2016

159.339 Assignment 2

REPORT

CHI FUNG STANLEY YEUNG (15316357)

Contents

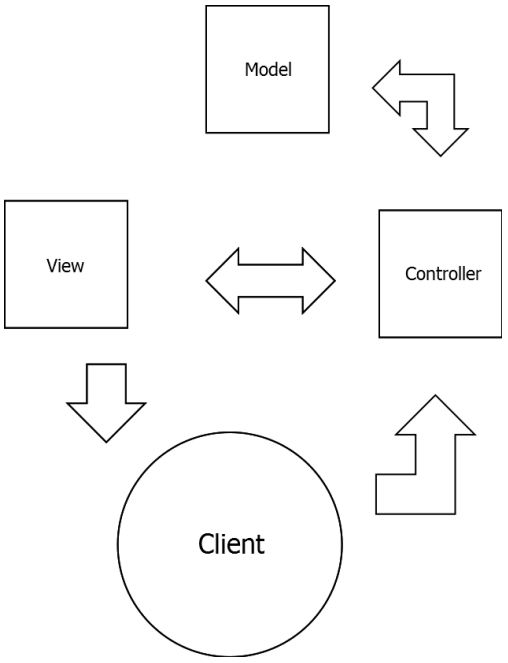
Specifications 2
Design Choices 2
Database schema and relations 4
Installation instructions 5
Instructions for end-user 5
Conclusion 7

Specifications

This is a simple bank application for “First National Bank”. The application provides the following functionalities:

1) Log-in / Sign-up / Log-out
2) Users logging in can ask the system to remember them for 30 days or until they log out manually
3) Users will be logged out automatically after 15 mins or inactivity
4) Add / Deleting accounts
5) View a list of accounts associated with the user with balance
6) View a list of transactions associated with the account
7) Perform transactions (deposit/withdraw)
8) Search accounts by ID or Balance
9) Sort accounts by ID or Balance
10) Created and added a fav icon

Design Choices



The bases of the application follow the MVC (Model-View-Controller) architecture. The Client first interacts with the controller, which will decide on what information from the model it needs and what view would the client interact with. The client sees the page from view and interacts with the controller. The application is built around this architecture.

The application is made persistent using both Sessions and Cookies variables. I chose to use Cookies to store the logged in user to allow timing logins. I understand that this is dangerous as the cookie is stored on the client and may cause security issues. Due to this being an assignment, this is ignored, but if further development is done, this will be the a first priority that needs to be attended.

There are also Session variables used. The base controller class is stored as a session variable. This is done so that when the client closes the browser and decides to prompt for the server again, all variables are reset and the client starts the session new. (With login if the time limit is not exceeded.)

```php
private function logInAction()
{
    if(isset($_POST['user_name'], $_POST['user_pass'])){
        $user_id = $this->user_model->logIn($_POST['user_name'], $_POST['user_pass']);

        if (isset($_POST['remb'])) {
            $time = time() + (60 * 60 * 24 * 30);   // 30 days
            setcookie('remb', 'y', $time, "/");
        } else {
            $time = time() + (60 * 15);             // 15 mins
        }

        setcookie('user_name', $_POST['user_name'], $time, "/");
        setcookie('user_id', $user_id, $time, "/");

        header("Location: index.php");
    }
    include ("views/loginview.php");
}
```

The code snippet on the left shows the setup of the $_COOKIE variable for ‘user_name’ and ‘user_id’.

The application follows OOP (object orientated programming). The controllers and models are contained as classed objects. There is also a user, account, and transaction class used to store these objects, functions in these classes also allow modifying the information stored and or provide linkage to other functions or parts of this application.

The code snippet on the right shows the Account class. It is a class for 1 Account variable, which contains id and balance. There is also a function to modify the balance with additionally parsed parameters.

```php
class Account
{
    private $id;        # Account ID
    private $bal;   # Account Balance

    /*******************
    // Constructor
    ********************/
    public function __construct($id, $amount)
    {
        $this->id = intval($id);
        $this->bal = floatval($amount);
    }

    /*******************
    // FUNCTIONS
    ********************/
    public function getId()
    {
        return $this->id;
    }

    public function getBal()
    {
        return $this->bal;
    }

    public function depositWithdraw($type, $amount)
    {

        // Checks for correct transaction input.
        if (!ctype_alpha($type) || strlen($type) != 1)
            throw new Exception("Invalid Type input: " . $type);
        if (!is_numeric($amount))
            throw new Exception("Invalid Amount input: " . $amount);

        // Checks if amount is negative
        if ($amount < 0)
            throw new Exception("Negative Transaction Amount");

        // Identify deposit or withdraw. Case insensitve.
        if ($type == 'D' || $type == 'd'){
            $this->bal += $amount;

        } else if($type == 'W' || $type == 'w'){
            //  Checks that there is enough balance to withdraw
            if ($this->bal < $amount)
                throw new Exception("Insufficient Balance");

            $this->bal -= $amount;

        } else {
            throw new Exception("Invalid Type");
        }
    }
}
```

