**Database and ERP systems final project**

**Online Banking System**

Members:

Prathiphan Dubey 57070503454

Passavich Limgritnuwat 57070503477

Prayuth Patumcharoenpol 57070503476

Saket Khandelwal 57070503483

Naruethep Sripumra 57070503474

**Abstract**

In our project, we created an online banking system based on real life online banking system. This include all possible feature a banking system could have and have features likes interaction between admin and customer where admin have supreme power over any other employee. The bank features are listed below with respect to person using it.

**Admin**

* Can see and edit all customer in the bank.
* Can see all account that is registered with the bank.
* Can see all message send and receive in the bank.
* Can see all detail about employee.
* Admin have power to remove any employee or user
* Admin can add new employee to the bank
* Admin have inbox same as every other employee.
* Admin can send email to any customer in the bank.
* Admin can see the analysis report of the bank.

**Employee**

* Can see and edit all customer in the bank.
* Can see all account that is registered with the bank.
* Can see all account of the other bank.
* Can sanction or delete loan requested by the user.
* Can see all going customer loan.
* Can see all payment made by the customer.
* Employee have inbox as any other user.
* Employee can send email to any customer in the bank.

**Customer**

* Customer can register to the bank by themselves. (Add 10000 to account because we don’t have real account database).
* Customer cannot transfer money to the to their own account number.
* Customer can transfer money to their other account.
* Customer cannot transfer money more than they have.
* Customer cannot transfer money in the negative value.
* Customer can transfer money to another bank payee registered with the bank.
* Customer have option to at transfer money at the same time or can transfer later with two days processing time.
* Customer need to enter transaction password to proceed with the transaction.
* Customer can see their account statement with respect to the account number they choose.
* Customer can select between two date to see their account statement.
* Account statement show amount and transaction history of both incoming and outgoing transfer.
* Customer have inbox like any other user.
* Customer can email to any admin or employee in the bank.
* Customer can see their ongoing loan in the bank.
* Customer can see the payment they made for the loan.
* Customer can see all information on type of loan provided by the bank.
* Customer can send request for loan to the employee. The loan can be pass or reject depend on the employee.
* Customer can add new payee of the bank that is registered with the bank.
* Customer can add new account number to the database.

**Changes**

Changes from old design, the difference is that Admin can’t edit the data. Only one change regarding user role.

We added new tables like pre loan and pre transaction for transfer and loan request purposes.

The pre loan table is where the loan requests are stored in order to let the employee decide and verify it.

We added pre transaction to interact with the pay later process.

**Tools**

Tools we used to accomplish the project are as follows: -

* XAMP to provide basic workspace to work on with.
* HTML to write basic text and form.
* CSS to decorate up the website
* Bootstrap to use set of CSS class that is already defined.
* SQL to fetch, insert and update the database.
* PHP to write all of our backend of the website.
* JavaScript to smooth out and redirect the web page.
* PHPmyadmin to use as interface to interact with the database.

**Detail information on implementation**

Here we are going to explain the detail explanation on implementation technique. We divide it into three part as follow: -

**Admin**

**Customer in bank**

* The data is fetch from the customer table showing all information about the customer
* There is a delete button on each customer name
* When delete is pressed the sql execute delete command and the customer is deleted

**Account in bank**

* This is simple sql fetch from the customer table where it shows all detail about the customer

**Mail details of bank**

* This page shows all massage that has been send amongst the user and customer
* We used three table to join customer , employee and the mail
* Mail table only provide information about customer ID and Employee ID
* Using customer and employees table to join with mail allow to fetch name from the customer and the employee.
* Then we use join to display all message in the bank.

**Employee in Bank**

* The data is fetch from the employee table showing all information about the employee
* There is a delete button on each employee name
* When delete is pressed the sql execute delete command and the employee is deleted

**Other bank**

* Other bank shows information about the other banks.
* There is 3 section Other Bank, Payee, Account type.
* The information from other bank is fetch from otherbank table
* The information from payee is fetch from payee table
* The information from Account type is fetch from the accounttypeother.

**Add new employee**

* Admin have to fill form required to add the customer
* The form send information to same page
* Using insert from sql the information is added to the employee table.

