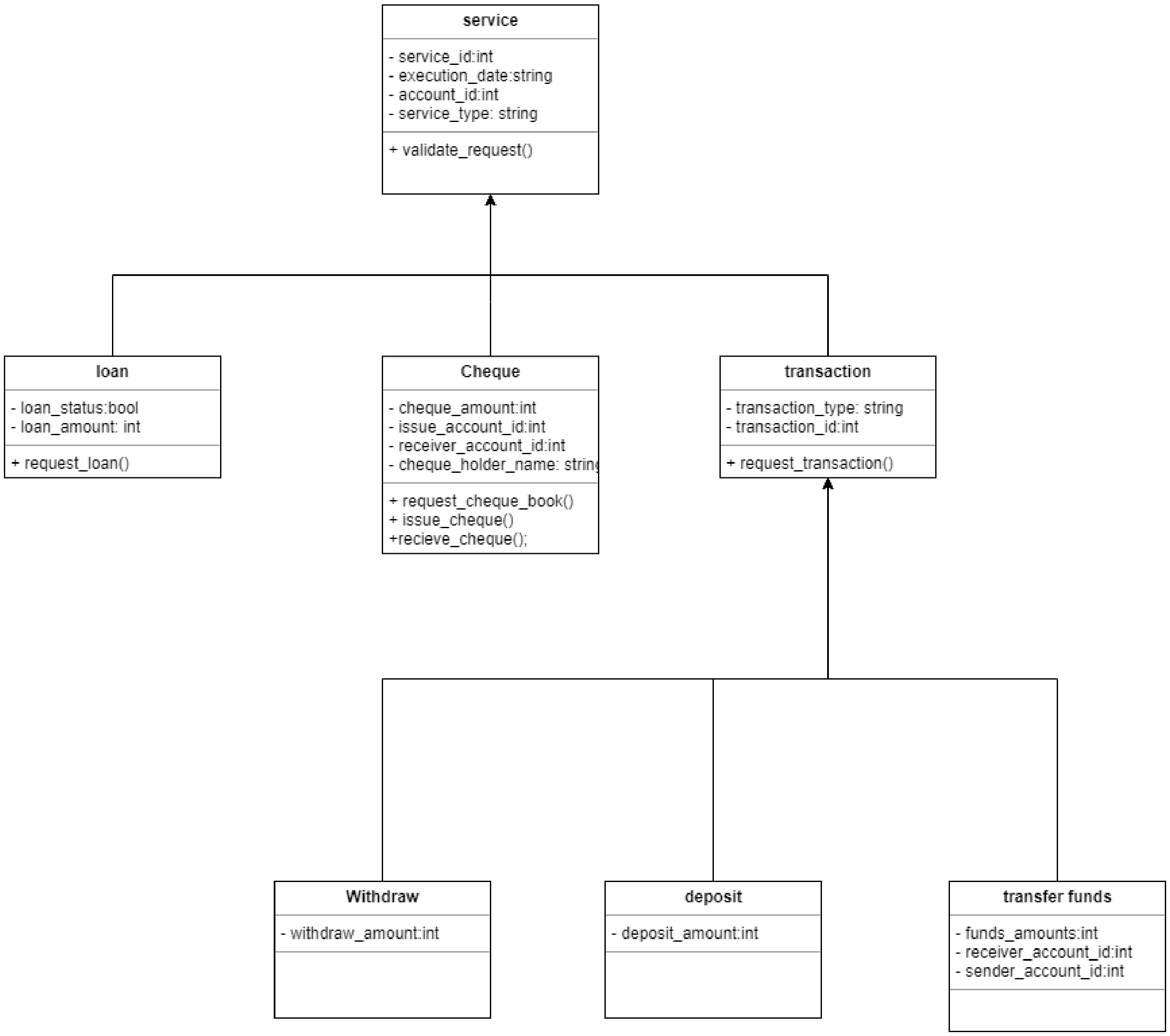


Figure 4 Object Diagram

Cheques, Loans, Transactions and all its types are all services that should be provided by the banking system, for that they inherit the class Service.

Withdraw, deposit & transfer funds all branch from transactions as they are all types of transactions performed by the customer and fulfilled by the bank.

ARCHITECTURE DESIGN

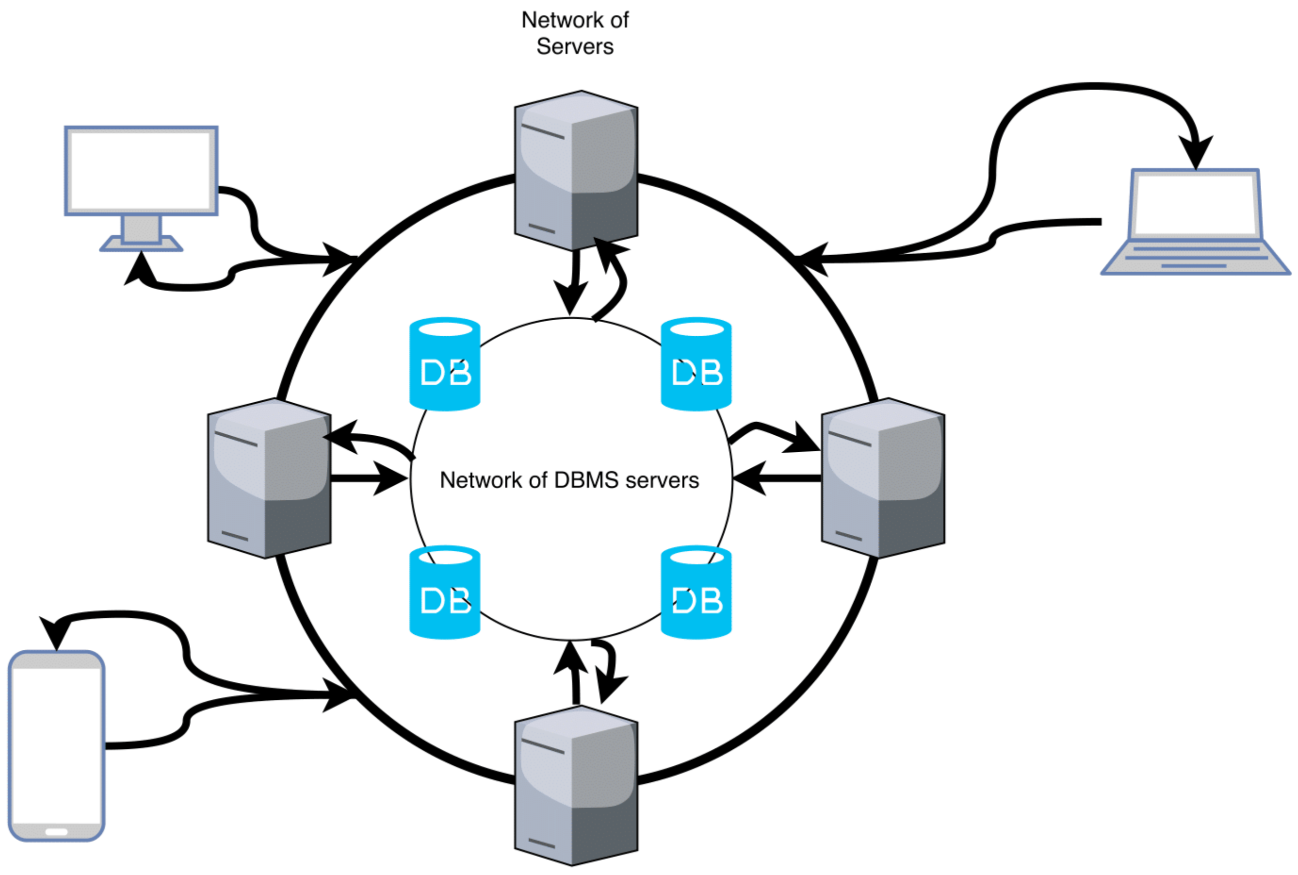


Figure 5 Architecture Design

Architectural Style: Client-Server.

COMPONENTS AND COMPONENT DIAGRAM

List of Components Used

- DBMS
- Security
- Cheque Dashboard
- Accounts Dashboard
- Bills Dashboard
- Fund Transfer Dashboard
- Settings Services
- Cheque Services
- Accounts Services
- Bills Services
- Fund Transfer Services
- Settings Services

Component Development Process

The Component Development Process: our System was developed through independent processes of development of components and composition.

Upon analysis of user requirements, needed components were determined as shown above.

Market Analysis:

- Security and DBMS components can be bought as COTS products.
- UI and Services components must be developed from scratch due to the need of customization.

Upon purchasing of third-party Security and DBMS components, they were integrated and tested for compliance with the system.

UI and Services components were designed as per user requirements, then implemented incrementally regularly checking their compliance with our design.

Upon Implementing the individual components, they were tested individually then integrated into the system and retested as a whole system of interacting components.

Finally, the System will be released and maintenance scheme will be followed to ensure integrity, security and reliability of the system.

Usable and Reusable Components

Components were designed to be as reusable as possible

- Minimal application-specific methods.
- Change names to make them general.
- Add methods to broaden coverage.
- Added a configuration interface for component

Components Interface

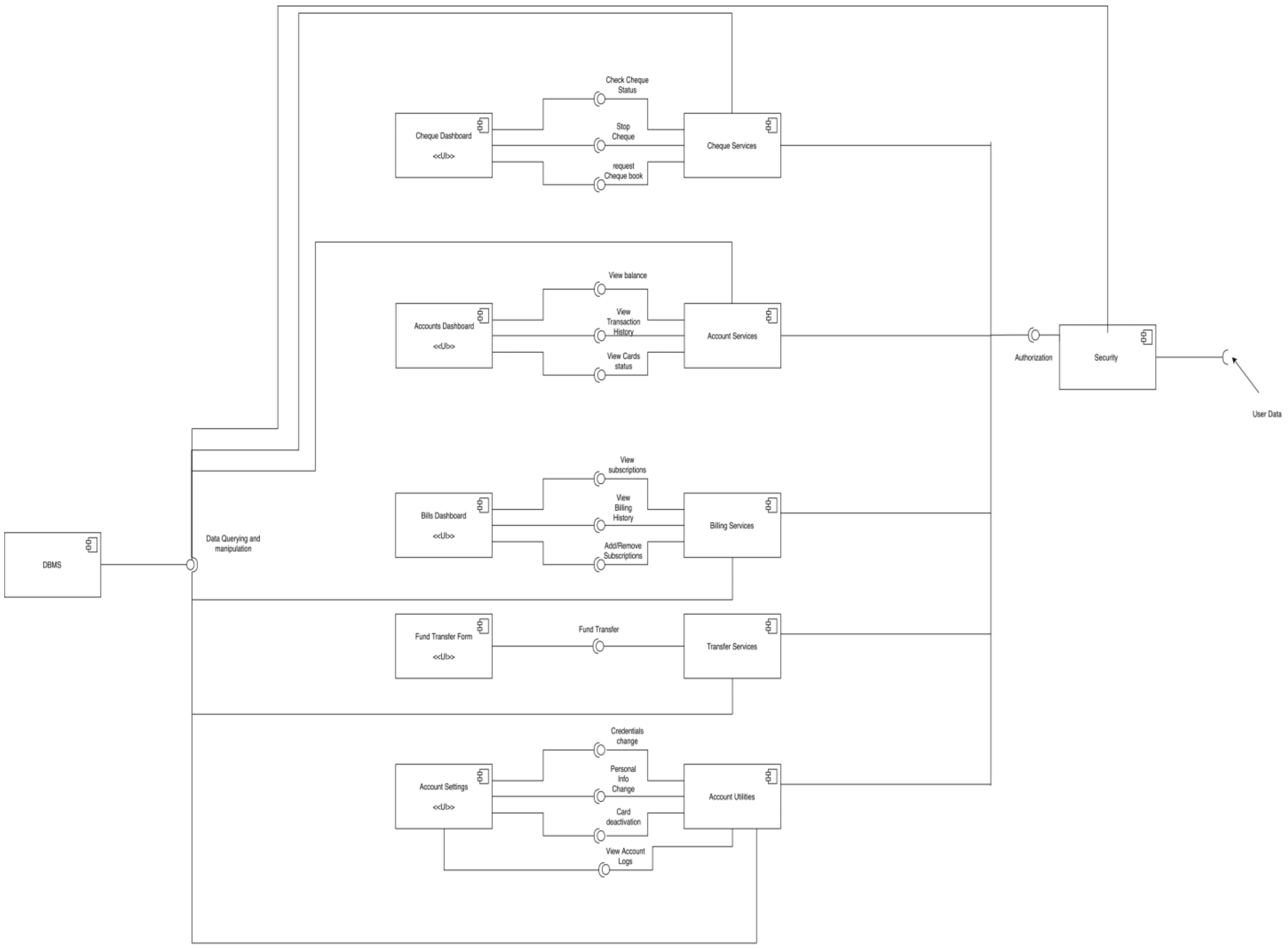


Figure 6 Component Diagram

Components' Characteristics

These components are split into 4 groups:

1. The DBMS
2. Security
3. The bank services
4. The UI

Component 1: The DBMS

The DBMS is a reusable component that manages three important things: the data, the database engine that allows data to be accessed, locked and modified, and the database schema, which defines the database's logical structure. These three foundational elements help provide concurrency, security, data integrity and uniform administration procedures. Typical database administration tasks supported by the DBMS include change management, performance monitoring/tuning and backup and recovery.

