ONLINE BANKING SYSTEM

Authors:

Name	**Email**
Ananta Arora	arora.ana@husky.neu.edu
Manthan Thakker	thakker.m@husky.neu.edu
Siddhant Pasari	pasari.s@husky.neu.edu

Course Name: Database Management Systems

Semester: 2nd Semester

Instructor Name: Prof. Kenneth Baclawski

ABSTRACT:

The purpose of this project, is to gain in-depth understanding about relational databases and gain hands on experience in MySQL. The project is like an online banking system that deals with multiple accounts pertaining to several registered customers. The customer can perform multiple transactions like withdraw, deposit, transfer and check balance. Customer can also request for debit / credit cards, request for various types of loans and pay loan amounts. This project simulates the working of internet banking, trying to enhance a user's banking experience.

INTRODUCTION:

Purpose Statement:

- The purpose of this project, online banking system is to gain in-depth understanding about relational databases and gain hands on experience in MySQL.

Objectives

The objectives of the project are to implement the following features such as

- **User Management:** Managing the basic information of all the people related to the bank (e.g. Customers, Employees).
- **Account information:** Managing different types of accounts such as checking and savings account for efficient transaction processing of all the customers.
- **Customer Transaction**: Processing transactions such as withdrawal, deposits and money transfers and simultaneously updating the database thereby maintaining the **ACID** properties.
- **Loans:** Providing several types of loans such as home loan, car loan and education loan.
- **Cards:** Providing credit and debit cards to the customers.

REQUIREMENTS

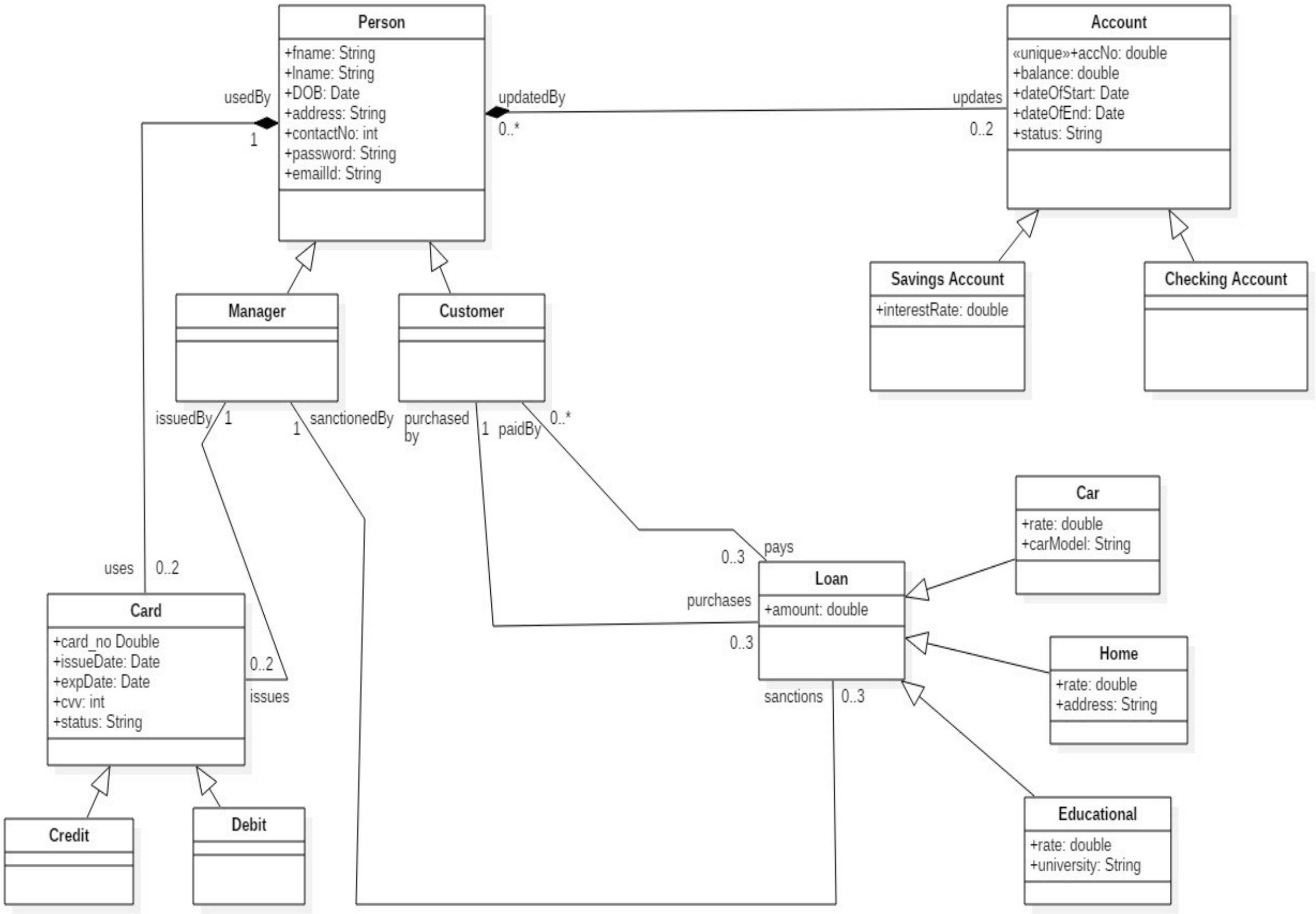
The Use Case Description Ontology is designed for formally specifying requirements. The file below is a link to the use cases of our project.