**Mail**

* In mail the admin can select any customer username using dropdown
* Admin have to fill subject and message that is needed to send to customer.
* Then the form send information to the same page for further processing.
* Using the information send by the form we query the EmployeeID of the username and insert all of data which include subject, heading , customerID, EmployeeID, MaildateTime to the table mail.
* To classify who sent the message we used the sendtype in the table mail as tiny int which represent 0 or 1 value. So if admin sent the message the sendtype will be 1. The data then will be saved in the database.

**Inbox**

* To show message received from the customer we fetch information from three table (Employee, Customer, mail)
* We joined three mention above to find the name of the sender, receiver and mail. Since the table mail only provide information CustomerID and EmployeeID and the mail content.
* We used select statement where the mail is sent to the customerID and have sendtype as 0
* As mention above the sendtype 0 means that customer sent the message.

**Employee**

**Customer in bank**

* The data is fetch from the customer table showing all information about the customer
* There is a delete button on each customer name
* When delete is pressed the sql execute delete command and the customer is deleted

**Account in bank**

* This is simple sql fetch from the customer table where it shows all detail about the customer

**Other bank**

* Other bank shows information about the other banks.
* There is 3 section Other Bank, Payee, Account type.
* The information from other bank is fetch from otherbank table
* The information from payee is fetch from payee table
* The information from Account type is fetch from the accounttypeother

**Loan Requested**

* In this page the loan requested by user is shown in this page.
* Employee can either sanction the loan or delete it.
* The requested loan is saved the preloan table.
* If sanction the preloan table is deleted and inserted to loan where the loan become active

**Loan Ongoing**

* It shows all loan which is active.
* It is fetch from loan table.
* Employee can select Customer ID and look up how many payment they paid from the table loanpayment.

**Mail**

* In mail the admin can select any customer username using dropdown
* Admin have to fill subject and message that is needed to send to customer.
* Then the form send information to the same page for further processing.
* Using the information send by the form we query the EmployeeID of the username and insert all of data which include subject, heading , customerID, EmployeeID, MaildateTime to the table mail.
* To classify who sent the message we used the sendtype in the table mail as tiny int which represent 0 or 1 value. So if admin sent the message the sendtype will be 1. The data then will be saved in the database.

**Inbox**

* To show message received from the customer we fetch information from three table (Employee, Customer, mail)
* We joined three mention above to find the name of the sender, receiver and mail. Since the table mail only provide information CustomerID and EmployeeID and the mail content.
* We used select statement where the mail is sent to the customerID and have sendtype as 0
* As mention above the sendtype 0 means that customer sent the message.

**USER/CUSTOMER**

**User mail implementation**

* In the mail implementation, first we give user to choose all user name of admin/employee by querying the information about the employee from the employee table.
* Then we provide user with the form to enter the heading and subject of the message and send the form to the same page.
* Using the information send by the form we query the EmployeeID of the username and insert all of data which include subject, heading , customerID, EmployeeID, MaildateTime to the table mail.
* To classify who sent the message we used the sendtype in the table mail as tiny int which represent 0 or 1 value. So if user sent the message the sendtype will be 0. The data then will be saved in the database.

**Inbox**

* To show message received from the either Admin/Employee we fetch information from three table (Employee, Customer, mail)
* We joined three mention above to find the name of the sender, receiver and mail. Since the table mail only provide information CustomerID and EmployeeID and the mail content.
* We used select statement where the mail is sent to the customerID and have sendtype as 1
* As mention above the sendtype 1 means that either Admin/Employee send the mail.

**Current Loan**

* In current there 3 three section
* First, we show on going loan which is fetch directly from the loan table and shown to the customer.
* Second, we show the payment done by the user just below the ongoing loan by fetching the table directly from loanpayment.
* Third, we have option for the loan payment and if user click on the button the modal will drop down and user can choose account can select and on clicking submit the money will be deducted from the account and the loan amount will also be updated.

**Loan Request**

* First on the we fetch the query the table loanttype and show all type of loan and information about it on a table.
* From the above information, the user can fill out the form which include loan type and loan amount.
* Once submitted, the information will be inserted to the preloan where the employee will further handle it.
* We use PHP to catch up any error such as loan amount cannot be greater them the loan amount bank set with.