Component 2: Security

The Security component is a reusable component that provides security using encryption techniques, implementation of firewalls and virus protection spyware etc.

Component 3: Bank Services

Bank Services is a usable component that delineates the bank's facilities which are cheque services, billing services, transfer services, and account information and utilities.

Component 4: The UI

The UI is a usable component that manifests the way the user sees the banking system's application interface and allow effective operation and control of the services from the user's end, all whilst the interface simultaneously feeds back information that aids the operators' decision-making process.

Components composition

Our system uses **Sequence composition** where one component calls on the services of another. The *provides interface* of one component is composed with the *requires interface* of another.

THE STAKE HOLDERS

stake holders are groups, organizations, or individual people that have interest or concern in an organization.

The stake holders in a banking system are:

1. Architects
2. Requirements engineer
3. Designers
4. Implementors
5. integrators
6. Maintenance team
7. Bank Manager

THE BEHAVIORAL MODEL

Context Diagram:

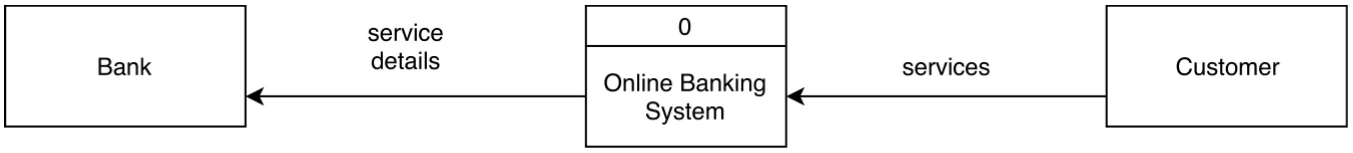


Figure 7 Context Diagram

This diagram is a basic overview of the whole system, it defines the boundary between the system and its environment, showing the entities that interact with it and the overall process it performs (Process 0).

DFD Level 0:

This level of data-flow diagram provides a more detailed breakout of pieces of the Context Level Diagram. The main functions of the system is highlighted. It also features the major services performed by the bank and requested by the customer. Data log from performing each process is saved in each corresponding data store respectively.

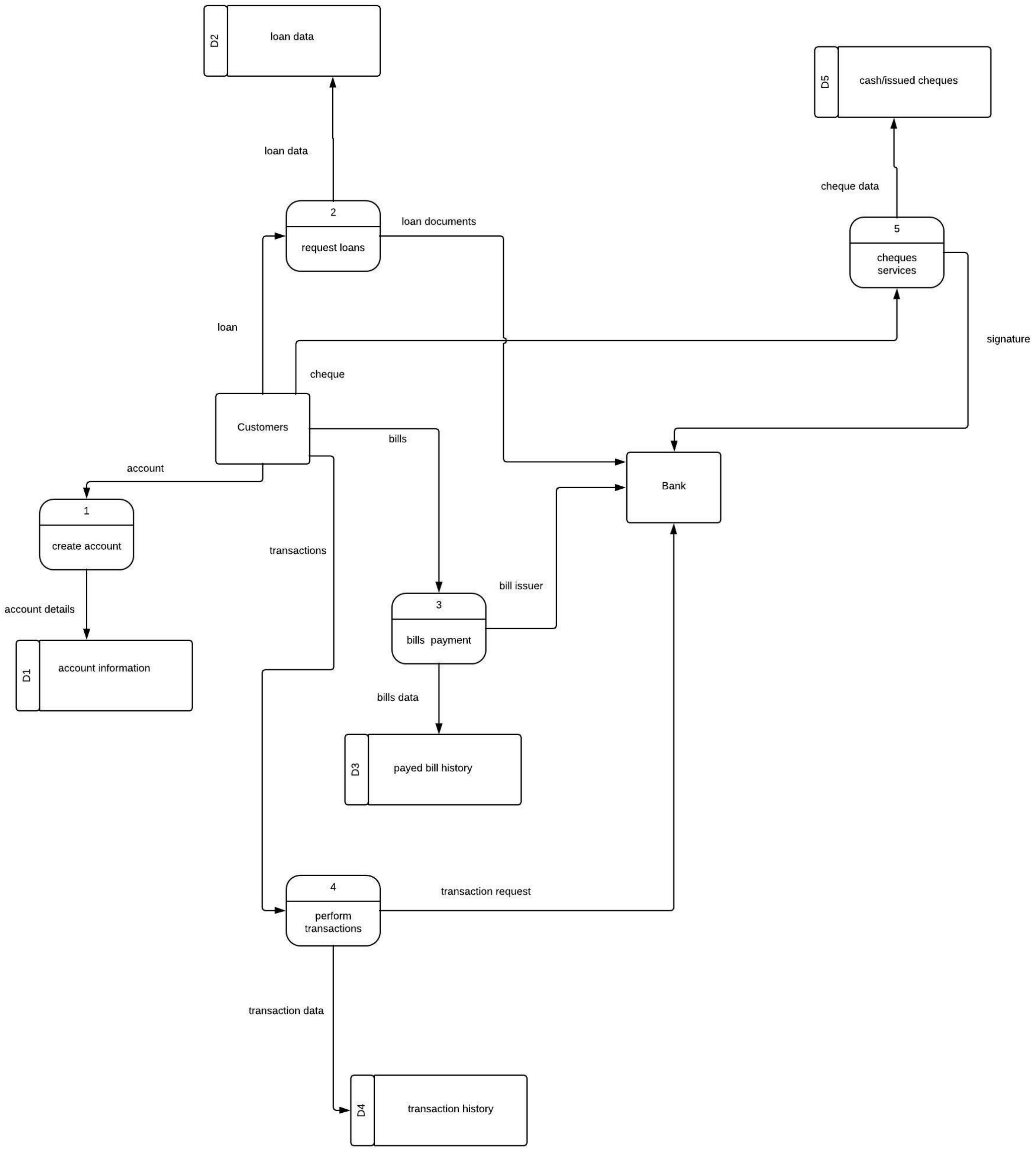


Figure 8 DFD Level 0

DFD Level 1:

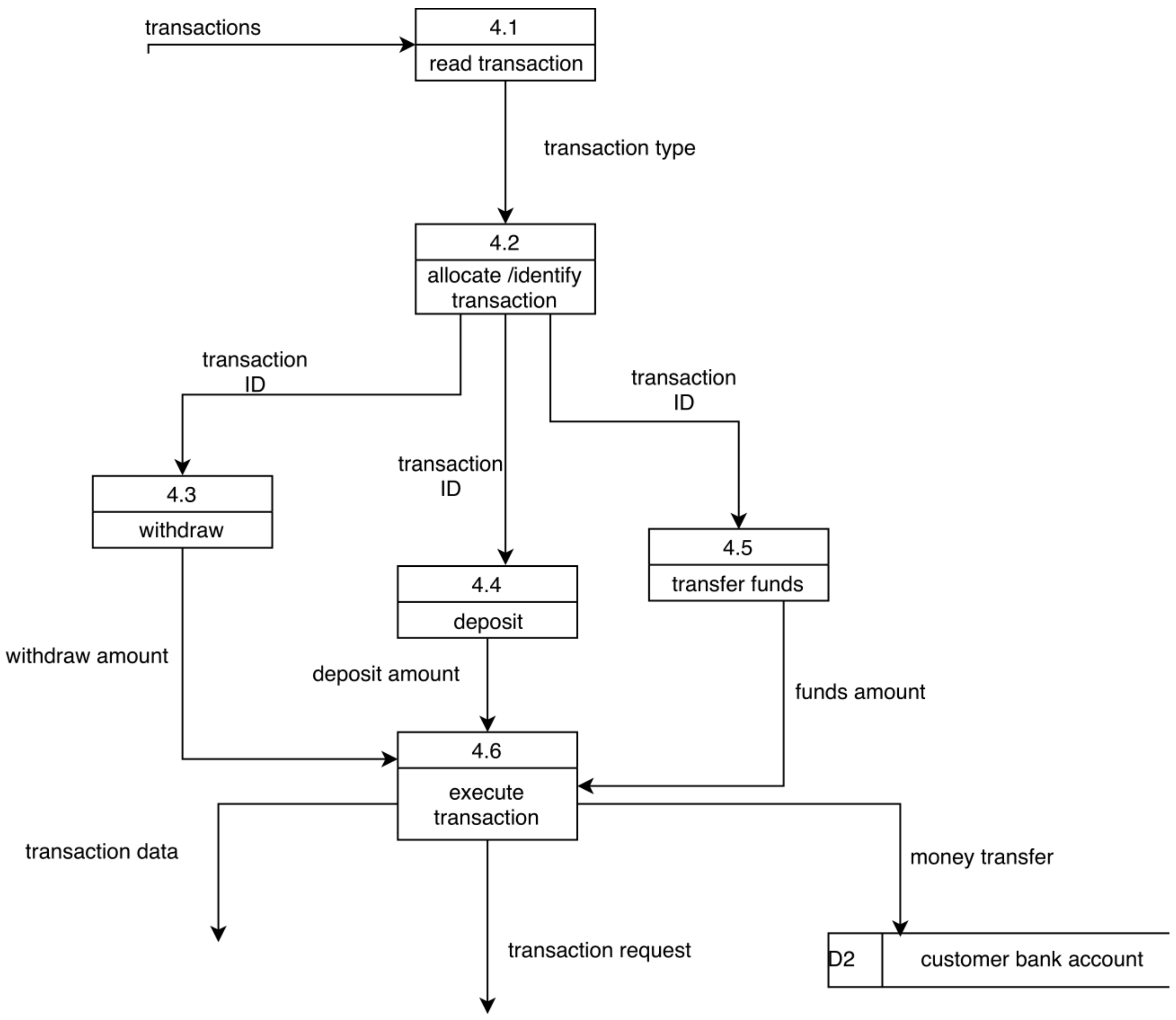


Figure 9 DFD Level 1

This level of data flow diagram then goes one step deeper into parts of Level 0. *Perform Transactions* is the process that will be broken down as its too much of vague process. Transactions branches into its three types each identified by its ID.