use_case_final.pdf

DESIGN

The aim of the project was to build an online banking system so that a user can bank on the go, thereby enhancing a user's banking experience. Our goal was to provide the user with all the basic banking facilities such as deposit, withdraw, transfer, balance check etc. We began the project by planning and designing the requirements and the use-cases. The implementation covers the requirements depicted in the use cases. We developed 14 tables and designed java codes to interact with the SQL database and to perform various operations. The project was developed using Java 8 and MySQL.

Class Diagram:



IMPLEMENTATION

Programming language: Java 8

IDE: Netbeans 8.1

Database: MySQL 5.7.1

JAVA ARTIFACTS

The Java classes created to implement the Online Banking System are listed below:

- **Person.java** – This class provides all the basic user details such as first name, last name, contact number, email id, address etc. A person is a customer and/or manager.
- **Customer.java** – This class inherits features from the person class.
- **Manager.java** – This class inherits features from the person class.
- **Account.java** – This class provides various details such as account number, dateOfStart, dateOfEnd, and status
- **SavingsAccount.java** – This class is same as Account except there is an additional field Rate of Interest which is fixed by the bank and dispatched yearly.
- **CheckingAccount.java** – This class is same as Account except there is no rate of interest like savings account.
- **Card.java -** This class provides all the card details of the user. The details include the card number, cvv and expiration date, issue date.
- **Debit.java –** This class inherits features from the card class. Whenever Debit Card is used balance is deducted from the savings account.
- **Credit.java -** This class inherits features from the card class. Max Credit of 1.5 * Balance is provided to the customer. The Customer can use the credit and pay it later through savings account.
- **Loan.java –** This class provides details of the various types of loans that are offered by the bank. The details include the status of the loan, the amount of loan taken, amount paid uptil now and the amount remaining to be paid.
- **EducationalLoan.java –** This class provides educational loan details. This includes the id of the customer that has requested for the loan, name of the university for which the loan is requested and the ROI associated with the loan.
- **CarLoan.java –** This class provides car loan details such as the id of the customer who has requested for the loan, name of the car for which the loan is taken and the ROI associated with it.
- **HomeLoan.java –** This class provides details such as the customer id, address of the house for which the loan is requested and the associated ROI.

