

**Banking System**  
Arrenn Baral, Prakriti Basyal  
Team 10  
CS 157A Summer 2020  
Instructor Mike Wu

## **Project Description**

This project will be a database application that is based in the general banking system. The goal of this application is to provide a convenient way to access user's banking information. Users will be able to login with designated username and password, access bank account, deposit, withdraw, get a report of transaction if a user wishes to. This application will also provide a user-friendly interface that is clear and intuitive. We use bank system in our daily lives. We login into bank systems like Chase, Wells Fargo, Bank of America, and list goes on. Deposits, withdrawals, transfers, statements are some of the example of our daily used operations on the bank systems. But we never thought what actually runs behind the screen. Since we are taking database class, we are getting great opportunity to explore system behind the screen. We got the motivation from our daily used operations in the bank system. Let's start with simple structure our structure of the banking system

### **Word on structure**

We will be Apache Tomcat webserver to run our banking system.

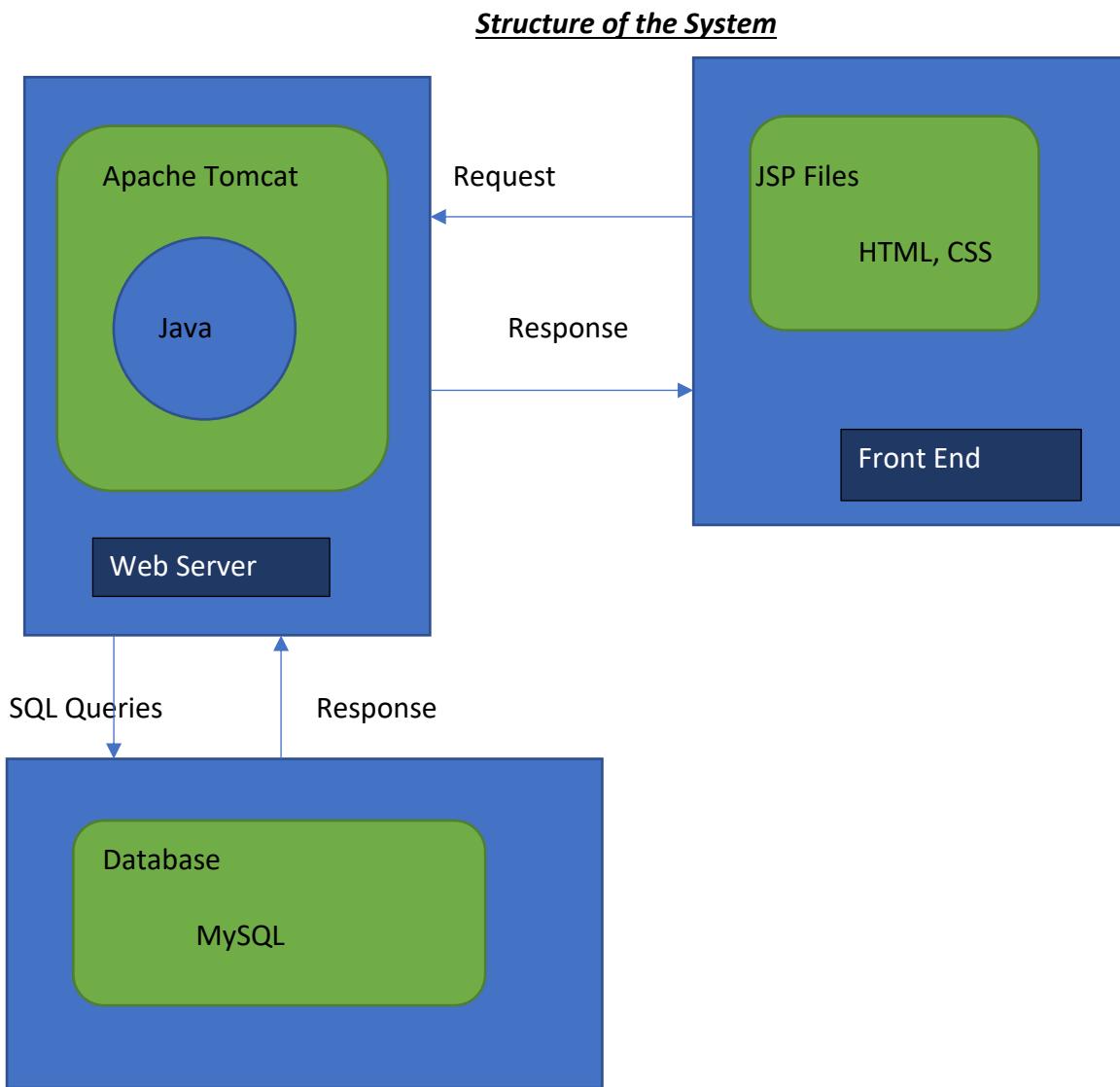
We will be implementing our code through Java IDE.