DFD Level 2:

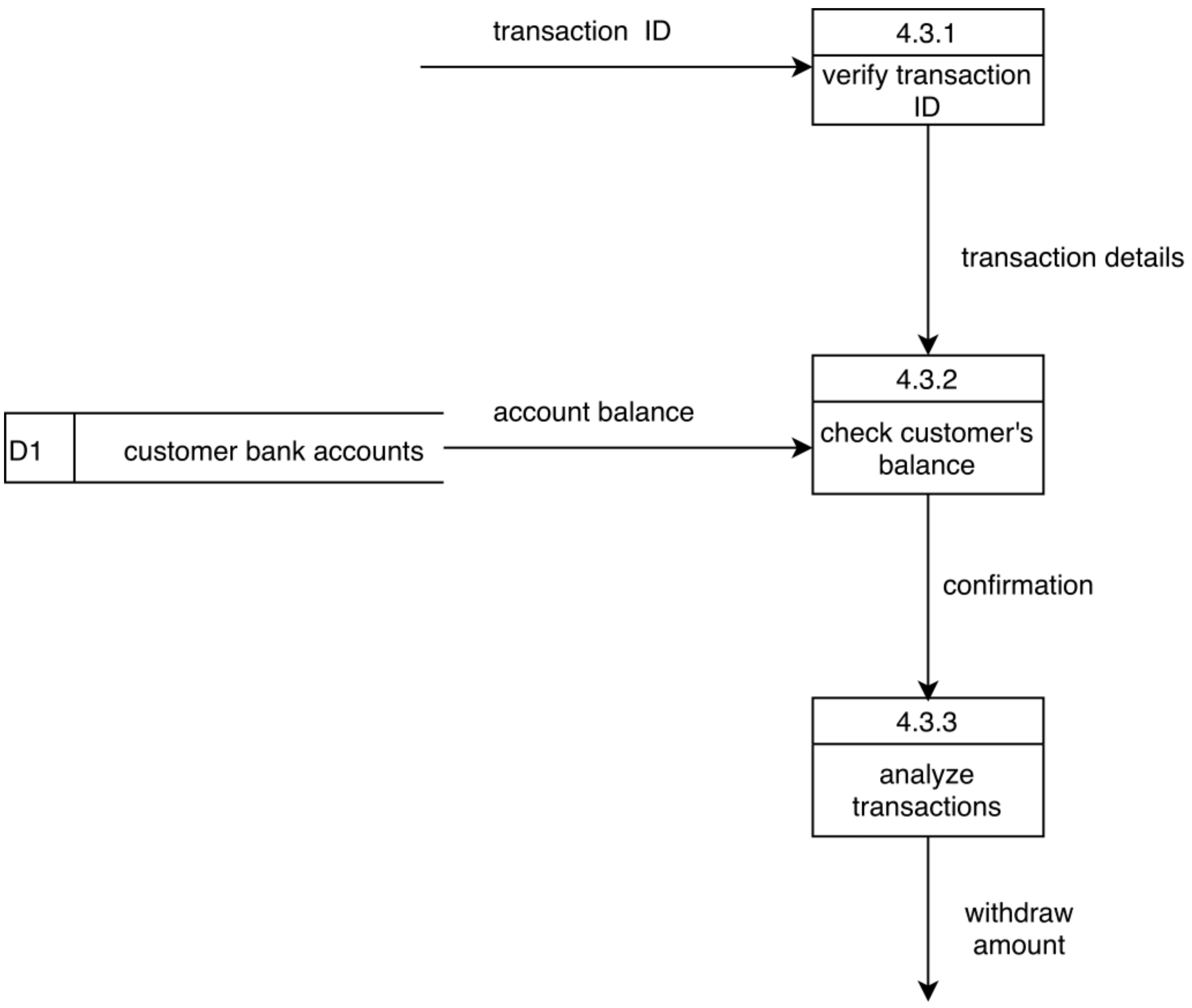


Figure 10 DFD Level 2

This level of data flow diagram also delves deeper into parts of the latter level, which is level 1. The process split in this case is withdraw service. For a customer to withdraw an amount from his account, he must have the same amount or more in his account; thus this level confirms and validates if amount is present.

The State Diagram

The Diagram:

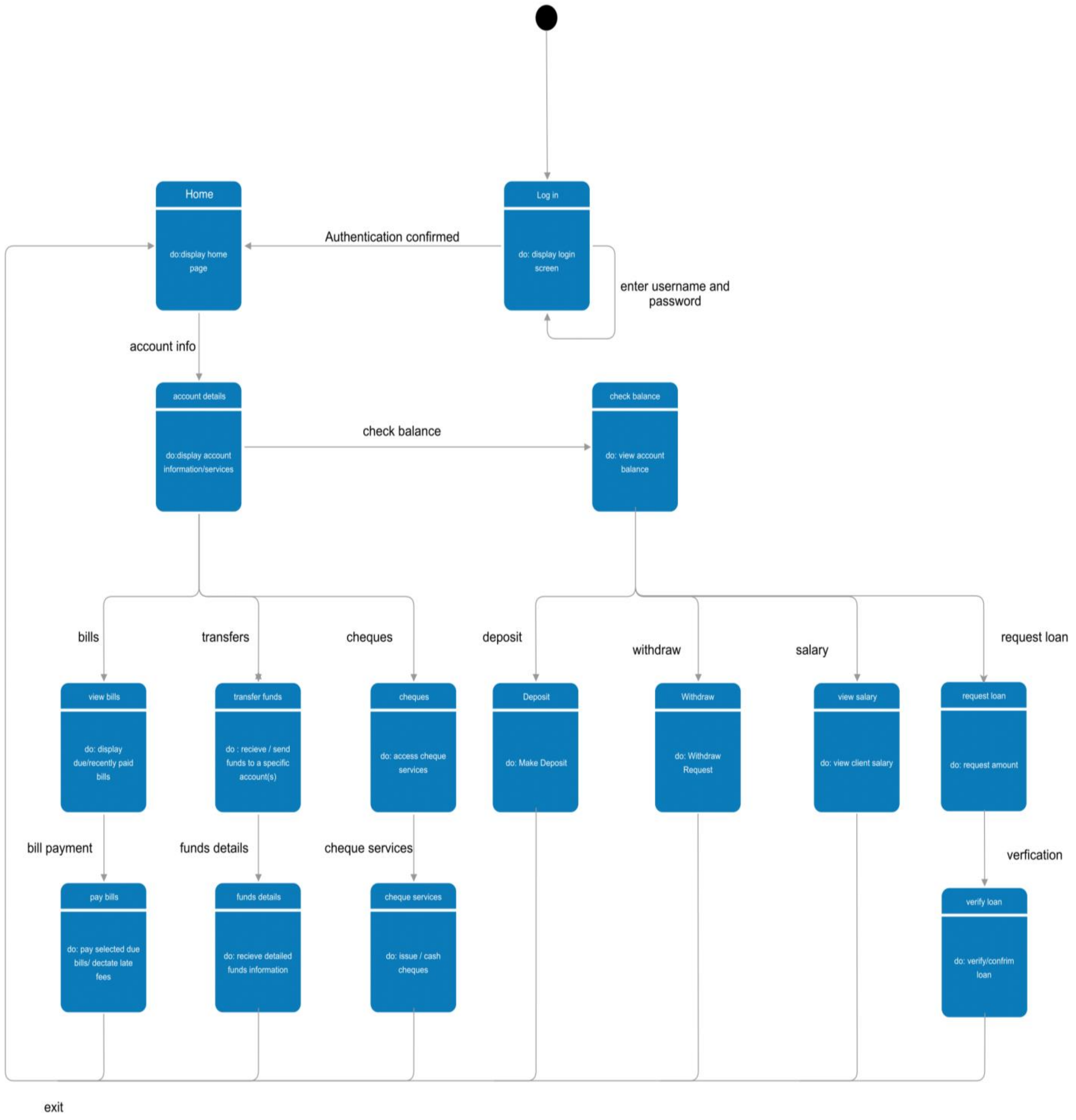


Figure 11 State Diagram

Diagram Explanation:

This diagram describes the overall behavior of the system as well as each state the system resides on at any given time. The diagram sets about from its first state, which awaits user to log in using a valid account. User is then situated in a *Home Page*, user can from there access one of many services the bank offers, though not all services provided by the bank are situated in one state, some services require a two-state process for verification such as requesting loans, paying bills, transferring funds, and all the cheque services.

The state diagram is also aiding in the explanation of the *Logical Viewpoint*.

VIEWPOINTS AND VIEW MODELS

Logical Viewpoint

This viewpoint specifies the functional requirements of the system.

Using the state diagram [Described in detail in a previous section]**, we will give a concrete description of the functional behavior of the system.**

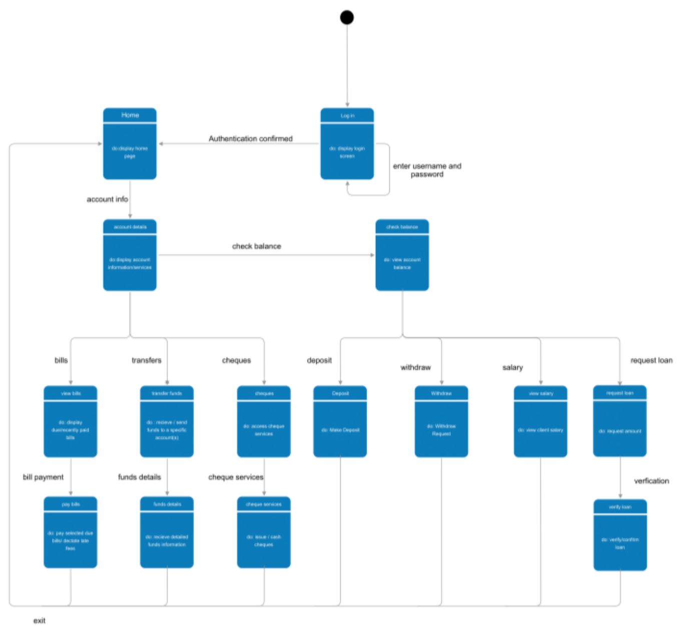


Figure 12 Logical State Diagram

This diagram displays the various services fulfilled by the bank each service satisfies its corresponding functional requirement specified at the beginning of the project.

Process Viewpoint

The viewpoint deals with the dynamic aspects of the system, explains the system processes and how they communicate, and focuses on the runtime behavior of the system. The process view addresses concurrency, distribution, integrators, performance, and scalability. All that can be represented using the process model.

Process Model:

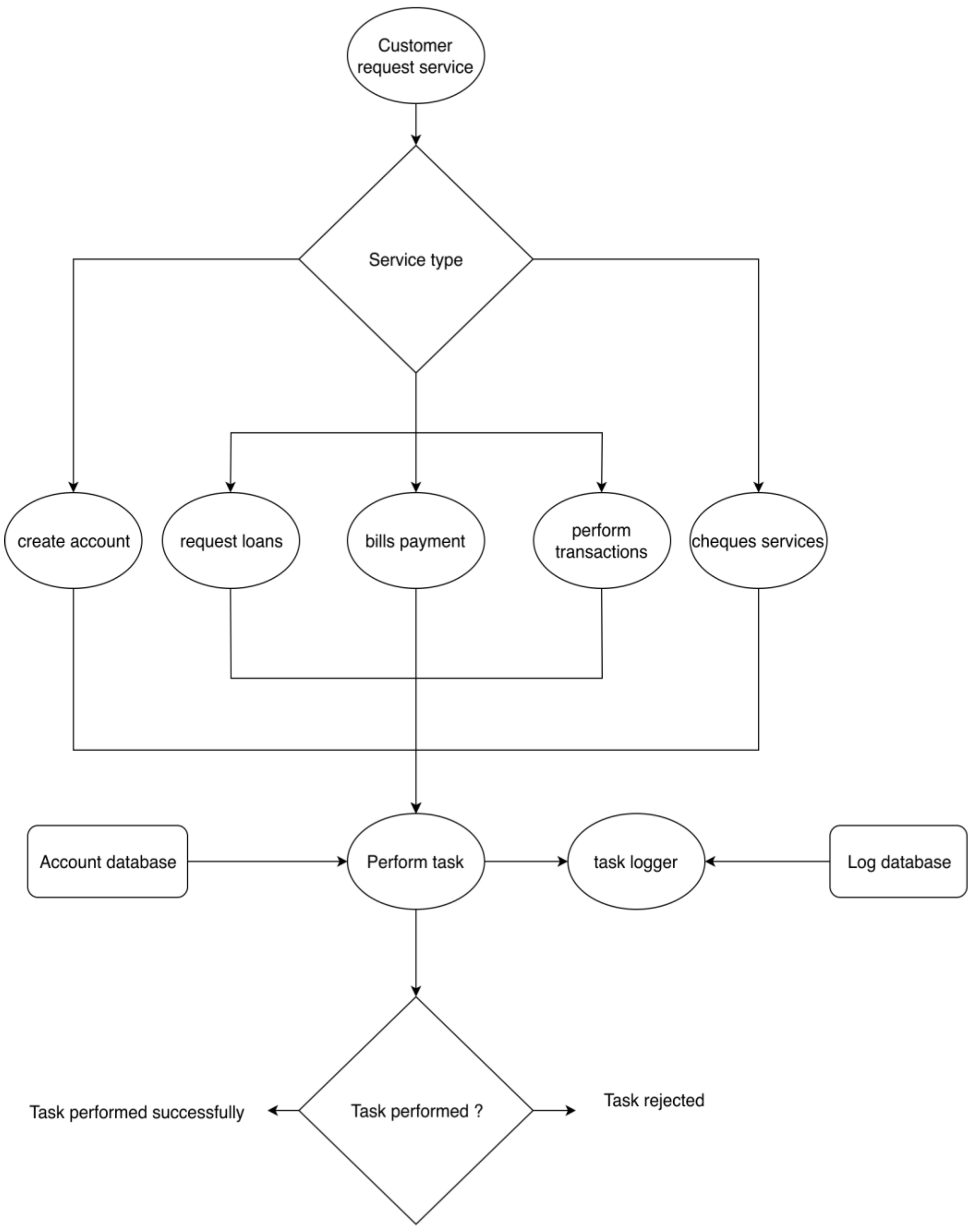


Figure 13 Process Model

All the processes performed, which is in the banking system case are all services performed by the bank, are grouped into this process model. If a process is requested, the account that requested it and the task in hand are recorded in the log database. Then it's up to the bank to either perform said task or reject it.

