# Bank Portal Report

Ruhul Amin

Cis 344 81LC[54131]

Professor Yanilda Peralta Ramos

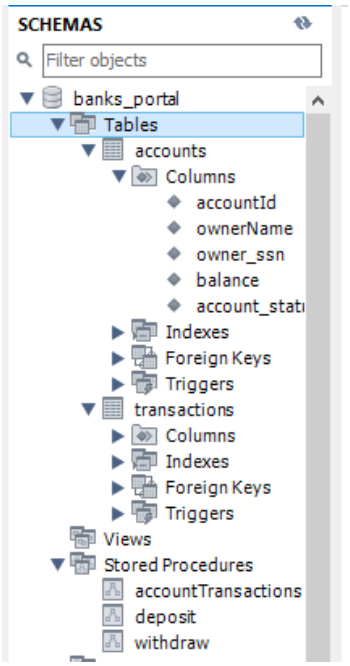
GitHub repository link: https://github.com/united789/Ruhul

## Introduction

This report shall look to define the overall development done in order to make a fully functional bank teller portal that works to help add accounts and both withdraw and deposit amounts. The functionality being used here is Python scripts with MySQL connectors to help define a backend database through a web portal. Stored procedures on the MySQL end are used to call certain functionalities for the bank teller.

## MySQL Setup and Procedures

The overall MySQL setup was done to ensure the development of a proper backend database that can be used to store banking information for the bank teller. The database was created with an accounts and transactions table that are populated thought he portal with the help of backend Python scripts in the portalServer.py file. The picture shown below provides the view of the schema from the MySQL workbench which was installed along with the server.



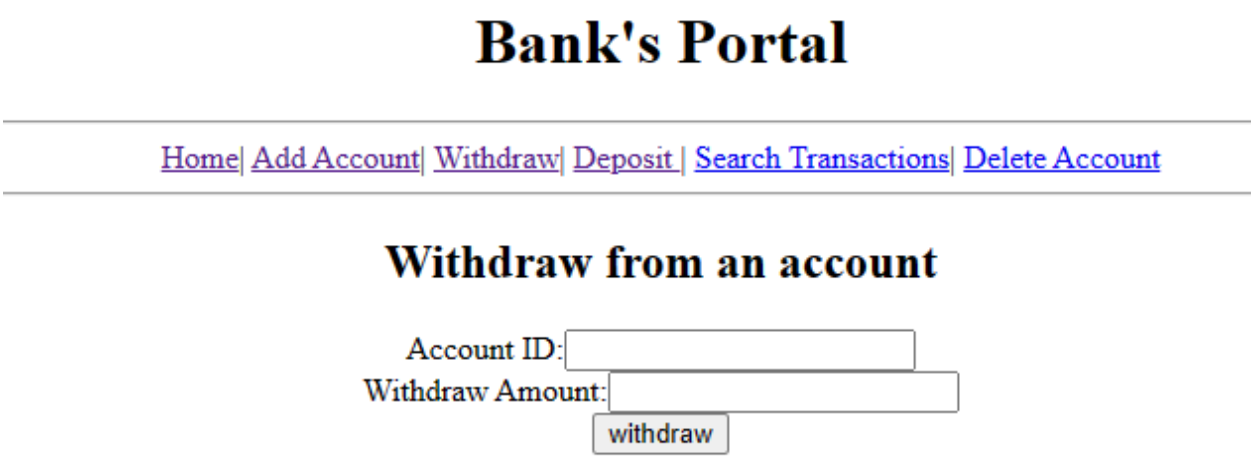
Some procedures were developed for account depositing and withdrawing as shown below.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `deposit`(IN param_accountID INT, IN amount
DECIMAL(10,2))
BEGIN
  START TRANSACTION;
  INSERT INTO Transactions (accountID, transactionType, transactionAmount)
  VALUES (accountID, 'deposit', amount);
  UPDATE accounts
  SET balance = balance + amount
  WHERE accountId = param_accountID;
  COMMIT;
END
```

This procedure for example is called whenever an amount is to be deposited in the bank account of any customer.

## Example of a Task: Withdrawing

To better explain some functionality, the withdraw tab is chosen to help call the withdraw procedure. The portal provided below is to be used to add the account ID and the amount that is to be withdrawn from it. This is made possible by the Post and Get functionality in the portalServer.py file.




The nature of Get and Post codes for this specific function is provided below.

GET:

```
if self.path == '/withdraw':
                self.send_response(200)
                self.send_header('Content-type','text/html')
                self.end_headers()
                self.wfile.write(b"<html><head><title> Bank's Portal
</title></head>")
                self.wfile.write(b"<body>")
                self.wfile.write(b"<center><h1>Bank's Portal</h1>")
                self.wfile.write(b"<hr>")
                self.wfile.write(b"<div> <a href='/'>Home</a>| \
                                <a href='/addAccount'>Add Account</a>|\
                                 <a href='/withdraw'>Withdraw</a>|\
                                 <a href='/deposit'>Deposit </a>|\
                                 <a href='/searchTransactions'>Search
Transactions</a>|\
                                 <a href='/deletAccount'>Delete
Account</a></div>")
                self.wfile.write(b"<hr><h2>Withdraw from an account</h2>")
                self.wfile.write(b"<form method='POST' action='/withdraw'>")
                self.wfile.write(b"<label for='accountId'>Account ID:</label>")
                self.wfile.write(b"<input type='number' name='accountId'><br>")
                self.wfile.write(b"<label for='withdraw_amount'>Withdraw
Amount:</label>")
```

```
                self.wfile.write(b"<input type='number'
name='withdraw_amount'><br>")
                self.wfile.write(b"<input type='submit' value='withdraw'>")
                self.wfile.write(b"</form>")
                self.wfile.write(b"</center></body></html>")
```

POST:

```
elif self.path == '/withdraw':
                self.send_response(200)
                self.send_header('Content-type','text/html')
                self.end_headers()
                form = cgi.FieldStorage(
                    fp=self.rfile,
                    headers=self.headers,
                    environ={'REQUEST_METHOD': 'POST'}
                )
                accountId = form.getvalue("accountId")
                print(accountId)
                withdraw_amount = int(form.getvalue("withdraw_amount"))
                print(withdraw_amount)
                self.database.withdraw(accountId,withdraw_amount)
                self.wfile.write(b"<html><head><title> Bank's Portal
</title></head>")
                self.wfile.write(b"<body>")
                self.wfile.write(b"<center><h1>Bank's Portal</h1>")
                self.wfile.write(b"<hr>")
                self.wfile.write(b"<div> <a href='/'>Home</a>| \
                                <a href='/addAccount'>Add Account</a>|\
                                 <a href='/withdraw'>Withdraw</a>|\
                                 <a href='/deposit'>Deposit </a>|\
                                 <a href='/searchTransactions'>Search
Transactions</a>|\
                                 <a href='/deletAccount'>Delete
Account</a></div>")
                self.wfile.write(b"<hr>")
                self.wfile.write(b"<h3>Amount has been withdrawn</h3>")
                self.wfile.write(b"<div><a href='/addAccount'>Add a New
Account</a></div>")
                self.wfile.write(b"</center></body></html>")
```

These two work in conjunction to provide the HTML page along with the calling of the function for withdraw procedure. An SQL connection cursor is then used to point towards the exact procedure in question which helps update the amount in the portal. The code for the database call procedure in the portalDatabase.py file is also shown below. This same code can be used for depositing and adding accounts with their particulars matched.

```
    def withdraw(self, accountId, amount):
        if self.connection.is_connected():
            self.cursor= self.connection.cursor()
            print("Connection Successful")
            self.cursor.callproc('withdraw', (accountId, amount))
        pass
```

## Conclusion

The overall exercise was highly fruitful and provided a proper method when it comes to the connection of Python with a local database instance. The database in this case was altered and modified according to the use case through the banking portal requests and stored procedures.