# 159.339

# Assignment 2 Report

**Team**

Katie Dempsey - 15309679

Tony Crompton - 17288296

Chris Bishop - 93067924

Our team has developed a simple banking application under the name KTC Banking - For the purpose of providing an example showing understanding of PHP/MySQL, MVC, ORM, Relational database design, persistence, and OOP.

## **Specifications**

* Log-in / create user / log-out
* Separate views for admin and users
* Create user account and banking accounts
* View a list of accounts (along with the current balance for each)
* Deposit and withdraw from said banking accounts
* View banking transactions of a selected account and the amount withdrawn or deposited and the date/time for when transaction was made

This application provides a simple example of how a bank might implement online banking for its customers. It allows customers to create a user account that can login and logout, create new bank accounts for that user, make transactions such as depositing and withdrawing, and view accounts and transactions made.

The UI is simple and easy to use - with appropriate controls for whether you are an administrator or a customer. The App provides simple tabs that users can click on that link to appropriate pages.

The program follows the MVC framework and divides the related program logic into their respective elements - Model, View, Controller. A router provided by the lecturer is used to direct traffic to the appropriate destinations.

A SQL database stores all data made by administrators and customers. The database and tables are filled with some sample data at the first creation of the application.

**Specifications set by the assignment outline**

The code used must be PHP 7.3 and MySQL 8 compatible.

Must use Docker - to create, deploy, and run the application using containers.

Must implement the coding style guidelines set forward by PSR-1 and PSR-12

standards – As referred by http://www.php-fig/psr/.

The database must be created (if it doesn’t exist) and populated with required tables and sample data in the code.

Must use the example MVC application provided as a base for the code.

Database connection - Must use the database name a2 and access it using user root with password root.

Files must be arranged in the following way:

* Controllers in *controller* subdirectory
* Models in *model* subdirectory
* Views in *view* subdirectory

Code must demonstrate an understanding of the following concepts through their practical application:

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

## **Design Choices**

**Passwords**

For securing passwords - When a customer creates a user account, the password they choose will be hashed using the php password\_hash() function which creates a new hash using a strong one-way hashing algorithm. It will use the BCRYPT algorithm to create the hash. Which will produce a standard crypt() compatible hash using the "$2y$" identifier. The result will always be a 60 character string, or FALSE on failure. As we are not manually entering a salt, a random salt will be generated by the password\_hash() function for each password hashed.

**CSS & website Layout**

The CSS used is simple but effective. With clear buttons and links to different pages. And unobtrusive colourings or stylings. Most forms and data are presented to the user in a table format for easy understanding and formality. There are two separate main views that can be accessed - the admin view and the customer view. This allows for the administrator to oversee and change user accounts. Some features include:

* Admin have full access and can delete accounts
* Customers have limited access and only see information relevant to them
* Customers can not delete accounts, only request to close accounts

**Database Layout**

The database is quite simple and self explanatory - with only three tables. An account table which holds all the different accounts users may have. A transaction table which holds all the transactions that are made on the accounts. And a user table that holds all the data about the users. The relationships between these tables are explained and shown below.