Deployment Viewpoint

This view illustrates a system from a programmer's perspective and is concerned with software management. It uses the UML Component diagram to describe system components.

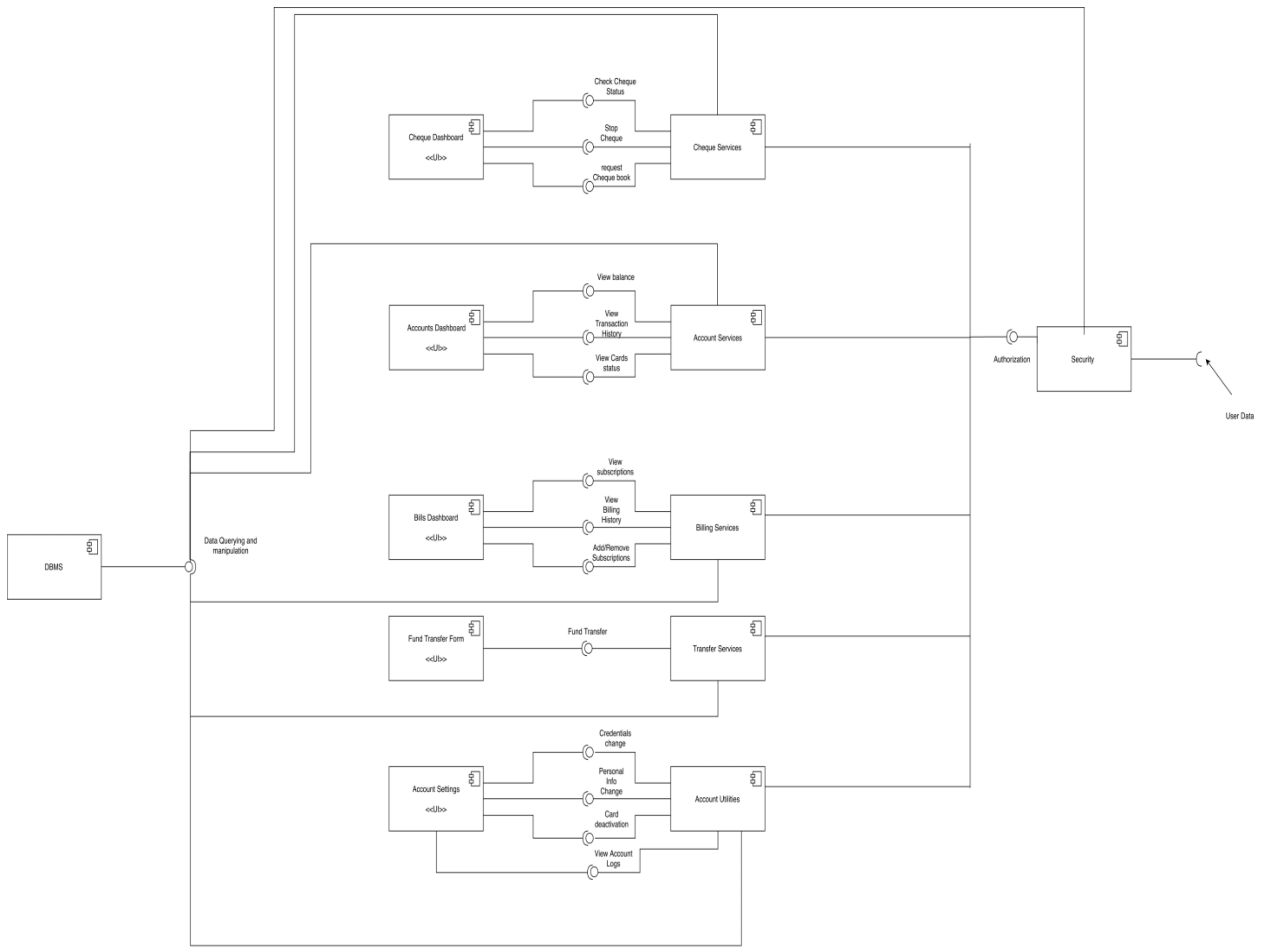


Figure 14 Component Diagram

This diagram describes the components used to produce the functionalities of the system by visualizing the physical components in a system. These components are libraries, packages and files.

Implementation Viewpoint

This viewpoint depicts the system from a system engineer's point of view. It is concerned with the topology of software components on the physical layer as well as the physical connections between these components. The Deployment diagram is used to represent just that.

The Deployment Diagram:

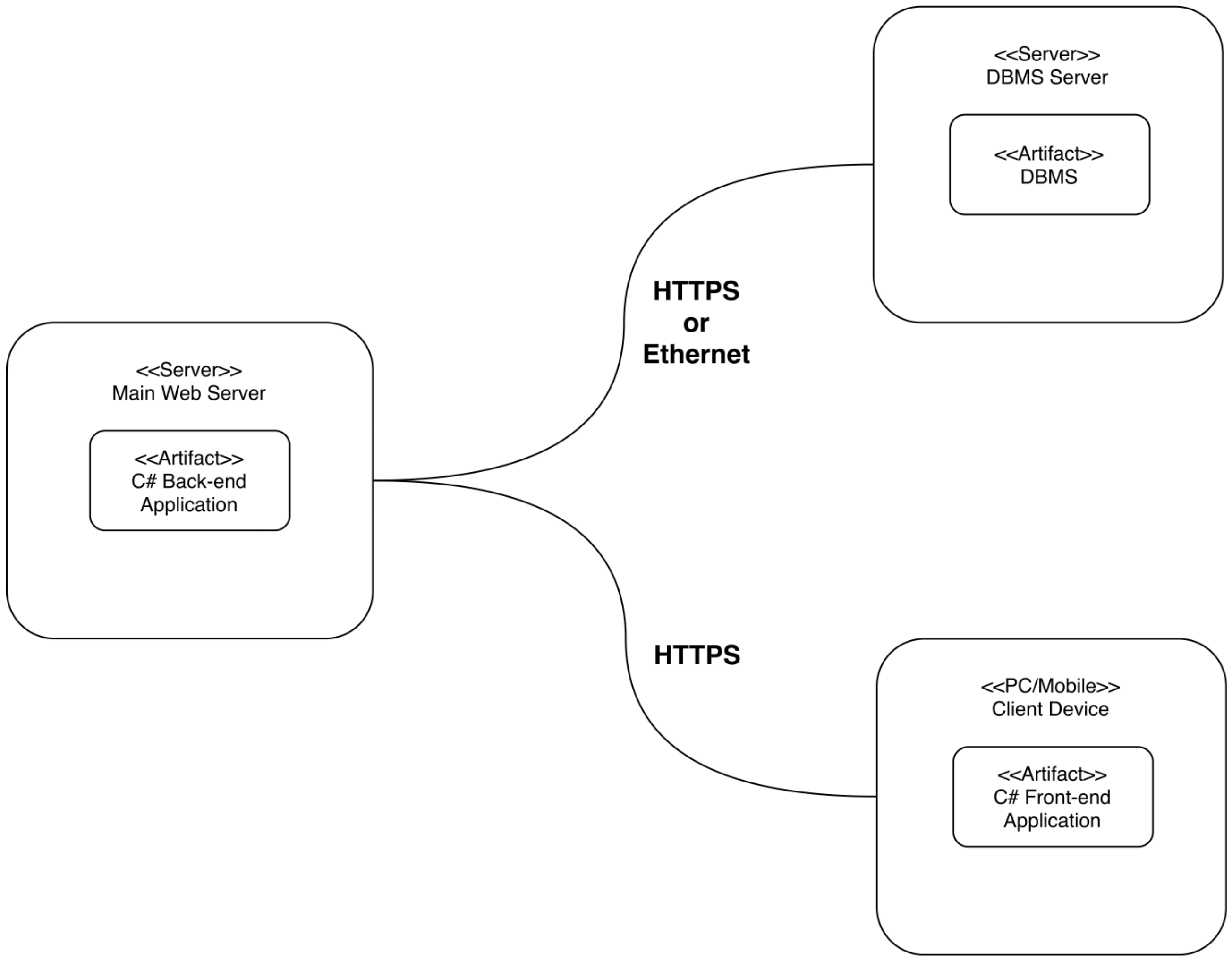


Figure 15 Deployment Diagram

Scenario Viewpoint

The description of an architecture is illustrated using a small set of use cases, or scenarios, which become the scenario viewpoint. The scenarios describe sequences of interactions between objects and between processes. They are used to identify architectural elements and to illustrate and validate the architecture design. They also serve as a starting point for tests of an architecture prototype.

Sequence Diagram:

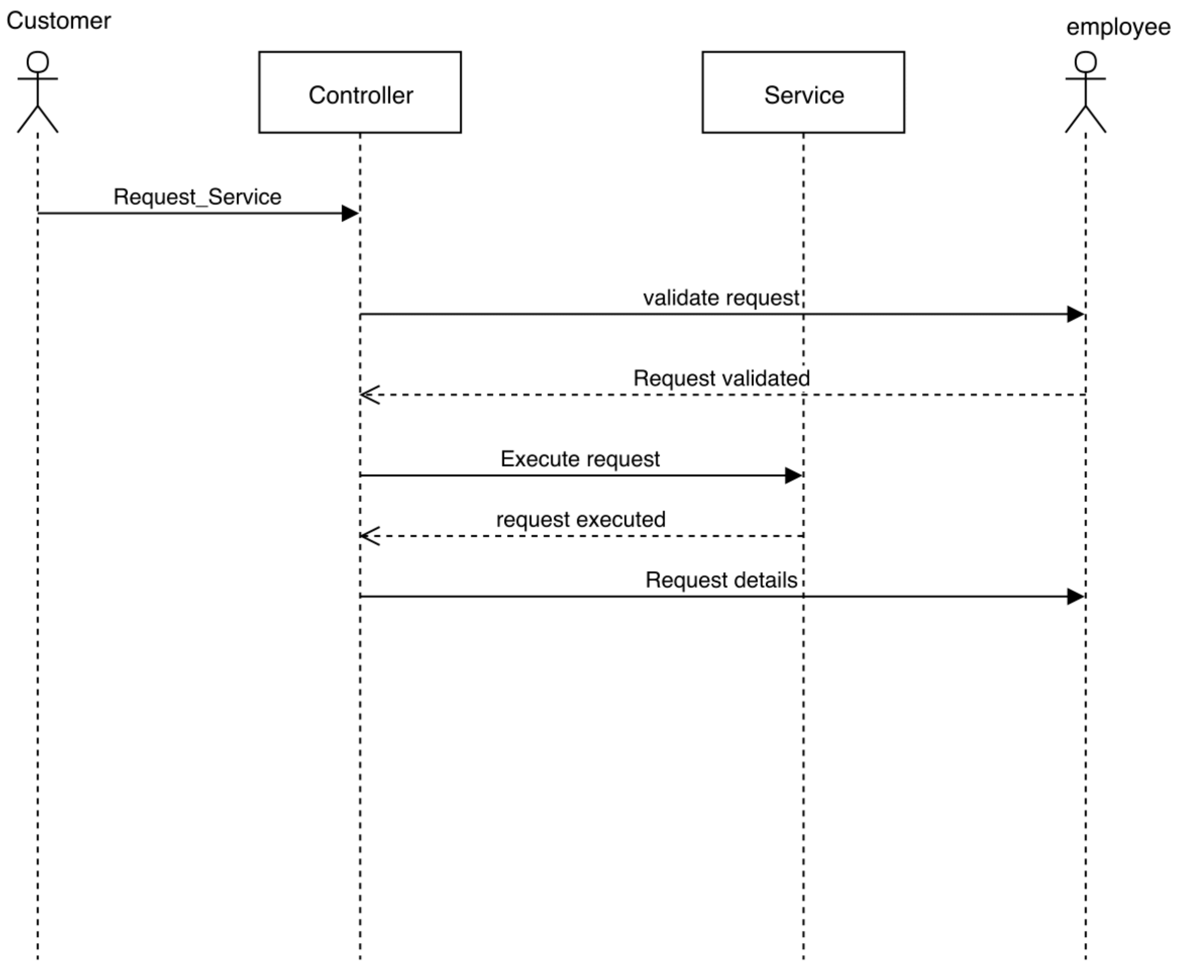


Figure 16 Sequence Diagram

This Sequence diagram shows the object interactions arranged in our desired time sequence, models the flow of control and illustrates our typical scenario, which is the customer requesting a service. It also depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. The Customer's request reaches the controller which is then advanced to the employee to validate said request which is then executed by the controller and the request details stored by the employee.