SQL TABLES

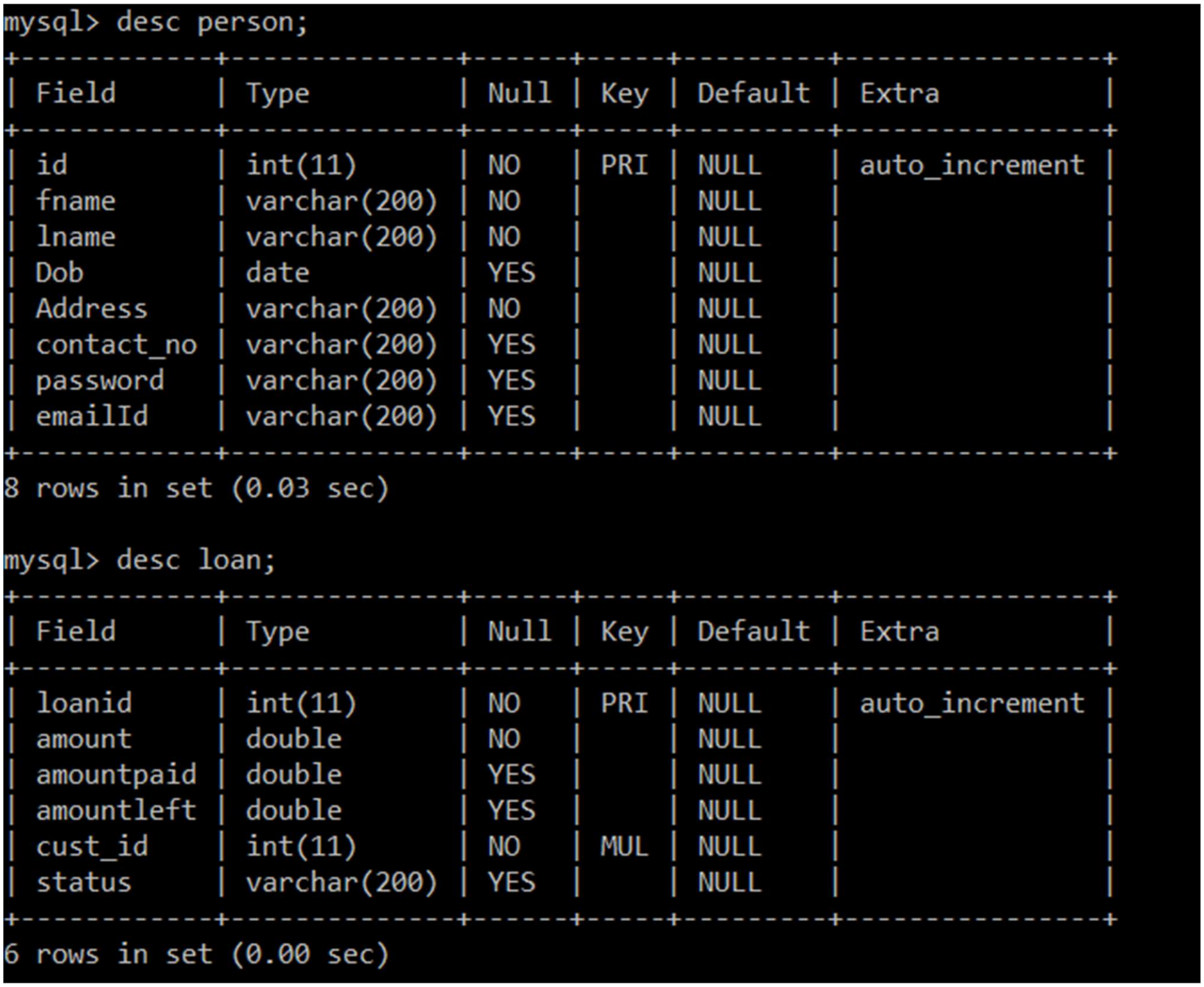
Following are the list of tables present in our Database