**Add an external payee**

* In this page, we can another bank payee where the another must be registered with our bank.
* We have drop down for user to select and enter the detail about the other bank payee.
* Then we use insert function of sql to insert information to table payee.

**Add a new bank account**

* We provide user with a form to fill out and send the information with POST to the same file
* When data is received, we check the account no if account no already exist in the database.
* If account number already exist an error box is shown and customer is send back to the user page
* Else information is added to the account table

**Account Statement**

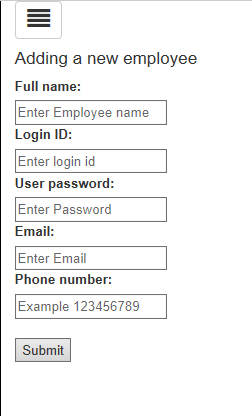
* We give dropdown information of bank account for user to select.
* Now the user can either choose two date to see their account statement according to their will or click submit directly without choosing the date.
* If clicked without date the data is fetched and shown
* The transfer table shown by using select statement where sendtype is 0 (sendtype 0 indicate that user transfer the money).
* The information is fetch from the transaction table.
* The fund received fund table shown by using select statement where sendtype is 1 (sendtype 1 indicate that user received the money).

**Transfer Money**

* We have form for user to enter and select.
* User can select Bank, (include another bank that is registered to the evil bank)
* User has to enter account number of the recipient and amount.
* User can choose to pay now or pay later
* Then to proceed user has to enter transaction password
* Now in the backend it checks amount, account number and transaction password if it is valid or not.
* Now if all information is valid the transaction backend proceeds.
* If user choose Paynow the amount will be deducted and added to amount of the recipient.
* If user choose Paylater it will take two-day process for amount to be added to the recipient.
* The amount however will be deducted anyways.
* If the Paylater transaction is successful, the data is inserted to the pretransaction.
* After two days, the transaction from the pretransactions will added to the transaction table.
* The old data from pretransaction will be deleted.
* If paynow is successful, the data will directly be inserted to the transaction table.
* If everything is a success the alert box will appear using JavaScript and will be directed to the user page.
* If error happen error box will appear using same JavaScript and user will be shown with an error and will be directed to the user page.

**Complex Forms**

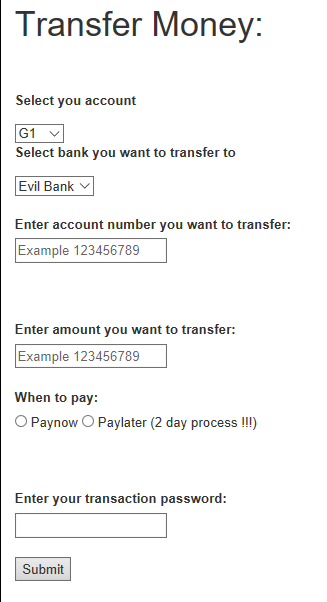
1. Adding a new Employee for admin



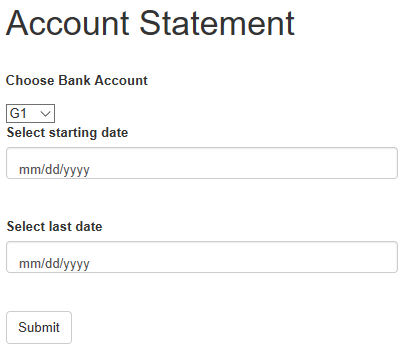
1. Composing New mail for all users



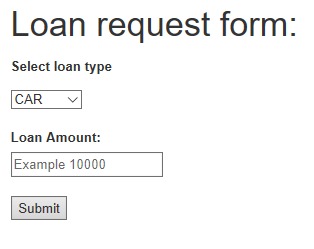
1. Transfer Money (Customer)



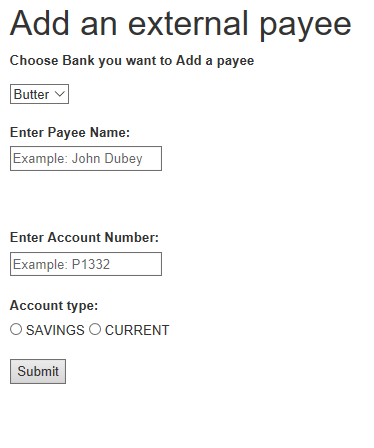
1. Account Statement Request (Customer)