Module View

Decomposition:

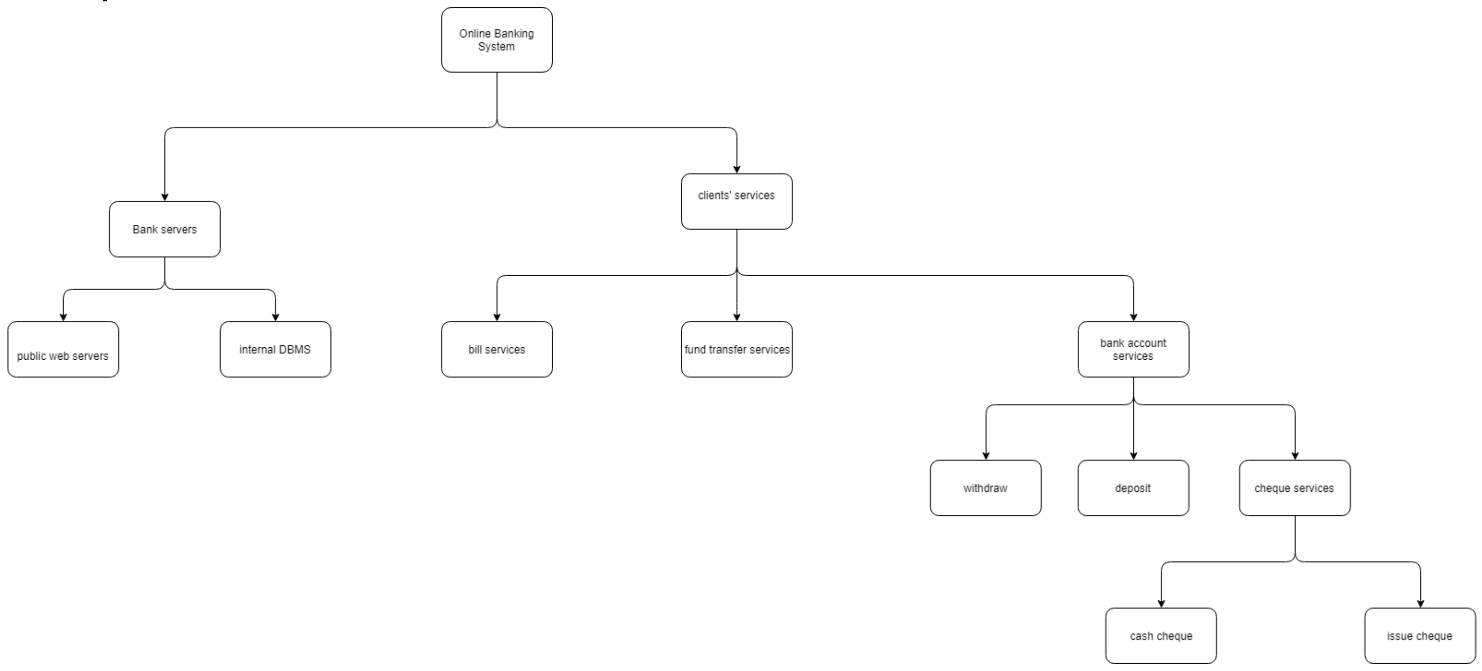


Figure 17 Module View Decomposition

- Bank branches: is a submodule of the online banking system
- public web server: is a submodule of the online banking system
- internal DBMS: is a submodule of the online banking system
- client's accounts: is a submodule of Bank branches
- bills services: is a sub module of client's services
- funds transfers services: is a sub module of client's services
- bank account services: is a sub module of client's services
- withdraw: is a sub module of bank account's services
- deposit: is a sub module of bank account's services
- cheque services: is a sub module of bank account's services
- cash cheque: is a sub module of cheque services
- issue cheque: is a sub module of cheque services

Uses:

- clients' devices use public web servers to communicate with DBMS
- client's use GUI to request account's services

Layers:

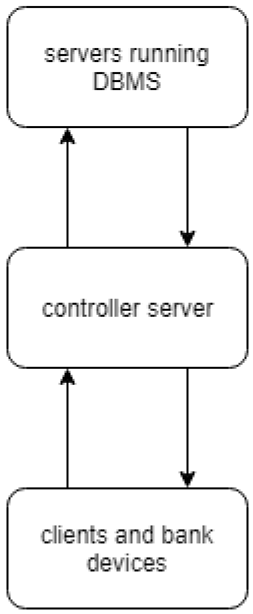


Figure 18 Module View Layers

only bank and client’s devices can use the main servers to complete requests.

- controller server use DBMS to store customer's requests’ data
- controller server can send messages to banks and client’s devices
- DBMS can send information to the controller server
- both bank and client’s devices cannot communicate directly with the DBMS

Classes:

Classes used in this project are presented in this class diagram

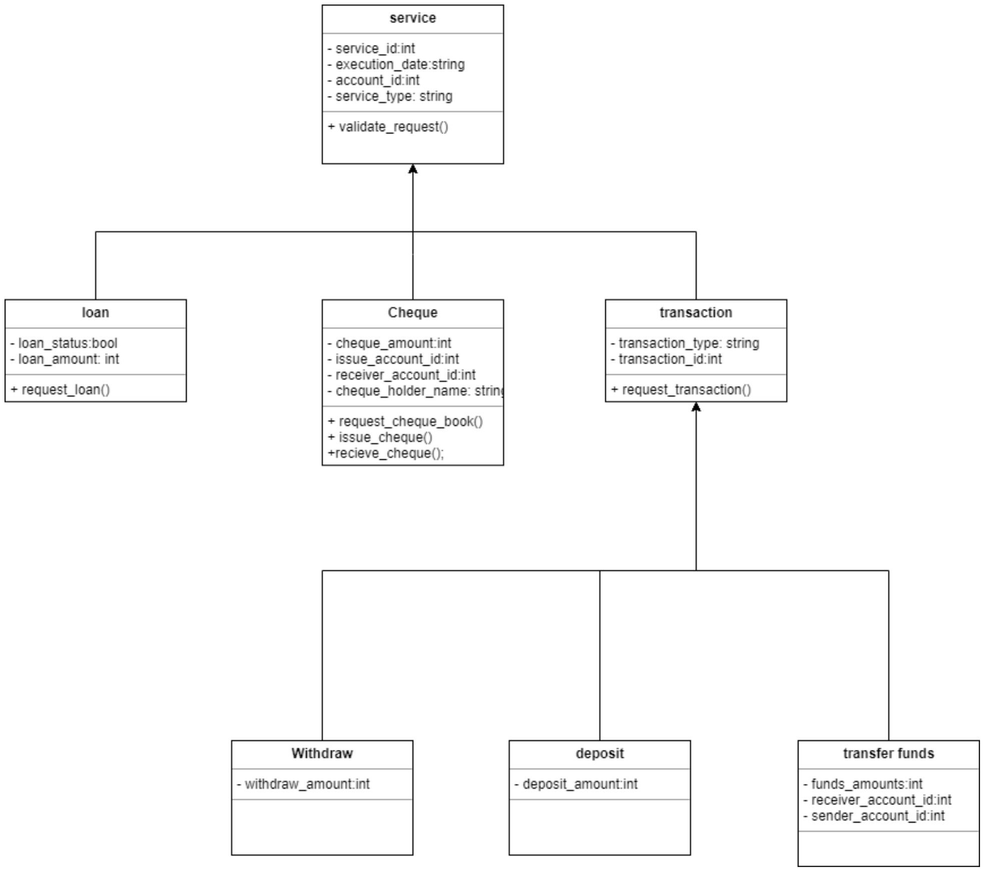


Figure 19 Module View Classes

ANALYZING THE FUNCTIONAL POINTS

Table 1:

Number	*Complexity weighting factor*	Value= 0	Value= 1	Value= 2	Value= 3	Value= 4	Value= 5	F_i
1	Backup and recovery	0	0	0	0	0	1	5
2	Data communications	0	0	0	1	0	0	3
3	Distributed processing	0	0	1	0	0	0	2
4	Performance critical	0	0	0	0	1	0	4
5	Existing operating environment	0	1	0	0	0	0	1
6	On-line data entry	0	0	0	0	1	0	4
7	Input transaction over multiple screens	0	0	0	1	0	0	3
8	Master files updated online	0	0	1	0	0	0	2
9	Information domain values complex	0	0	0	1	0	0	3
10	Internal processing complex	0	0	0	1	0	0	3
11	Code designed for reuse	0	0	0	0	1	0	4
12	Conversion/installatio n in design	0	0	0	0	1	0	4
13	Multiple installations	0	0	1	0	0	0	2
14	Application designed for change	0	1	0	0	0	0	1
	Total							41

Figure 20 FPA Table 1

Table 2:

Info Domain	Optimistic	Likely	Pessimistic	Est Count	Weight	FP count
Number of Inputs	2	4	6	4	4	16
Number of Outputs	8	11	14	11	5	55
Number of Inquiries	2	4	6	4	4	16
Number of Files	1	2	3	2	10	20
Number of External Interfaces	1	2	3	2	10	20
	UFC: Unadjusted Function Count					127
		Complexity Adjustment Factor				1.06
					FP	135

Figure 21 FPA Table 2

COST ESTIMATION

- C#=59 LOC/FP
- LOC=135*59=7965

1)LOC approach:

Assuming

Estimated project LOC = 7965 LOC

350 LOC/p-m

Burdened labor rate = 8000 $/p-m

Then

Effort = 7965/350 = 23 p-m

Cost per LOC = 8000/350 = (12.9) = 23 $/LOC

Project total Cost = 8000 * 23 = 18400 $

2)FP approach:

Assuming

Estimated FP = 135

5.5 FP/p-m

Burdened labor rate = 8000 $/p-m

Then

Estimated effort = 135/5.5 = 25 p-m

Cost per FP = 8000/5.5 = 1455 $/FP

Project cost = 8000 * 25 = 20,000 $

Two methods of estimations the were used:

Expert judgment:

We consulte two experts in software development field to provide an estimate for our project's cost to compare it to the previously calculated costs, which were based on our assumptions.

THE VERDICT: The experts' estimates were close to our calculations.

Estimation by analogy:

We searched for previous similar projects for online banking system of significant quality, then compared these projects' actual costs with our estimation costs.

THE VERDICT: Almost the same calculations and costs as the other banking systems projects.

User Guide

LOG IN:

- The first screen all users will see is the login screen.
- Customers can login to their accounts using this screen

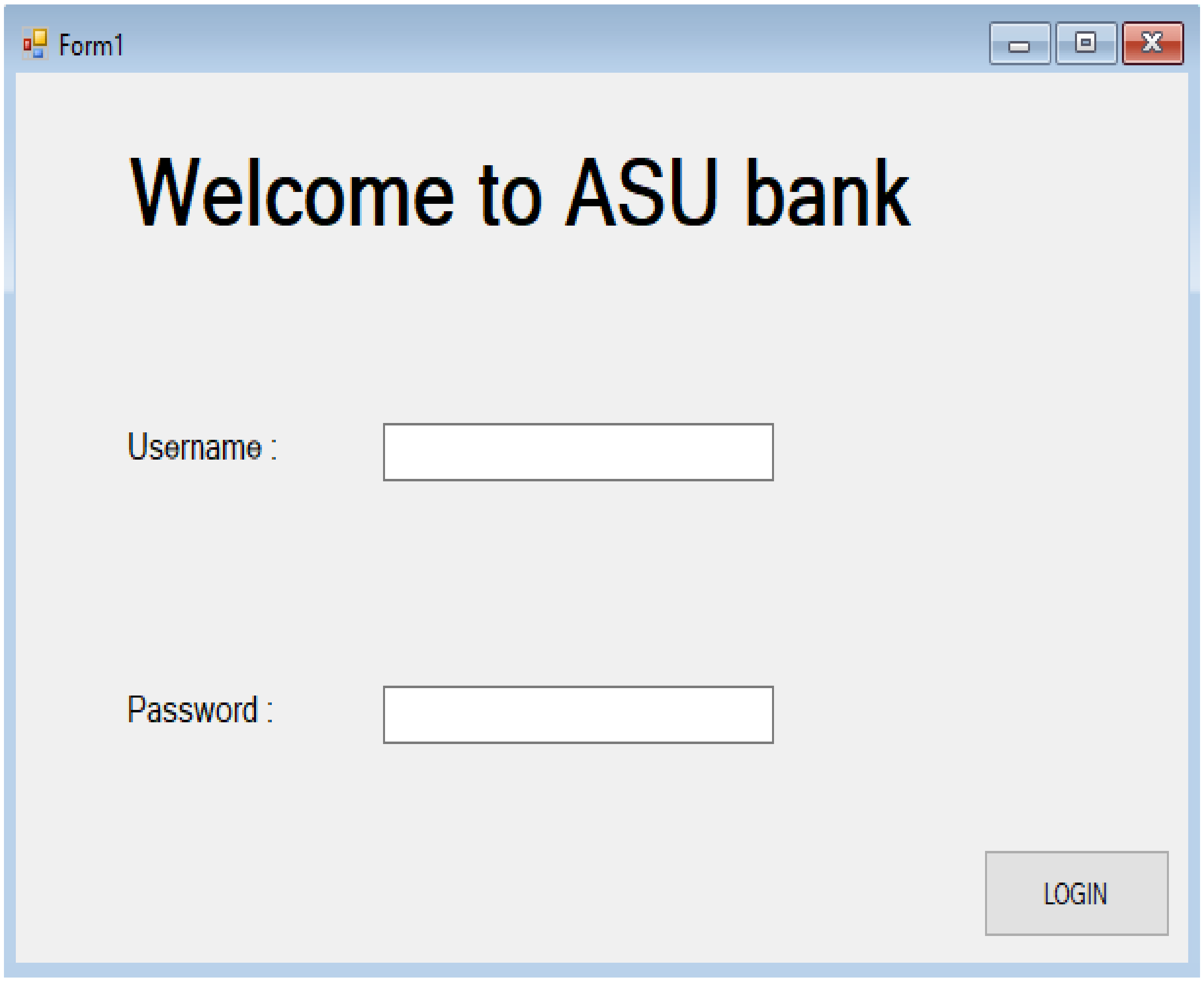


Figure 22 Log In screen

SERVICES

- After Login, all customers go to the service page.
- In this page customers can go to cheque services, view account, transfer funds, bills, utility form or exit the program.