The variables and inputs are checked and converted through ORM (Object Relational Mapping). Since the application follows OOP, the classes will ensure that the variables are in their suitable types before processing. An example of this is the above Account class, the __construct() function ensures that $id is an int value and $bal is a float value.

The database also ensures that the parameters are in the correct type. MySQL is capable of acting as OODBMS(Object Oriented Database Management System). The code snippet below shows the initiation of the ‘Accounts’ table. The variables ‘acc_id’ and ‘user_id’ are both int variables with a maximum of 11 digits, ‘acc_bal’ is a float variable with 10 characteristic digits and 2 decimal digits.

```
acc_id INT(11) NOT NULL AUTO_INCREMENT,
user_id INT(11) NOT NULL,
acc_bal FLOAT(10,2) NOT NULL,
```

Note that all inputs are sanitised before processed. The code below is used to sanitise the input variables.

```
mysqli_real_escape_string($this->conn, $input);
```

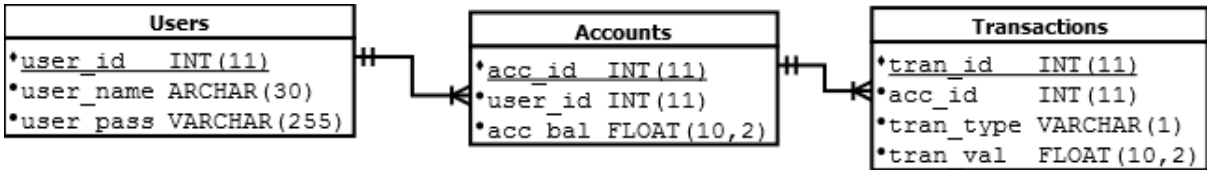
Database schema and relations

This application stores and retrieves its information from a MySQL database. The database was initialized with PHP. The PHP file used is “initializedb.php”.

The database itself is divided into 3 tables:

1) Users
   For storage of user information for login and account viewing
2) Accounts
   For storage of accounts information
3) Transactions
   For storage of all the transactions that occurred for different accounts

The tables are designed with one-to-many relationships. This is due to the nature of the data structure. A user can have zero to many accounts, but an account can only have one user. Similarly, an account can have zero to many transactions, but a transaction can only have one account. Below is a diagram of the tables within the database.



This relationship is processed through functions in PHP and also Foreign keys. These foreign keys link the tables in a CASCADE DELETE relationship. For example, if an account is deleted, all transactions associated to that account will be deleted as well. Although not implemented in the application, the users and accounts table also hold the same property.

Installation instructions

The application runs on MySQL and PHP, please ensure the server supports the above.

I stored necessary MySQL parameters in a file named “config.ini”. Please ensure that the following variables are correct.

The initial variables are:

servername = localhost

username = a2

password = a2

database = a2

The directory list on the right is a list of the files in the application.

The application was tested through vagrant on Windows 10 OS. The virtual machine was provided by the lecturer.

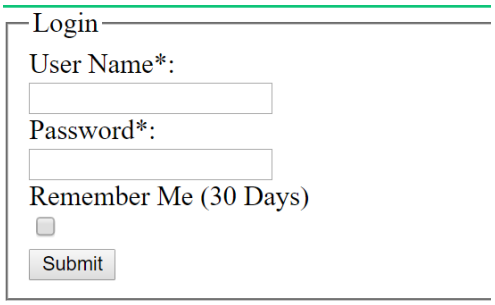
```
config-ini.php
configsql.ini
dump.sql
index.php
initializedb.php
Report.docx

controllers
    acccontroller.php
    basecontroller.php
    trancontroller.php
    usercontroller.php

models
    accmodel.php
    connectdb.php
    tranmodel.php
    usermodel.php

    obj
        account.php
        transaction.php
        user.php

views
    accview.php
    baseview.php
    homeview.php
    loginview.php
    signupview.php
    tranview.php

    css
        base.css
        table.css

    img
        download.jpg
        icon.ico
        icon.png
        icon1.ico
        icon1.png
```

