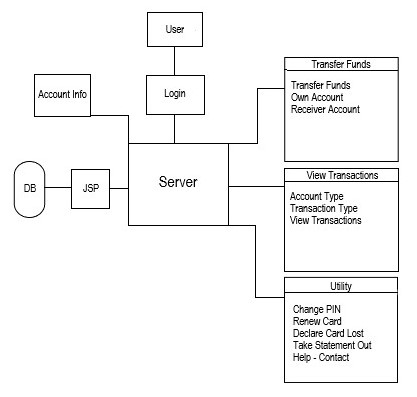
**3. SYSTEM ARCHITECTURE**

**3.1 Architectural Design**

Architectural design involves major system components and the communication between them. The architectural design looks as follows:

****

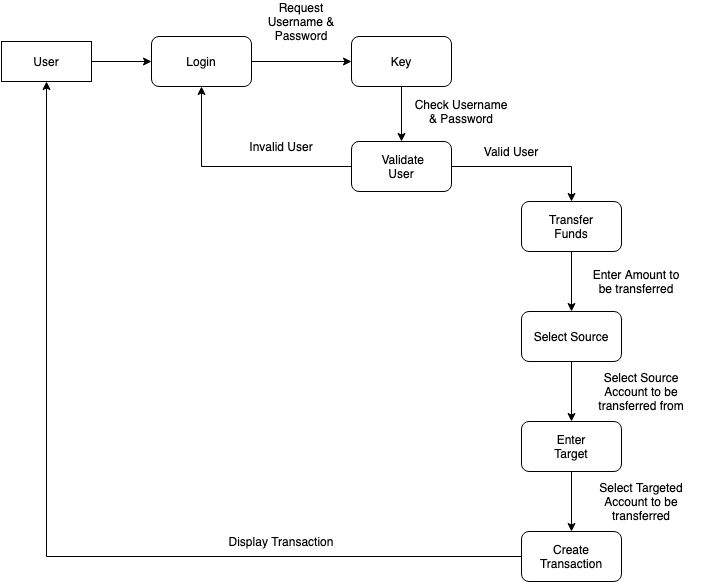
All the subsystems collaborate with each other through a server, which communicates through ASP, which is a server-side language used in web development to implement dynamic web pages, with the database. All the data that the user asks for and all the data that the user updates or modifies are stored in a database.

**3.2 Decomposition Description**

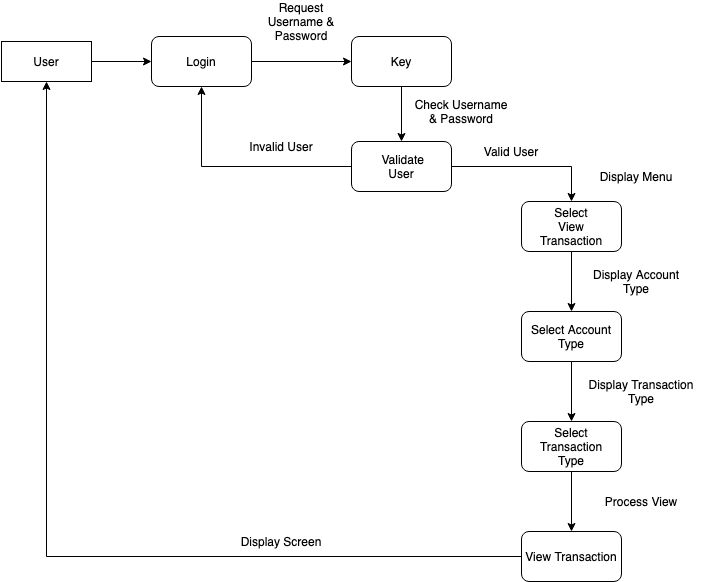
In our online banking architecture, every subsystem needs firstly to go through a login process (*Login -> Key)*. The user then goes through another process, which is the validation process. Here, the user is tested if it is a *Valid User* or an *Invalid User.*

The data flow of the present subsystems is as follows:

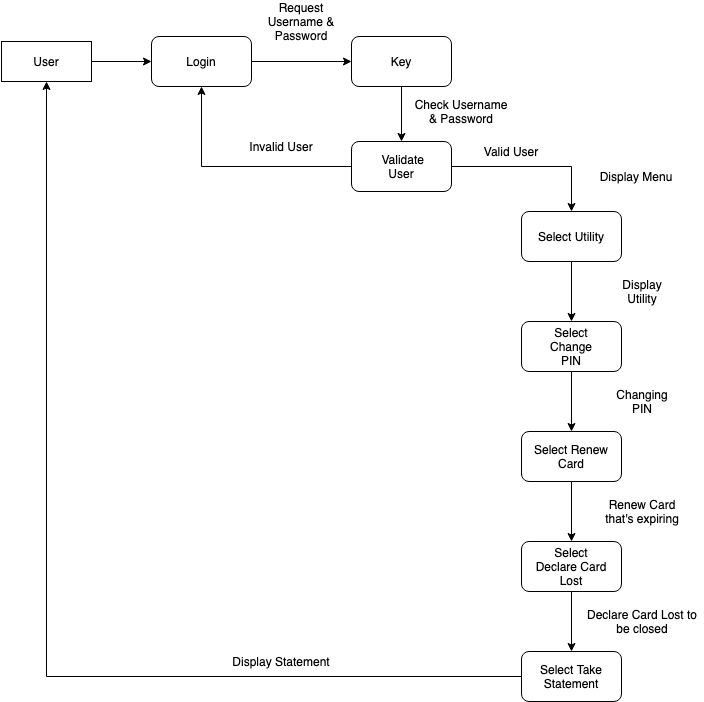
1. ***Data Flow Diagram – Transfer Funds***

******

1. ***Data Flow Diagram – View Transactions***

******

1. ***Data Flow Diagram – Utility***

******

**3.3 Design Rationale**

The design rationale is easy to understand. The user has three main sections that are at his disposal. One is a section dedicated only to the transfer of funds. Here, the user has the option to transfer money from his own account to other users’ account or pay bills, or he can transfer money between two accounts of his own, and can also receive money from another user. We came to the conclusion that this is a vital option to our architecture.

A second section is the view transactions section. Here the user can view at any point in time what transactions were made, to whom the money where transfer or from whom where received, if there are any unclear transfers, he can visualize there all the details. This way a user can always keep an eye if everything is alright.

A third, and last section, is the utility section. Here the user has a list of options dedicated to utilities, like: changing the PIN – in case the user forgot the PIN, he can change it here without any problems; renewing the card – if the card is close to expiring, the user is able to request a new card; declaring the card lost – if the user lost his card, he can declare it as being lost, so the bank can close it before someone else trying to use it illegal; taking a statement out – the user can take at any time a statement if he needs one; help/contact – if the user has any unclarities, he can access this option to solve them.

Everything here is processed through a server connected to a database. The database contains all the information that a user can ask for and also saves all the data that the user changes.