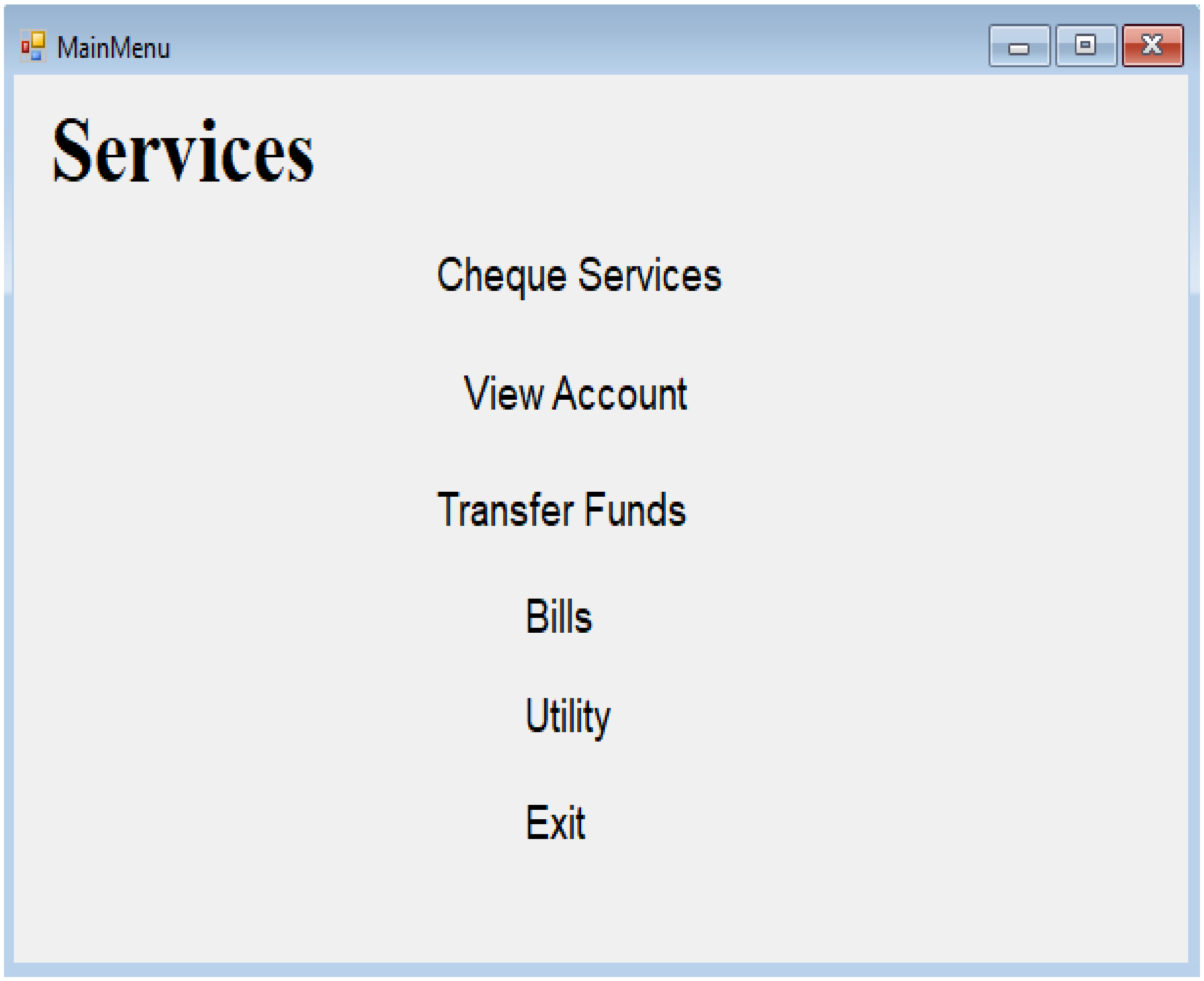
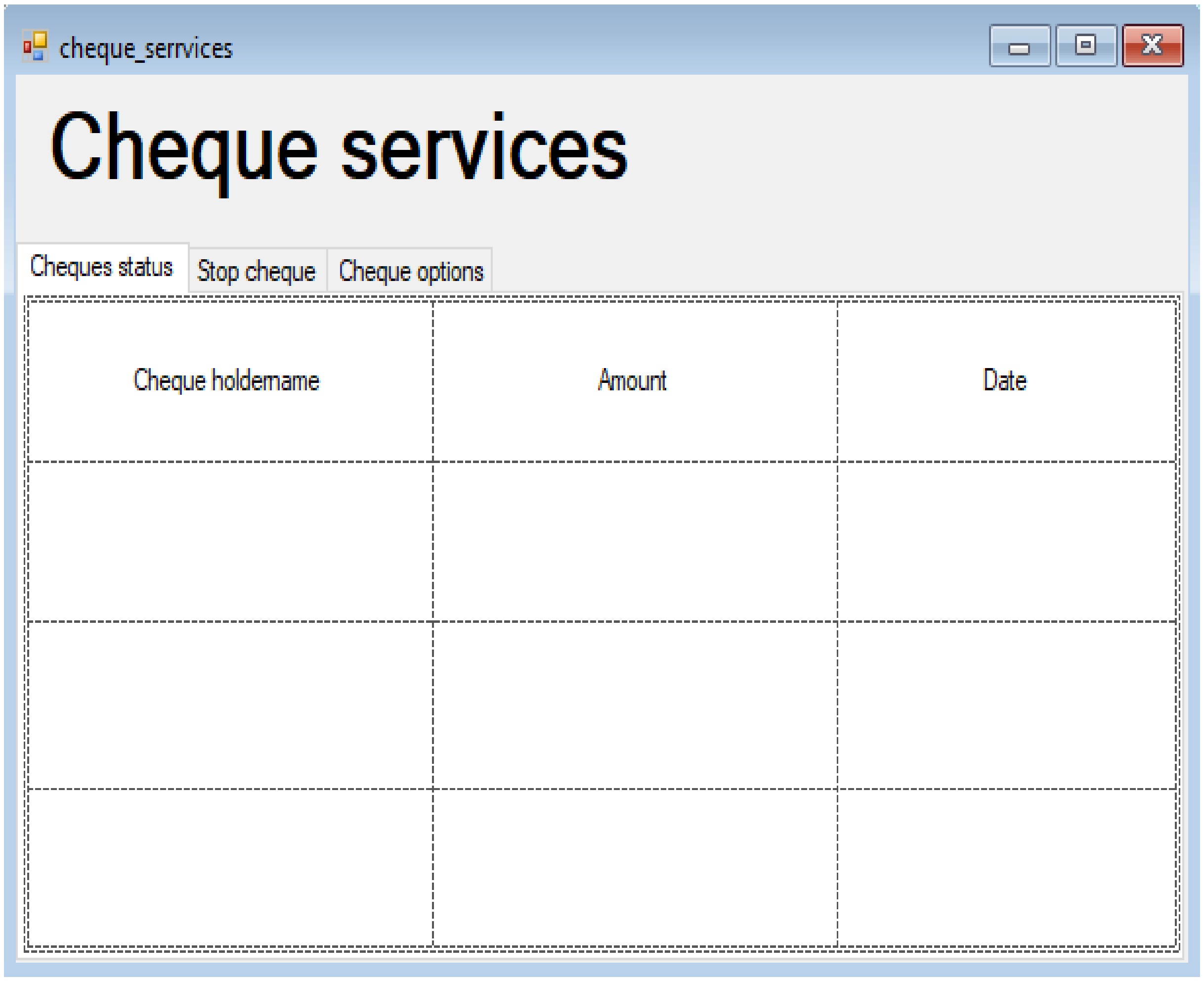


Figure 23 Services Screen

CHEQUE SERVICES

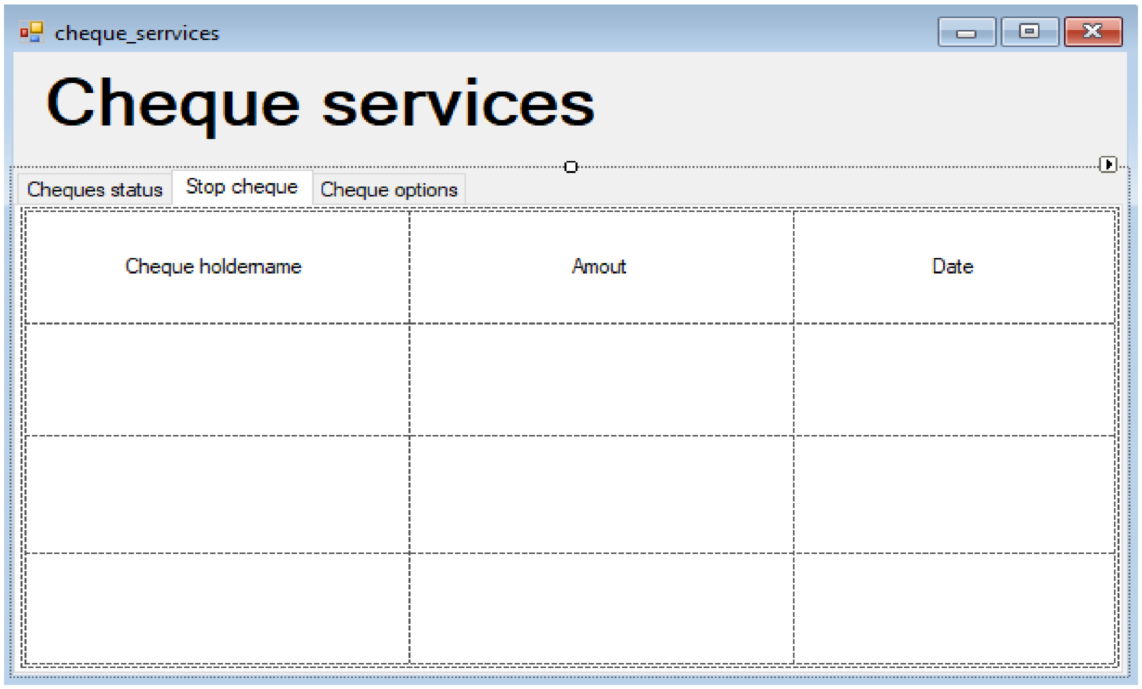
- In the Cheque Services window, the customer can control everything concerning his cheques
- Starting with the Cheque status, User can view the status of all his issued and waiting clearance cheques.



Cheque holdemame	Amount	Date

Figure 24 Cheque services screen

In This tab, user can view his issued cheques and can choose to stop them if he pleases.



Cheque holdername	Amout	Date

Figure 25 Cheques Viewer Screen

In this tab, the user can request a new cheque book for the bank to receive and collect it afterwards.