DATABASE TABLES	DESCRIPTION
Account	Contains information about every customer's accounts.
SavingsAccount	Contains savings account information of every customer. This is a subclass of the Account class. This table is created using the JOINED strategy.
CheckingAccount	Contains checking account information of every customer. This is a subclass of the Account class. This table is created using the JOINED strategy.
person	Contains all the basic details of every registered user.
customer	Contains customer information. This is a subclass of the Person class. This table is created using the JOINED strategy.
manager	Contains information about the manager. This is a subclass of the Person class. This table is created using the JOINED strategy.
personaccount	This table keeps track of which customer owns which accounts.
card	Contains information about the credit and debit card details of the customer, if the customer has issued them.
creditcard	Contains credit card information of every customer. This class is a subclass of the card class. This table is created using the JOINED strategy.
debitcard	Contains debit card information of every customer. This class is a subclass of the card class. This table is created using the JOINED strategy.
Loan	Contains information about the loans that are issued by various customers.
EducationalLoan	Contains information about the educational loan purchased by various customers. This is a subclass of the Loan class. This table is created using the JOINED strategy.
CarLoan	Contains information about the car loan purchased by various customers. This is a

	subclass of the Loan class. This table is created using the JOINED strategy.
HomeLoan	Contains information about the educational loan purchased by various customers. This is a subclass of the Loan class. This table is created using the JOINED strategy.

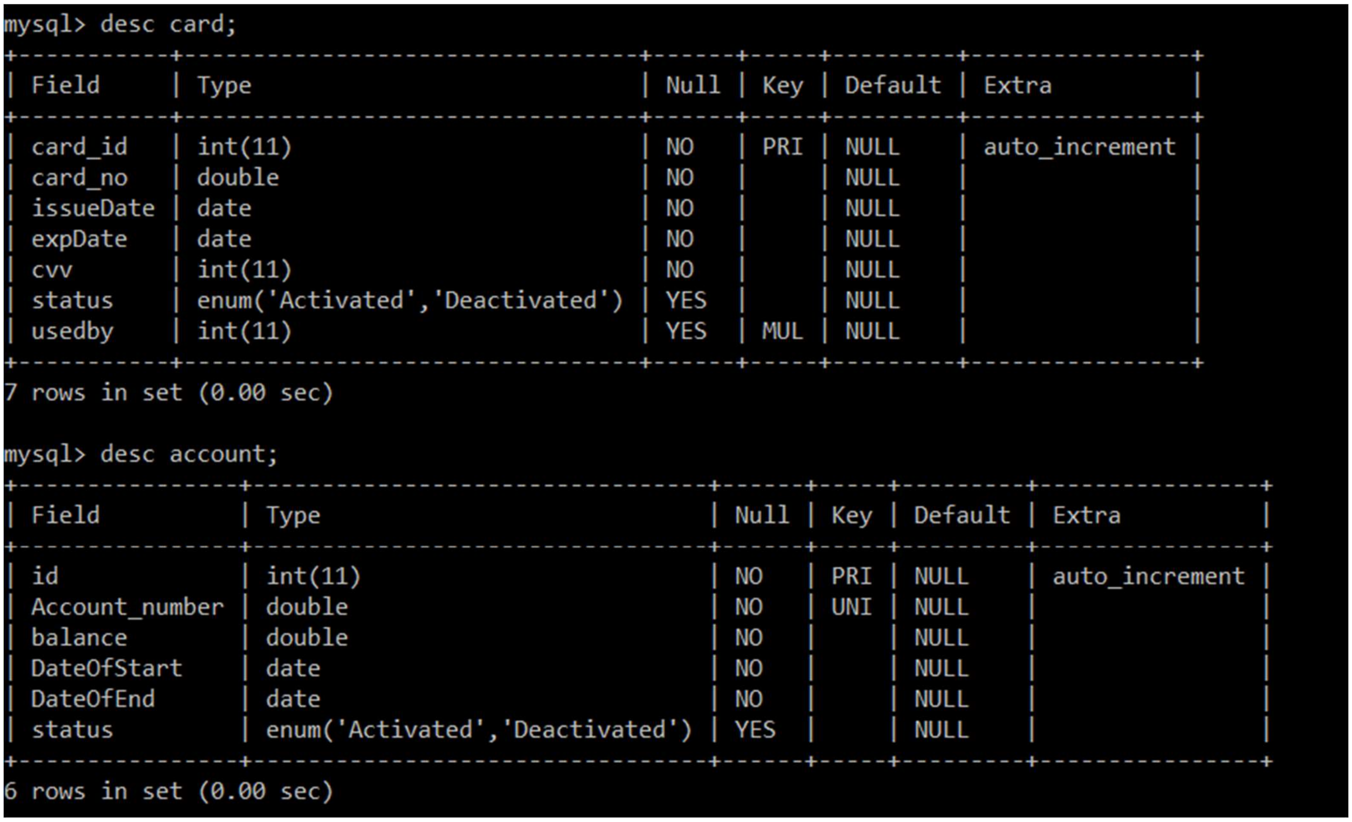
<u>DESCRIPTION OF THE MYSQL DATABASE TABLES</u>