For database system, we are using MySQL but data entries.

We will be using DMLs, and DDLs for operations like deposits, withdrawals, and get statement.

We will be using HTML and CSS for beautify our banking system. It is important to give user-friendly look to users.

## System Environment



### Hardware and Software

Apache Tomcat

MacOS

### RDMS

MySQL Community Server 8.0.20

Application Languages

Java, HTML, CSS, PHP (maybe), SQL

## **Functional Requirements**

### ***How user will access the system***

The system provides functionality to create an account. User will be asked to set up a new account if he/she is new user else simply log in using their existing user credential in the database. If user wishes to setup a new account, the system opens new page for the user where the system asks several information for setting up the account such as First name, Last name, Date of Birth, Email Address, Gender, username, and password. Once the setup process completed, user will be redirected to banking system where they can select different operations. User cannot login if user's credential is not saved in the database. User cannot access without setting up the account. If user tries to access with fake username and password, they will be redirected to login page to use valid username or password. However, once user logs in, system will not ask to sign in again. System will assume user logged in through trusted device. Once logged in, users will be able to use several functions.

## **Functions**

### ***Log in***

User shall set up an account to get an access to the banking system.

User's credentials must be saved in the database.

### ***Withdraw***

Once logged in, user shall select the withdraw option to get their money.

Once user chooses this function, the withdrawal sum will be deducted from the total amount.

### *Deposit*

Once logged in, user shall use this function to deposit the money.

Once deposited, the deposited sum will be added to the total amount.

### *Statement*

User shall choose this function to get a statement of the transaction after successful withdrawal and deposits.

### *Apply for Loan*

User shall use this function to apply for a loan.

User gets approved only if thousand dollars is available in his bank account.

### *Order a cheque book*

User can use this function to order a cheque book.

There will be a charge of \$1.00 for ordering a check.

### *Manage/update account information*

User shall use this function to edit account information such as address, and phone number.

### *Transfer fund*

User shall use this function to transfer fund to other.

Receiving person must be setup before transferring.

### *Search statement by date*

If user wants to search statement from specific date, they can choose this function to select the date and search.

### *Sort the transactions*

If user wishes to sort the transactions from higher to lower or lower higher, they can use this function to sort as wished.

*Delete account.*

If user wishes to terminate the account, they can simply use this function to delete the account. Once deleted, account will not be recovered. It will be permanent.