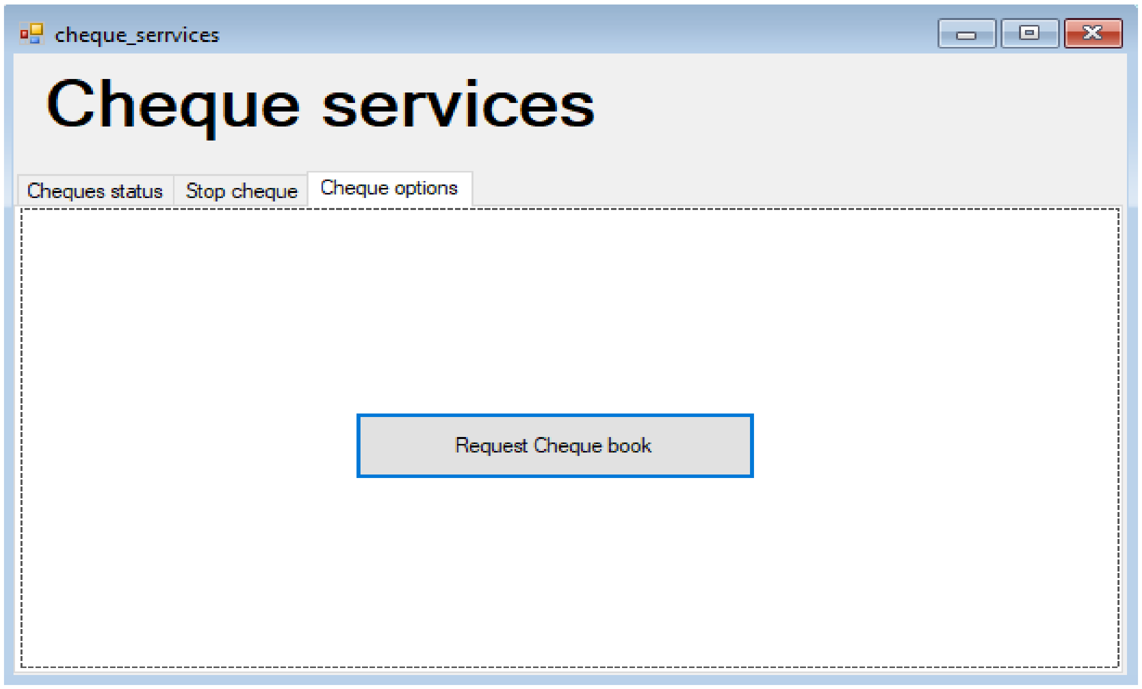


Figure 26 Request Book Screen

TRANSFER FUNDS

In the transfer funds form user can transfer funds from his account to any other account with the currency he wants.

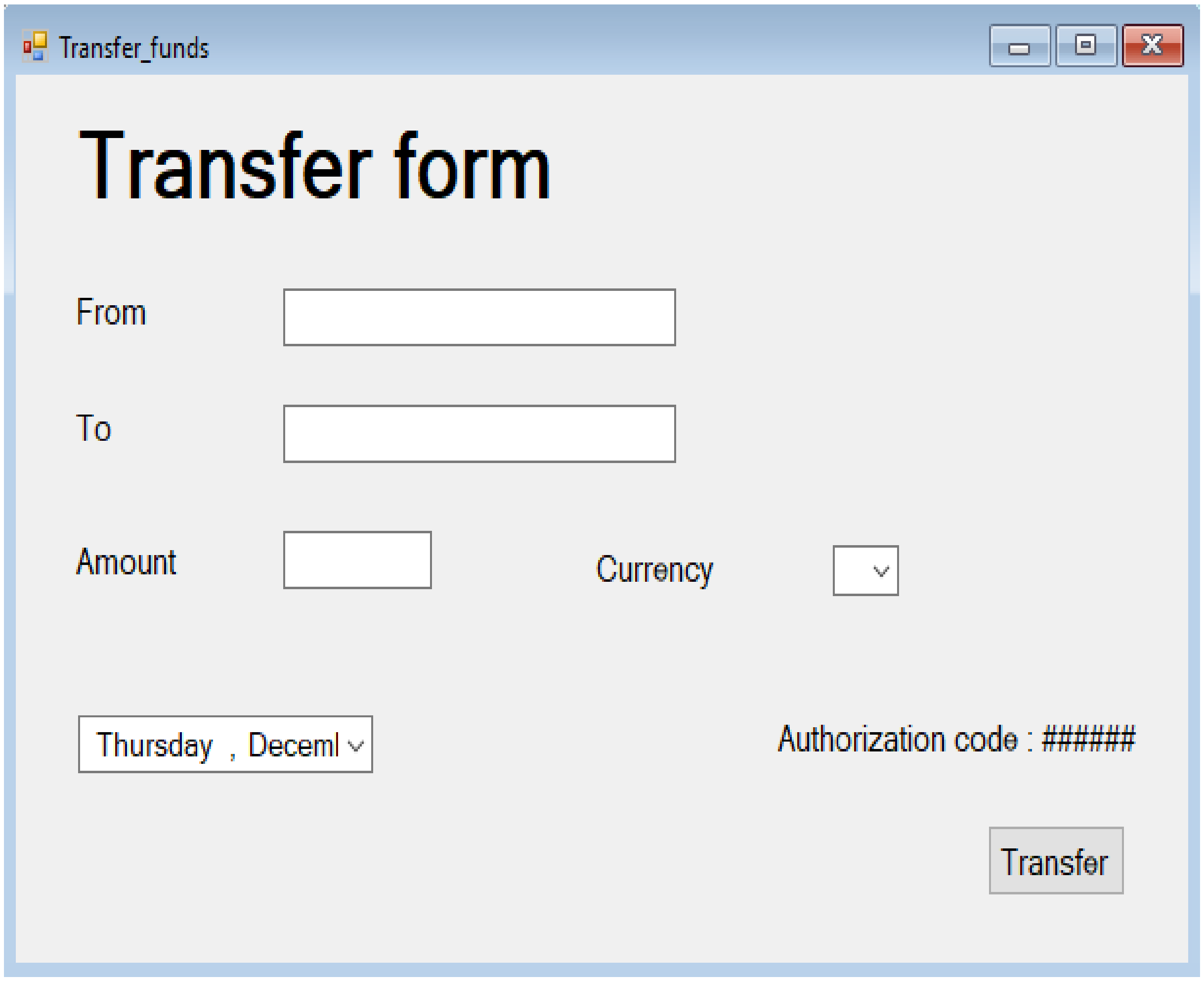
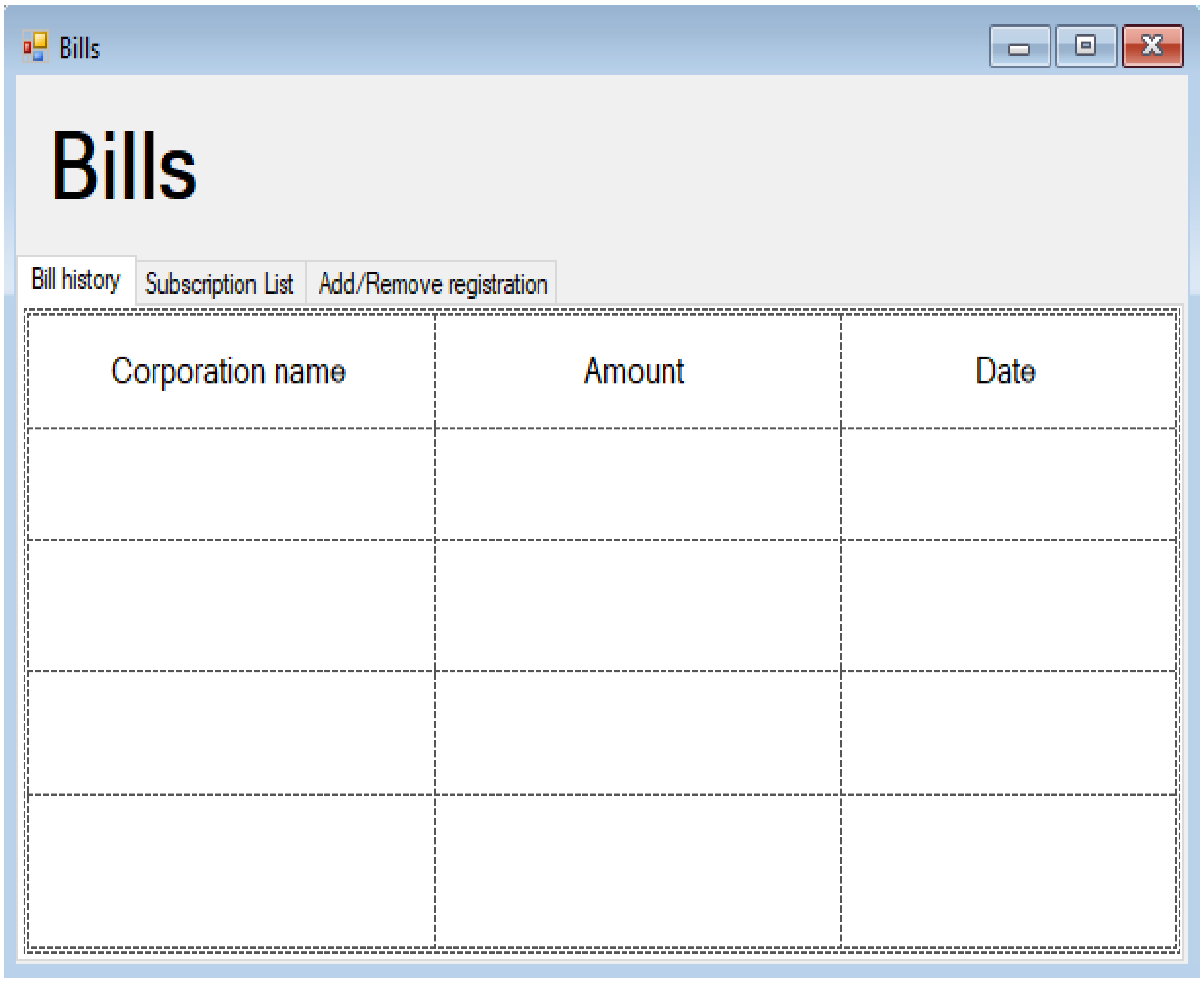


Figure 27 Transfer Funds Screen

BILLS

In the Bill Form customer can control everything about his Bills.

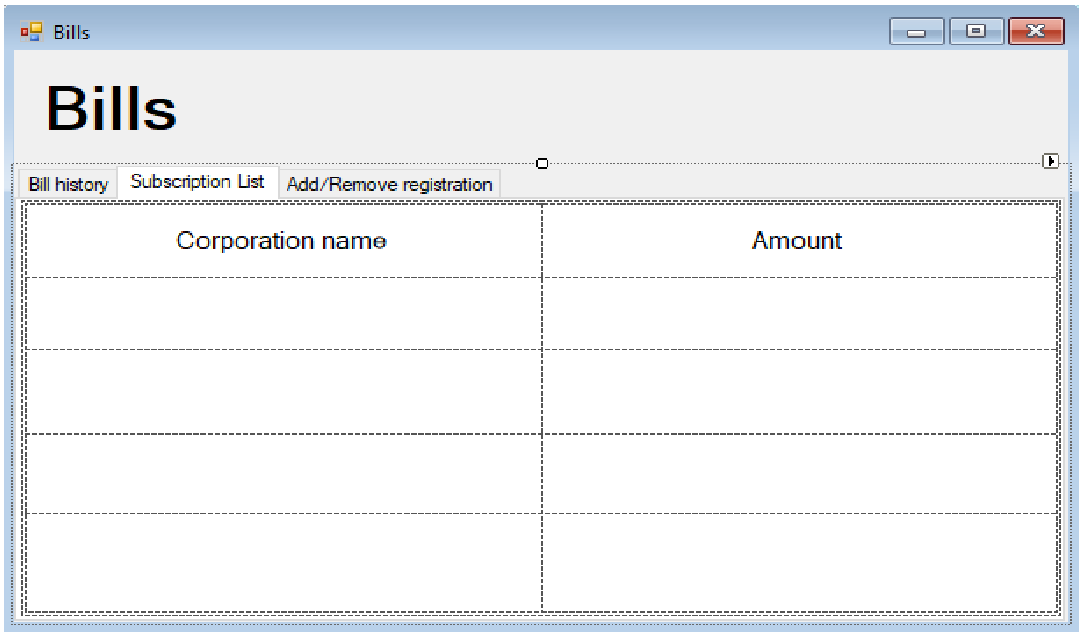
This first tab is the bill history, which shows the bills that have been paid by this account.



Corporation name	Amount	Date

Figure 28 Bills Main Screen

The second tab is the Subscription list, which shows the bills that are currently associated with the account for payment.



Corporation name	Amount

Figure 29 Subscriptions Screen

The last tab gives the user the ability to add or remove registration of bills (subscriptions).