Below are the screenshots of the main tables in the database.



```
mysql> desc person;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| id         | int(11)      | NO   | PRI | NULL    | auto_increment |
| fname      | varchar(200) | NO   |     | NULL    |                |
| lname      | varchar(200) | NO   |     | NULL    |                |
| Dob        | date         | YES  |     | NULL    |                |
| Address    | varchar(200) | NO   |     | NULL    |                |
| contact_no | varchar(200) | YES  |     | NULL    |                |
| password   | varchar(200) | YES  |     | NULL    |                |
| emailId    | varchar(200) | YES  |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
8 rows in set (0.03 sec)

mysql> desc loan;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| loanid     | int(11)      | NO   | PRI | NULL    | auto_increment |
| amount     | double       | NO   |     | NULL    |                |
| amountpaid | double       | YES  |     | NULL    |                |
| amountleft | double       | YES  |     | NULL    |                |
| cust_id    | int(11)      | NO   | MUL | NULL    |                |
| status     | varchar(200) | YES  |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```



```
mysql> desc card;
+-----------+-------------------------------+------+-----+---------+----------------+
| Field     | Type                          | Null | Key | Default | Extra          |
+-----------+-------------------------------+------+-----+---------+----------------+
| card_id   | int(11)                       | NO   | PRI | NULL    | auto_increment |
| card_no   | double                        | NO   |     | NULL    |                |
| issueDate | date                          | NO   |     | NULL    |                |
| expDate   | date                          | NO   |     | NULL    |                |
| cvv       | int(11)                       | NO   |     | NULL    |                |
| status    | enum('Activated','Deactivated') | YES  |     | NULL    |                |
| usedby    | int(11)                       | YES  | MUL | NULL    |                |
+-----------+-------------------------------+------+-----+---------+----------------+
7 rows in set (0.00 sec)

mysql> desc account;
+----------------+-------------------------------+------+-----+---------+----------------+
| Field          | Type                          | Null | Key | Default | Extra          |
+----------------+-------------------------------+------+-----+---------+----------------+
| id             | int(11)                       | NO   | PRI | NULL    | auto_increment |
| Account_number | double                        | NO   | UNI | NULL    |                |
| balance        | double                        | NO   |     | NULL    |                |
| DateOfStart    | date                          | NO   |     | NULL    |                |
| DateOfEnd      | date                          | NO   |     | NULL    |                |
| status         | enum('Activated','Deactivated') | YES  |     | NULL    |                |
+----------------+-------------------------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

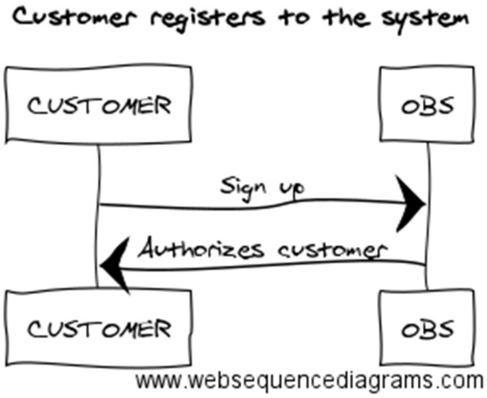
DISCUSSION

Few instances of the online banking covered in the project are:

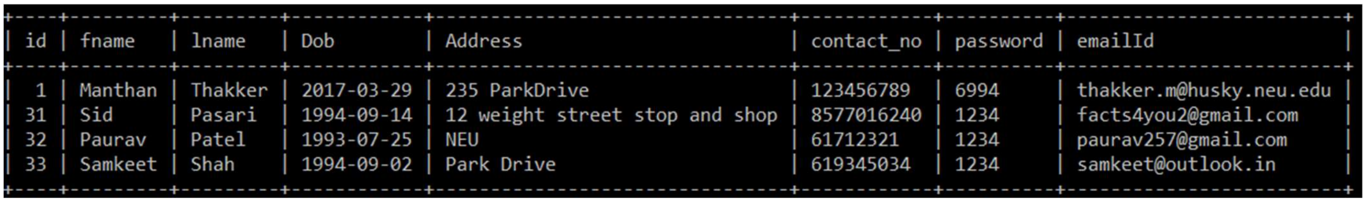
- **Customer Sign-Up**

  1. Description:
     The OBS prompts the user to input certain user information such as Password, LNAME, FNAME, DOB, Email Address etc. After which the OBS authorized the customer and provides him/her with his login credentials.
     Once the customer gets registered in the system, an email is sent to him on his registered email id. This email includes all the account details. This is implemented using SMTP API.

  2. Sequence Flow:



3. Output:
   Here a new record with id **32** is created and his details are stored in the database table



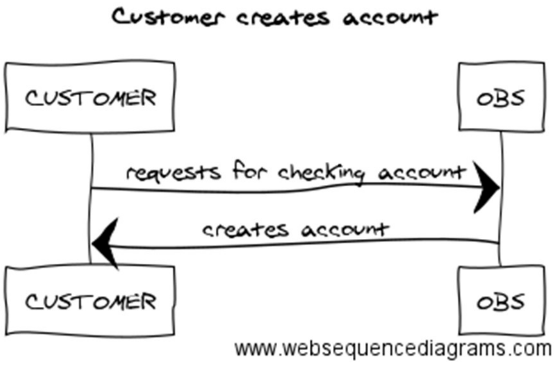
```
+----+---------+---------+------------+--------------------------------+------------+----------+-------------------------+
| id | fname   | lname   | Dob        | Address                        | contact_no | password | emailId                 |
+----+---------+---------+------------+--------------------------------+------------+----------+-------------------------+
|  1 | Manthan | Thakker | 2017-03-29 | 235 ParkDrive                  | 123456789  | 6994     | thakker.m@husky.neu.edu |
| 31 | Sid     | Pasari  | 1994-09-14 | 12 weight street stop and shop | 8577016240 | 1234     | facts4you2@gmail.com    |
| 32 | Paurav  | Patel   | 1993-07-25 | NEU                            | 61712321   | 1234     | paurav257@gmail.com     |
| 33 | Samkeet | Shah    | 1994-09-02 | Park Drive                     | 619345034  | 1234     | samkeet@outlook.in      |
+----+---------+---------+------------+--------------------------------+------------+----------+-------------------------+
```

➢ Account Creation and Transfer amount to another customer

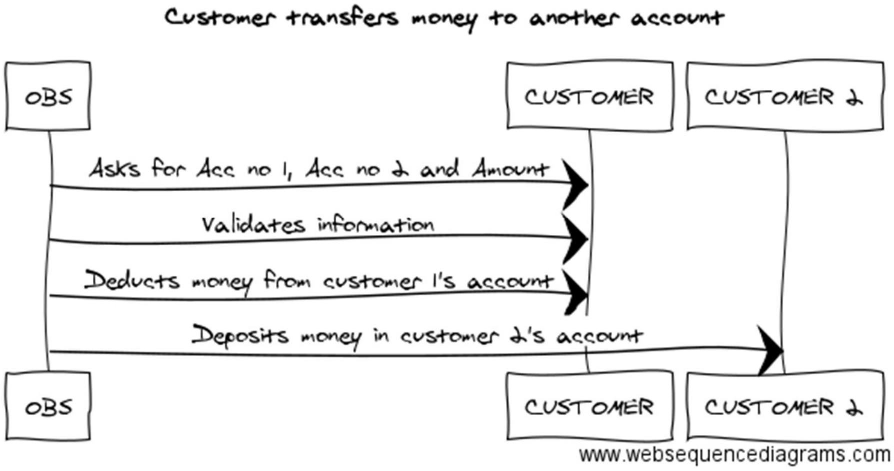
1. Description:
   The customer requests the OBS for creation of a checking account, to which the OBS accepts the request and creates the account for the customer.
   After the account is created successfully the user can deposit funds into his account and transfer funds to another customer.