**4. DATA DESIGN**

**4.1 Data Description**

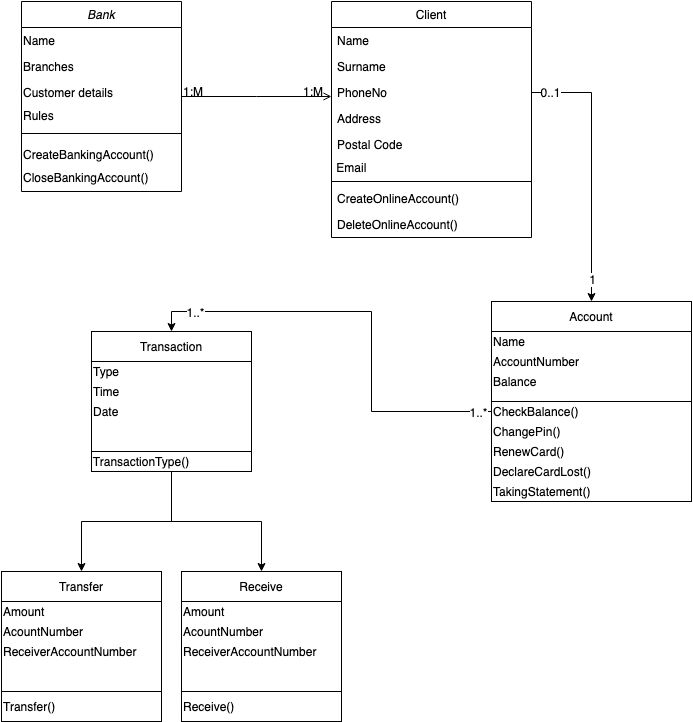
Every option that is available to a user has its own table of data, meaning that the entire data is stored in those tables, creating a data structure. When an option is used, it is added further in its respective table, memorizing what has been done, what has changed and when was changed. If a user wants to view his latest actions, he is able to view a history of those actions, last events being brought up from the table in order from bottom to top.

Let’s say if a user wants to view his last transactions from the past week, he can choose the *“View Transactions”* option for that period of time. The respective table is accessed and the transactions from that period are selected. Then, those are brought up to the user in chronological order, from latest to earliest.

**4.2 Data Dictionary**

All the functions available are:

* AccountType()
* ChangePIN()
* CheckBalanace()
* CloseBankingAccount()
* CreateTransaction()
* CreateBankingAccount()
* CreateOnlineAccount()
* DeleteOnlineAccount()
* DeclareCardLost()
* EnterTarget()
* Login()
* Receive()
* RenewCard()
* SelectSource()
* TakeStatement()
* TransactionType()
* Transfer()
* ViewTransaction()

****

Class Diagram

**5. COMPONENT DESIGN**

* **AccountType()**

Function AccountType()

‘If an account is created’

‘Choose a currency’

End Function

* **ChangePIN()**

Function ChangePIN()

‘If a PIN change is wanted’

‘Change the PIN – choose a 4 digit code’

‘Enter again the PIN to confirm’

End Function

* **CheckBalanace()**

Function CheckBalance()

‘Show the current balance’

End Function

* **CloseBankingAccount()**

Function CloseBankingAccount()

‘If closing account is wanted’

‘If sure’

‘Close account’

End Function

* **CreateTransaction()**

Function CreateTransaction()

‘If transaction is requested’

‘Choose transaction type’

‘Execute transaction’

End Function

* **CreateBankingAccount()**

Function CreateBankingAccout()

‘If creating a banking account is requested’

‘Choose account type’

‘Choose PIN’

‘Confirm’

End Function

* **CreateOnlineAccount()**

Function CreateOnlineAccout()

‘If creating an online account is requested’

‘Enter the requested data’

‘Confirm’

End Function

* **DeleteOnlineAccount()**

Function DeleteOnlineAccount()

‘If the deletion of online account is requested’

‘If sure’

‘Delete’

End Function

* **DeclareCardLost()**

Function DeclareCardLost()

‘If card is lost’

‘Close card’

End Function

* **Login()**

Function Login()

‘Enter username’

‘Enter Password’

‘Confirm’

‘If user is valid’

‘Continue’

End Function

* **Receive()**

Function Receive()

‘Receive sum’

‘Update balance’

‘View transaction’

End Function

* **RenewCard()**

Function RenewCard()

‘If old card is still valid’

‘Close old card’

‘Make a request for a new card’

End Function

* **SelectSource()**

Function SelectSource()

‘Enter the data of the desired source’

End Function

* **TakeStatement()**

Function TakeStatement()

‘If a statement is requested’

‘Ask what the statement is for’

‘Release statement’

End Function

* **TransactionType()**

Function TransactionType()

‘Choose the type of transaction’

End Function

* **Transfer()**

Function Transfer()

‘Choose amount’

‘Select sender account’

‘Enter receiver account number’

End Function

* **ViewTransaction()**

Function ViewTransaction()

‘Select date’

‘Select transaction’

‘View transaction’

End Function