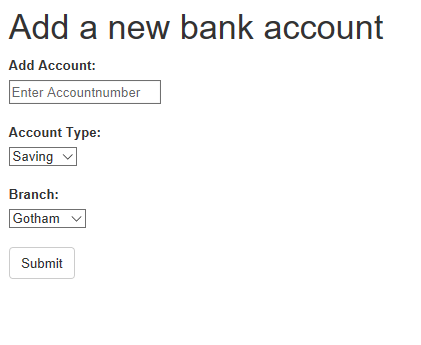
1. Loan request Form (Customer)



1. Add External Payee (Customer)



1. Add a new bank Account (Customer)



**Analysis**

1. **Find highest account balance**



SQL: SELECT a.Accountnumber,a.Accountbalance ,c.Firstname,c.LastName FROM customers c,accounts a WHERE a.CustomerID = c.CustomerID AND a.Accountbalance = (SELECT MAX(Accountbalance) FROM accounts) ORDER BY a.Accountnumber

1. **Find minimum account balance**



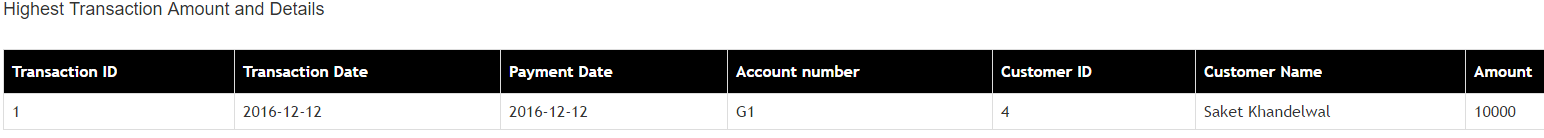
SQL: SELECT a.Accountnumber,a.Accountbalance ,c.Firstname,c.LastName FROM customers c,accounts a WHERE a.CustomerID = c.CustomerID AND a.Accountbalance = (SELECT MIN(Accountbalance) FROM accounts) ORDER BY a.Accountnumber

1. **Find number of account that register with each bank**



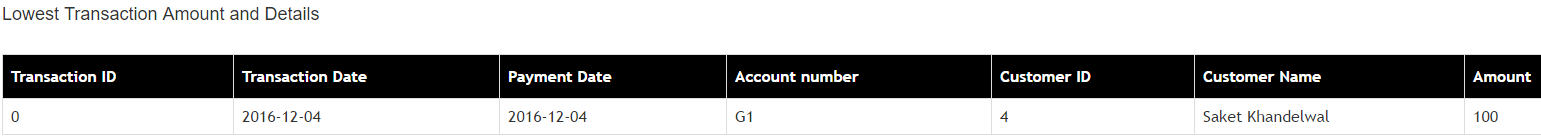
SQL: SELECT b.Branchcode ,COUNT(b.Branchcode) AS NUM FROM branch b INNER JOIN accounts a ON a.Branchcode = b.Branchcode GROUP BY b.Branchcode

1. **Find transaction that have highest amount and show it detail**



SQL: SELECT t.TransactionID,t.TransactionDate,t.PaymentDate,t.Accountnumber,c.CustomerID,c.Firstname,c.Lastname,t.Amount FROM transaction t,accounts a,customers c WHERE sendtype = 0 AND t.Accountnumber = a.Accountnumber AND a.CustomerID = c.CustomerID AND t.Amount = (SELECT MAX(Amount) FROM transaction)

1. **Find transaction that have lowest amount and show it detail**



SQL: SELECT t.TransactionID,t.TransactionDate,t.PaymentDate,t.Accountnumber,c.CustomerID,c.Firstname,c.Lastname,t.Amount FROM transaction t,accounts a,customers c WHERE sendtype = 0 AND t.Accountnumber = a.Accountnumber AND a.CustomerID = c.CustomerID AND t.Amount = (SELECT MIN(Amount) FROM transaction)