2. Sequence Flow for Creation of account:

a) Sequence Flow for Account creation

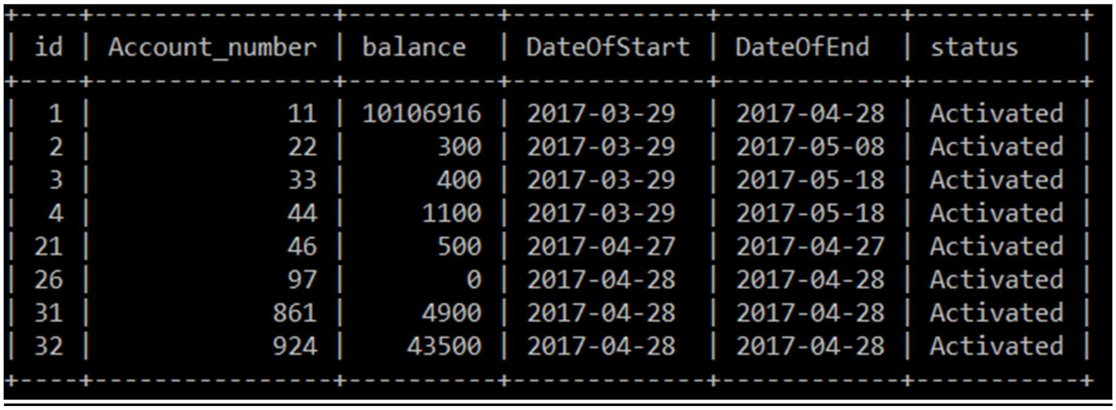


b) Sequence Flow for Transfer of funds to another Customer


Customer transfers money to another account
OBS
CUSTOMER
CUSTOMER 2
Asks for Acc no 1, Acc no 2 and Amount
Validates information
Deducts money from customer 1's account
Deposits money in customer 2's account
OBS
CUSTOMER
CUSTOMER 2
www.websequencediagrams.com

---

3. Output:
   The customer with id **32** has his account created with Account number: **924**
   The customer with id:**32** deposits $45000 into his account and later transfers $1500 to customer with id: **11**