## **Database Schema & Relationships**

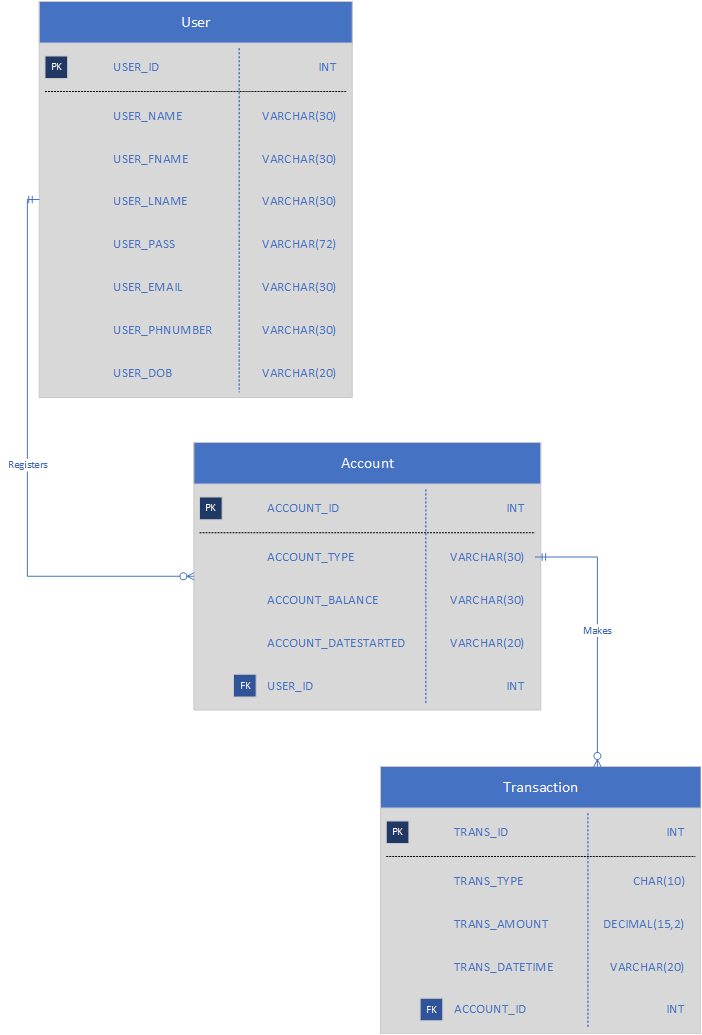
As shown in the diagram below -

PK - Primary Key

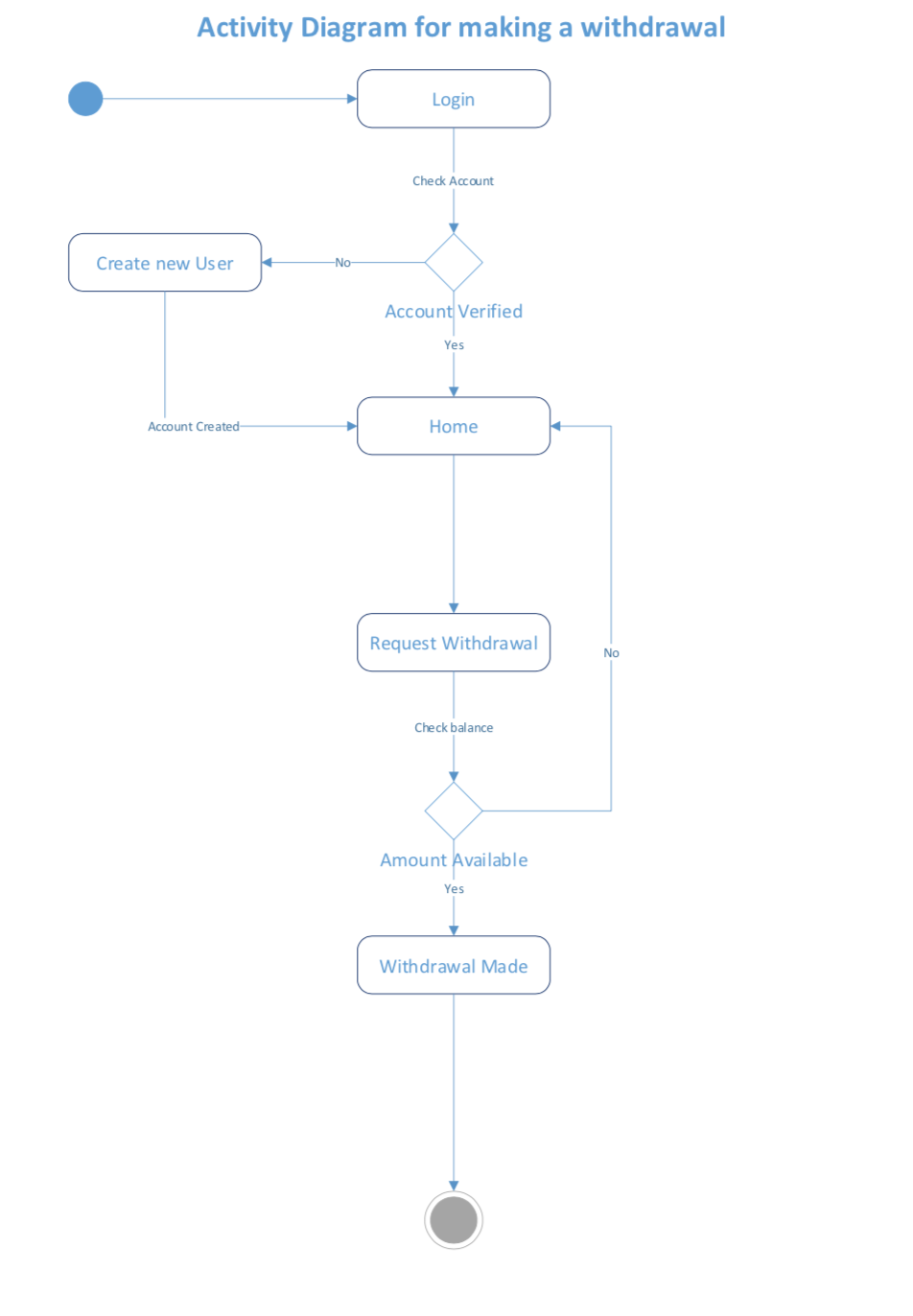
FK - Foreign Key

Between the User table and the Account table a one and only one notation on one side of a relationship (User) and a zero or one on the other (Account).

Between the Account table and the Transaction table a one and only one notation on one side of a relationship (User) and a zero or one on the other (Account).



The activity diagram below shows what processes are done when a customer makes a transaction on an account. This particular diagram shows the process of making a withdrawal. Verification checks are made on login and when the transaction request is made.