Instructions for end-user

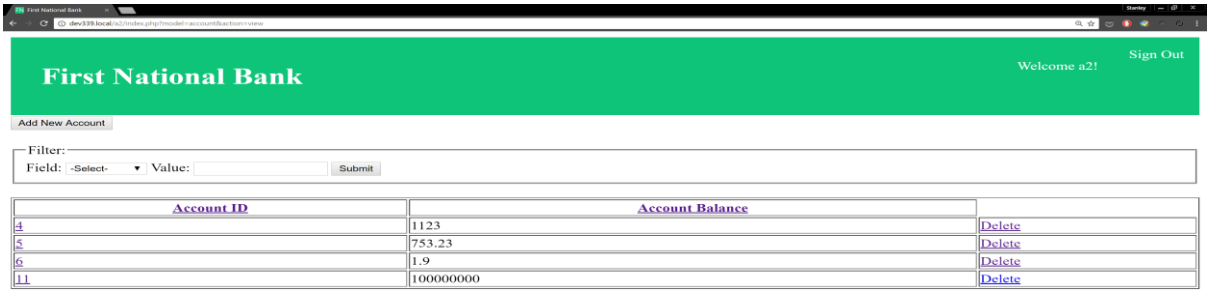
Please access the application by “\index.php”.

The application initialises with the following page:

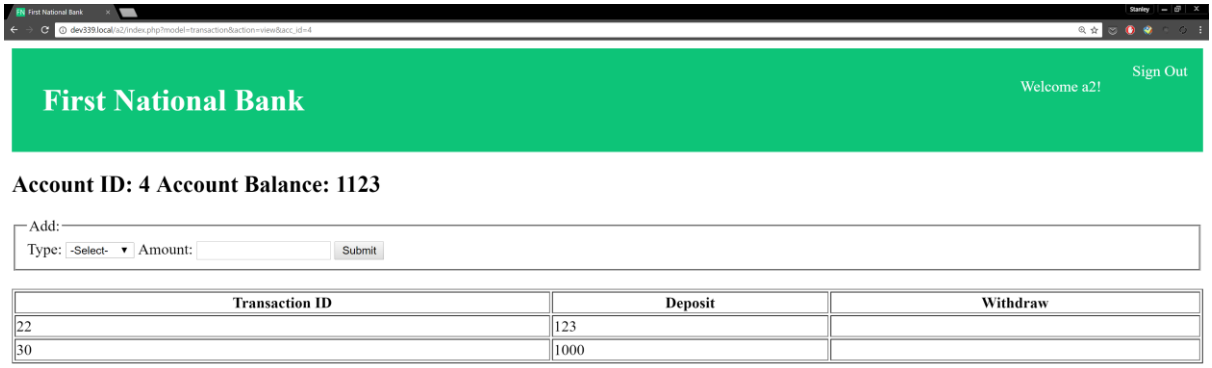
Please login via the options on the top right, “Sign In” or “Register”. The below is the “Sign In” page. Please check the “Remember me” box to allow persistent login for 30 days. A user “a2” with password “a2” has been created in the initial dump.sql.



After login, the main page will display a list of the accounts associated to the user. You can sort the fields by clicking the headers of the table. You can also search for an account using the account id and account balance.



Clicking on a account id will take you to a page that will show a list of transactions. Users can perform transactions by choosing type and inserting a value. After a new transaction has been added, the account balance will be updated.



Should you encounter any exceptions or errors, please try to go back to the previous page. Another option would be to click the “First National Bank” title of the navigation bar to refresh the application.

Conclusion

This application demonstrates the basics of the following concepts through implementation:

- Data modelling
- Model-View-Controller architecture
- Object-Relational mapping
- Relational database design and normalisation
- Persistence via Sessions/Cookies
- Object-oriented programming

Although the application does fulfil the requirements, it is far from complete. There are security issues with both login, users and also accounts information. There are no filters or sort functions for transactions. No dates were implemented when updating accounts and transactions. And much more. The application is developed in a way which more functionalities could be implemented without affecting the existing ones. Further development would allow a more complete version of the application.

Thank you for marking my assignment.