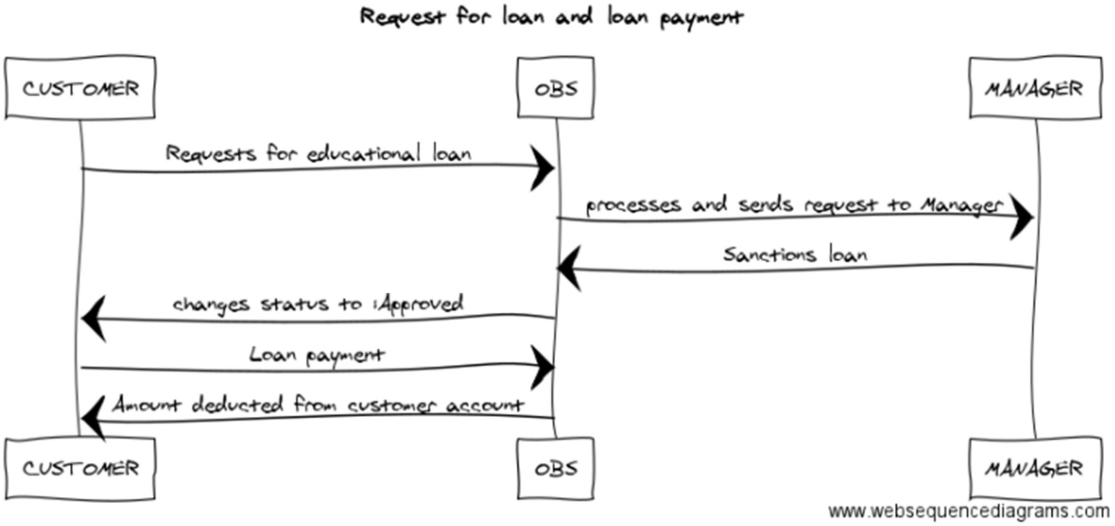


```
+----+----------------+----------+--------------+------------+-----------+
| id | Account_number | balance  | DateOfStart  | DateOfEnd  | status    |
+----+----------------+----------+--------------+------------+-----------+
|  1 |             11 | 10106916 | 2017-03-29   | 2017-04-28 | Activated |
|  2 |             22 |      300 | 2017-03-29   | 2017-05-08 | Activated |
|  3 |             33 |      400 | 2017-03-29   | 2017-05-18 | Activated |
|  4 |             44 |     1100 | 2017-03-29   | 2017-05-18 | Activated |
| 21 |             46 |      500 | 2017-04-27   | 2017-04-27 | Activated |
| 26 |             97 |        0 | 2017-04-28   | 2017-04-28 | Activated |
| 31 |            861 |     4900 | 2017-04-28   | 2017-04-28 | Activated |
| 32 |            924 |    43500 | 2017-04-28   | 2017-04-28 | Activated |
+----+----------------+----------+--------------+------------+-----------+
```

Loan Processing and Payment

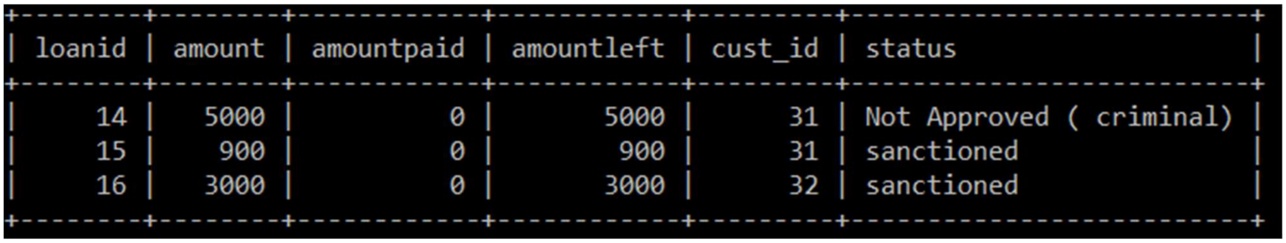
Description:
The customer id: **32** requests the Manager for loan sanction and the Manager Sanctions the Loan if the eligibility criteria is met.

Sequence Flow:


Request for loan and loan payment
CUSTOMER
OBS
MANAGER
Requests for educational loan
processes and sends request to Manager
Sanctions loan
changes status to :Approved
Loan payment
Amount deducted from customer account
CUSTOMER
OBS
MANAGER
www.websequencediagrams.com

Output:

The Manager accepts the loan request of cust_id: **32** of amount $3000



```
+--------+--------+------------+------------+---------+-------------------------+
| loanid | amount | amountpaid | amountleft | cust_id | status                  |
+--------+--------+------------+------------+---------+-------------------------+
|     14 |   5000 |          0 |       5000 |      31 | Not Approved ( criminal) |
|     15 |    900 |          0 |        900 |      31 | sanctioned              |
|     16 |   3000 |          0 |       3000 |      32 | sanctioned              |
+--------+--------+------------+------------+---------+-------------------------+
```

CONCLUSION

The implementation successfully achieved the goal of building an Online Banking System which ensures that a user can bank on the go, thereby enhancing a user's banking experience. It allows the user to make transfers, request and pay loans etc. We are planning to develop a front-end portal to automate the entire process.

REFERENCES

- http://stackoverflow.com/
- http://www.ccs.neu.edu/home/kenb/db/topics.html
- http://www.ccs.neu.edu/home/kenb/
- https://piazza.com/northeastern/spring2017/cs520001/resources