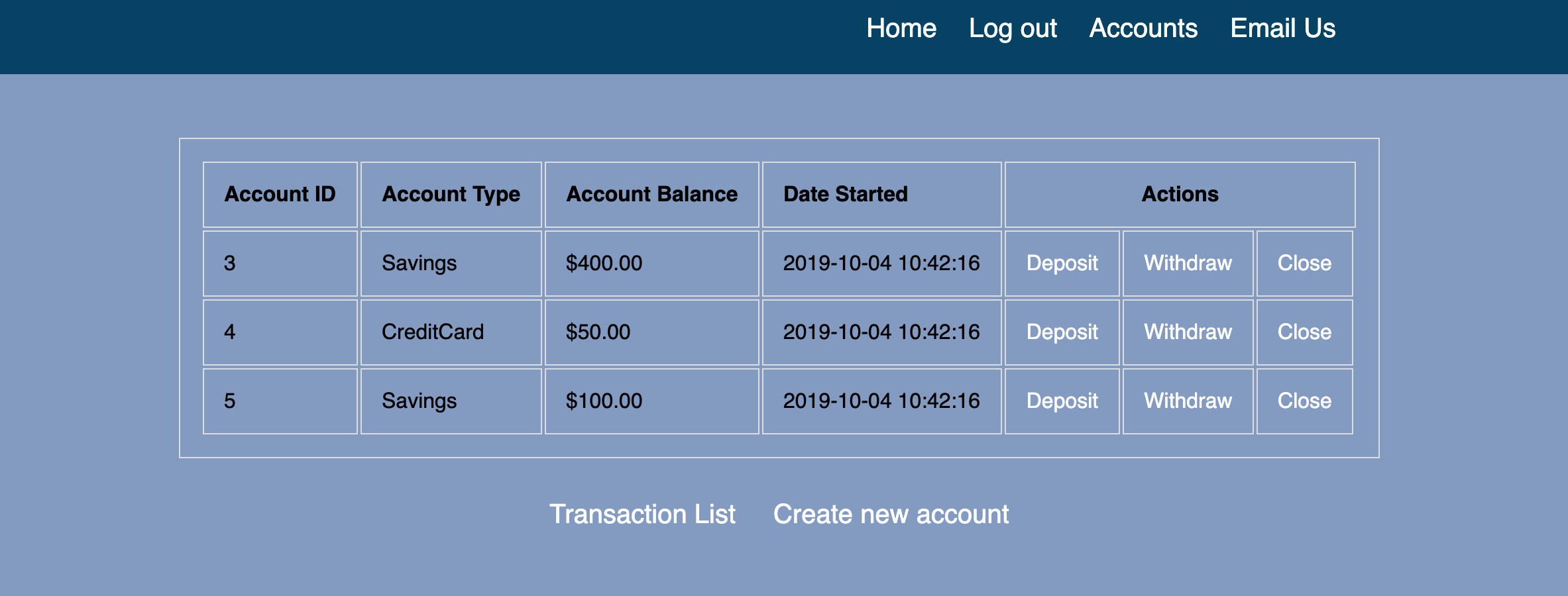


## **Installation Instructions**

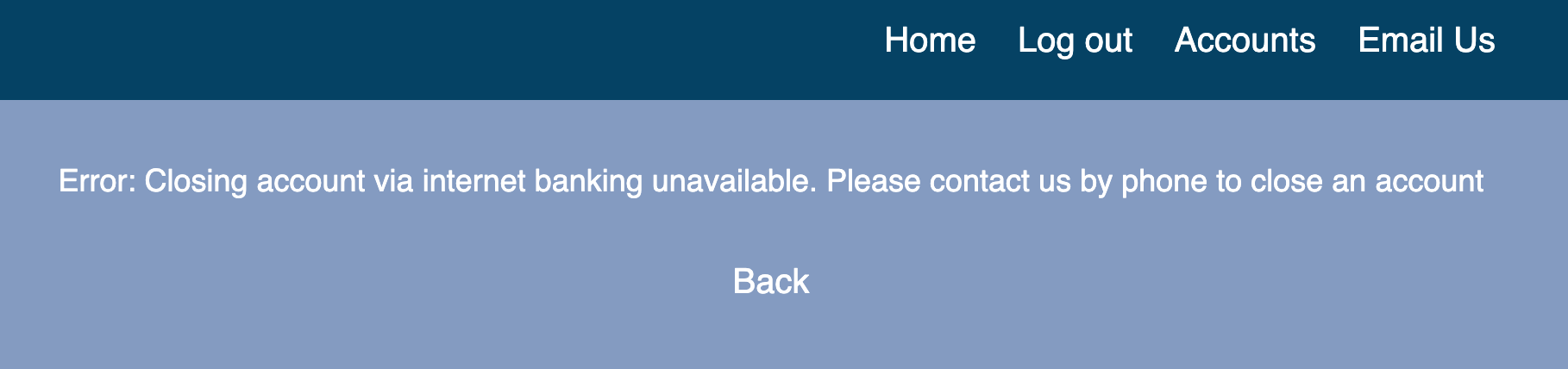
Standard for this assignment. Login with admin p/w admin for admin view.

## **Instructions For End-User** (with screenshots for clarity if needed)

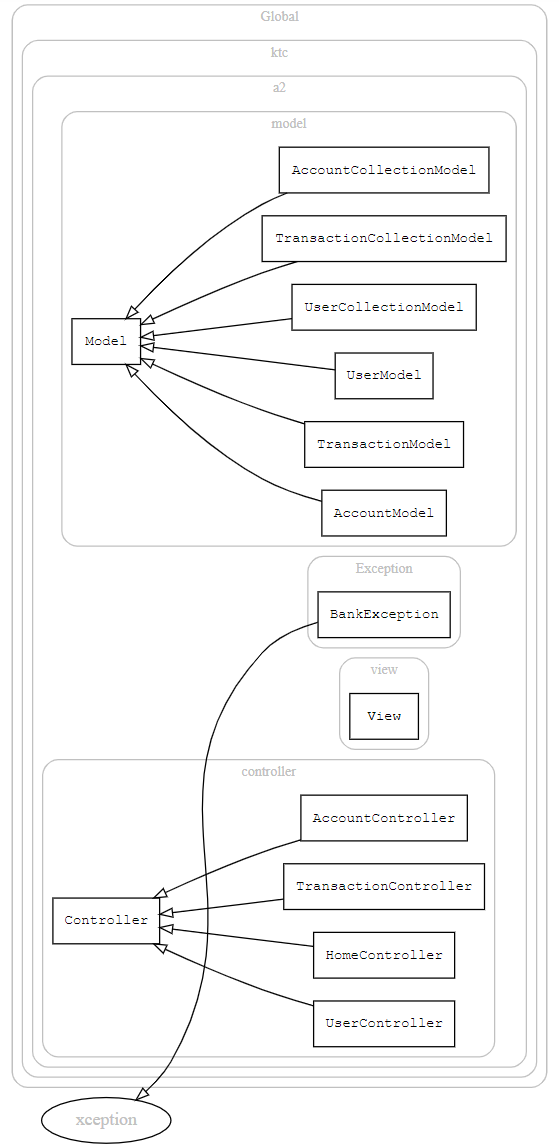
When first viewing this website, you will be greeted with a login page. If you do not have a login you can create one by clicking on the “Create a user” button below the login form. Once logged in, you can create an account to start depositing and withdrawing through the appropriate actions shown next to the accounts - As shown in the screenshot below.

You can access what transactions have been made through the “Transaction List” button below the account table. Next to this button is the “Create new account” button where you can create a new account and choose what account type you would like. 

Clicking on the “Close” button currently brings up the following error page shown below. As this change must be approved by an administrator.



**Class Hierarchy Diagram**

****