1. **Find average amount of transaction and number of transactions**



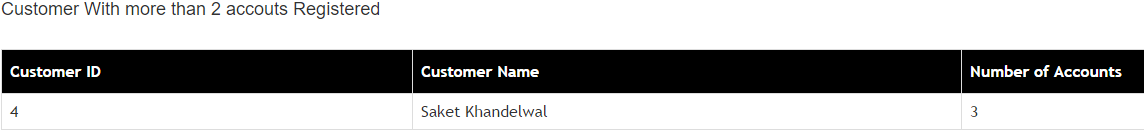
SQL: SELECT AVG(Amount) AS num,COUNT(TransactionID) AS numberof FROM transaction WHERE sendtype = 0

1. **Find number of external payee registered**



SQL: SELECT COUNT(CustomerOtherID) AS other FROM payee

1. **Find customer that have more than 2 account**



SQL: SELECT c.CustomerID,c.Firstname,c.Lastname,COUNT(Accountnumber) AS numberacc FROM accounts a,customers c WHERE a.CustomerID = c.CustomerID GROUP BY c.CustomerID

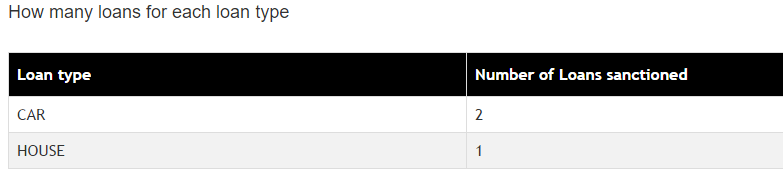
Then we use php to check if the nmber of accounts are more than two.

1. **Find number of transactions transferred to external payee and Total Amount**

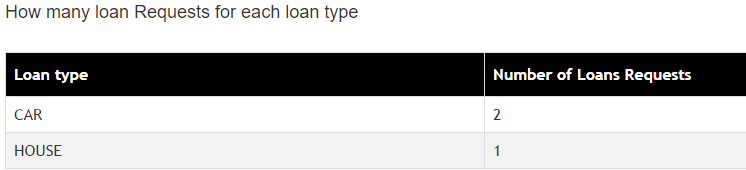


SQL: SELECT COUNT(TransactionID) AS num,SUM(Amount) as am FROM transaction WHERE sendtype = 1 AND CustomerOtherID !='NULL'

1. **Find how many loans for each loan type**

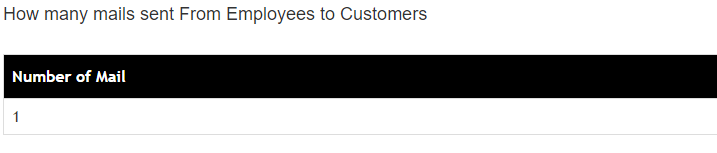


SQL: SELECT loantype,COUNT(LoanID) AS id FROM loan GROUP BY loantype

1. **Find how many loan requests for each loan type** 

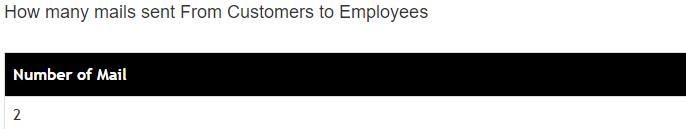
SQL: SELECT loantype,COUNT(PreloanID) AS id FROM preloan GROUP BY loantype

1. **Find how many mails sent from employees to customers**



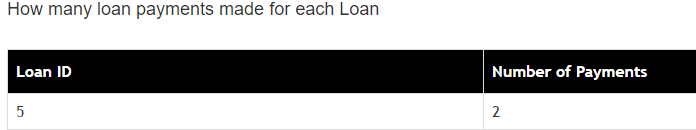
SQL: SELECT COUNT(MailID) AS id FROM mail WHERE sendtype = 1

1. **Find how many mails sent from customers to employees**



SQL: SELECT COUNT(MailID) AS id FROM mail WHERE sendtype = 0

1. **Find how many loan payments made for each Loan**



SQL: SELECT LoanID, COUNT(PaymentID) AS id FROM loanpayment GROUP BY LoanID

1. Find employee has not logged in for at least a day



SQL: SELECT EmployeeID,Employeename,lastlogin FROM employees ORDER BY EmployeeID

This analysis from using strtotime function to convert lastlogin column from string to time then subtract it with local time and check if it more than one day show it in this table 7