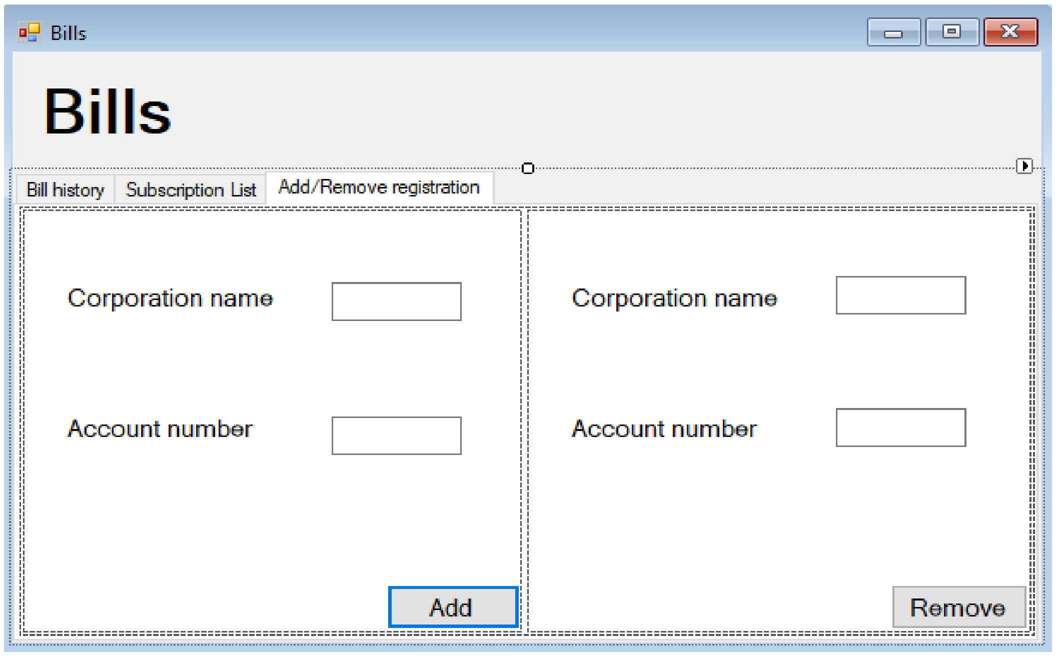


Figure 30 Subscription Adjustments Screen

VIEW ACCOUNT

In this form user can see his current account and his saving account information: balance, transaction history and his associated cards to each account.

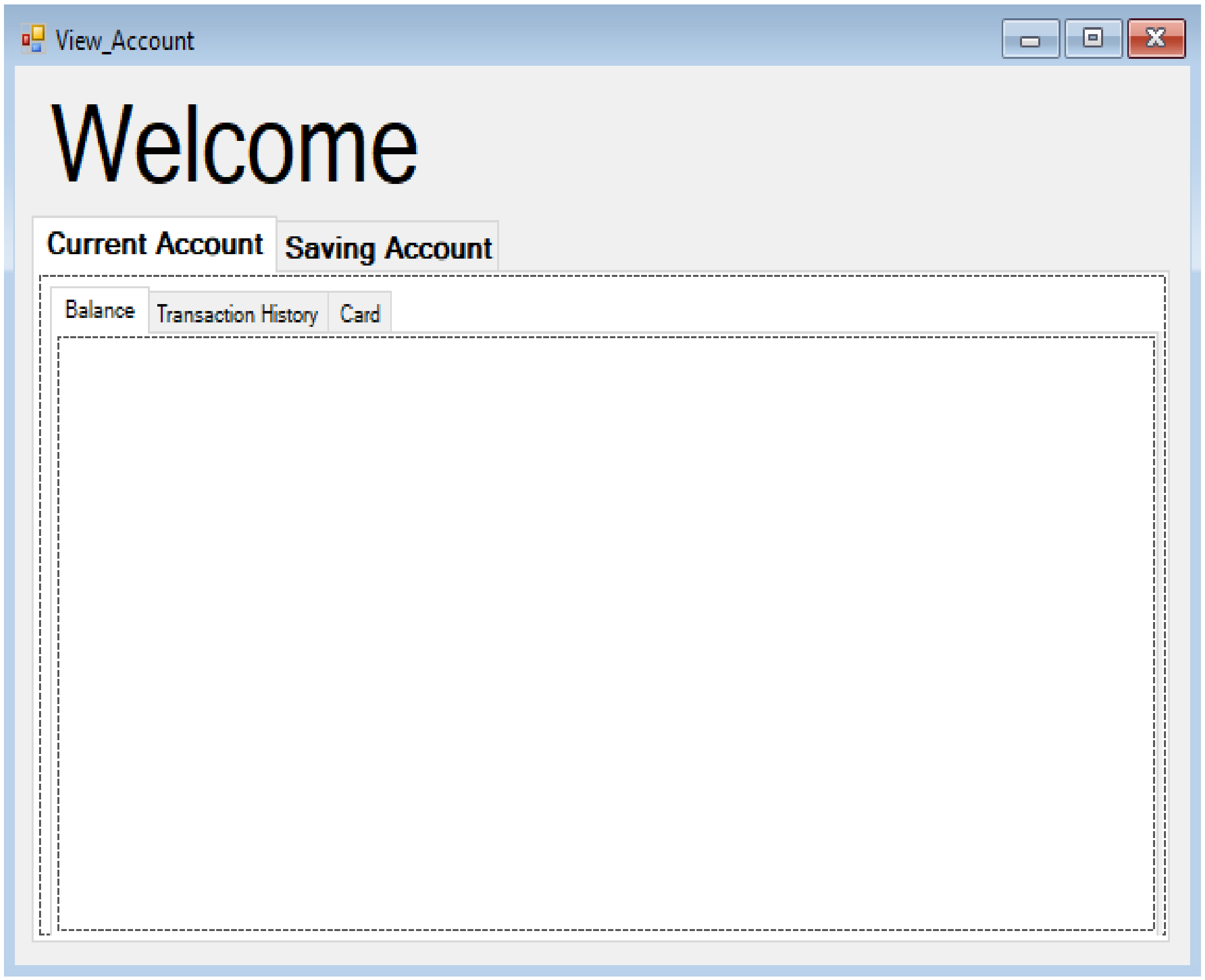


Figure 31 View Account Screen

UTILITY

In the utility form, user can change a lot of settings regarding his/her account.

The user can change his password, personal information, view logs to the account and cancel ATM facilities (the use of ATM).

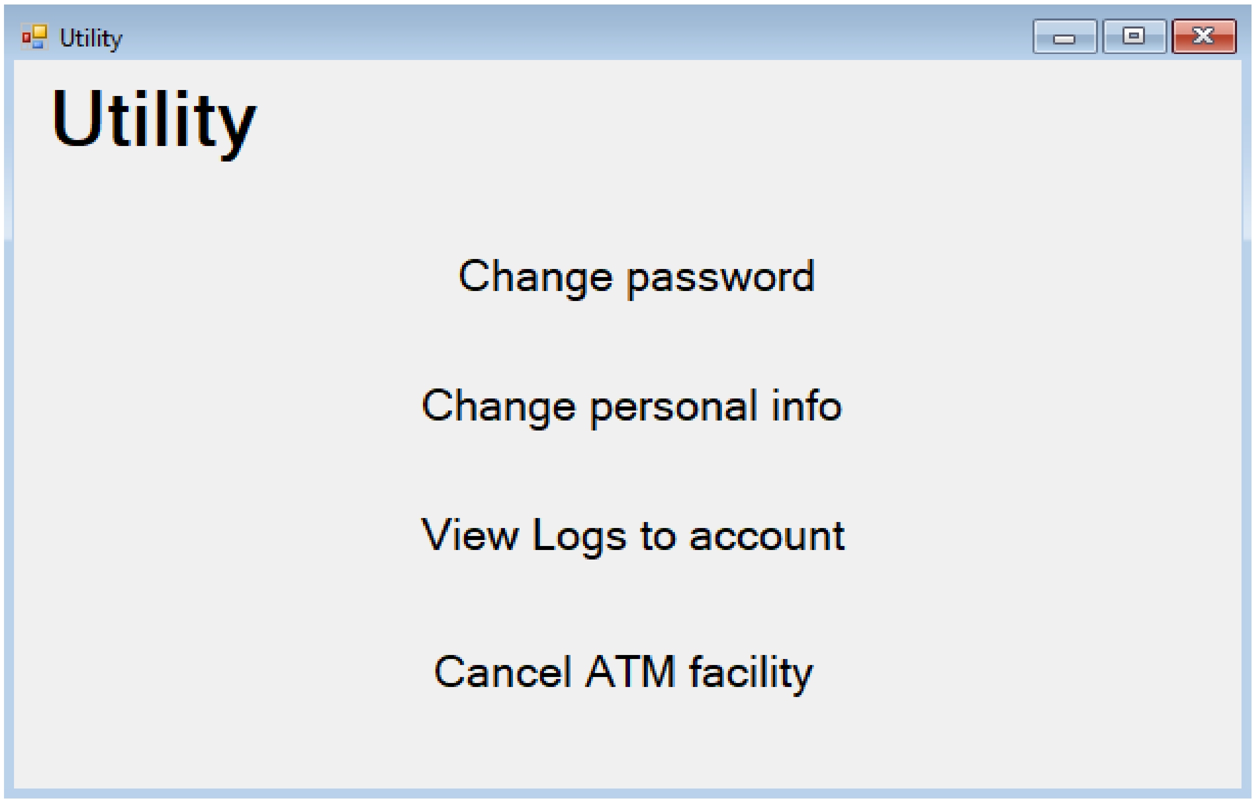


Figure 32 Utilities Screen

REFRENCES

1. Abran, Alain; Moore, James W.; Bourque, Pierre; Dupuis, Robert; Tripp, Leonard L. (2004). Guide to the Software Engineering Body of Knowledge. IEEE. ISBN 0-7695-2330-7.
2. Sommerville, Ian (2008). Software Engineering (7 ed.). Pearson Education. ISBN 978-81-7758-530-8. Retrieved 10 January 2013.
3. Pressman, Roger S (2009). Software Engineering: A Practitioner's Approach (7th ed.). Boston, Mass: McGraw-Hill. ISBN 978-0073375977.
4. Sommerville, Ian (2010) [2010]. Software Engineering (9th ed.). Harlow, England: Pearson Education. ISBN 978-0137035151.
5. Jalote, Pankaj (2005) [1991]. An Integrated Approach to Software Engineering (3rd ed.). Springer. ISBN 0-387-20881-X.
6. Bruegge, Bernd; Dutoit, Allen (2009). Object-oriented software engineering: using UML, patterns, and Java (3rd ed.). Prentice Hall. ISBN 978-0136061250.
7. Thibodaux, Patrick (2006-05-05). "As outsourcing gathers steam, computer science interest wanes". Computerworld.com. Retrieved 2016-12-06
8. "Computer Programmers". Bls.gov. Retrieved 2012-03-25.
9. Mullins, Robert (2007-03-13). "Software developer growth slows in North America". InfoWorld. Archived from the original on 2009-04-04. Retrieved 2012-03-25.
10. "Gartner Magic Quadrant" (PDF). Cognizant.com. Retrieved 2012-03-25.
11. Casey, Valentine (2010-08-20). "Virtual software team project management". Springer. Retrieved 2013-12-06.
12. https://www.dagstuhl.de/Reports/96/9635.pdf
13. http://www.stevemcconnell.com/psd/04-senotcs.html
14. "'SWEBOK Guide Version 3'". Retrieved 2015-03-09.
15. ""Software Engineering Code of Ethics"" (PDF). Retrieved 2012-03-25.
16. https://www.bls.gov/ooh/computer-and-information-technology/software-developers.htm
17. https://www.bls.gov/ooh/computer-and-information-technology/home.htm
18. https://www.bls.gov/ooh/computer-and-information-technology/computer-programmers.htm
19. "Software Engineering". Retrieved 2008-02-01.
20. "Computer Software Engineers and Computer Programmers". Retrieved 2009-12-17.
21. "SEI certification page". Sei.cmu.edu. Retrieved 2012-03-25.
22. Wyrostek, Warren (March 14, 2008). "The Top 10 Problems with IT Certification in 2008". InformIT. Retrieved 2009-03-03.
23. IEEE Computer Society. "2006 IEEE computer society report to the IFIP General Assembly" (PDF). Retrieved 2007-04-10.
24. IEEE. "CSDA". Retrieved 2010-04-20.
25. https://engiegirlsatuwaterloo.wordpress.com/2013/08/29/computer-engineering-software-engineering-or-computer-science/