*Display Account Information*

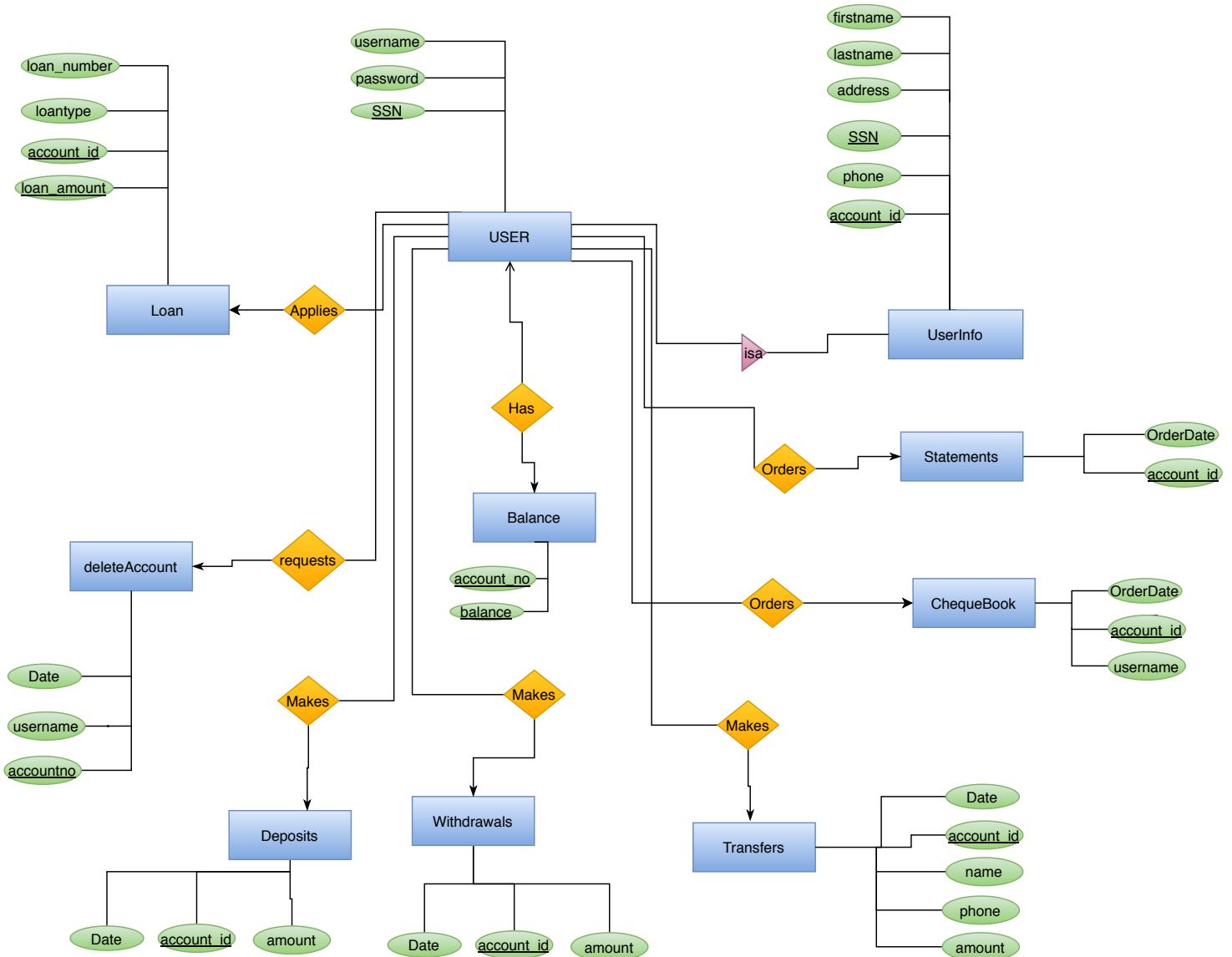
The user will be able to view the account information through this function.

## Graphics User Interface (GUI)

For the GUI, we will be using HTML to provide user-friendly environment. If possible, we will allow user to upload a picture of them for the account.

The security of each user account will be protected by a username and password created by the user. The username for each account will not be their email. User name will be the one which they choose while setting up the account. The user account information will be securely stored in the server. In order to login to the system the user must provide an existing username account along with the associated password. The website will be implemented using https providing an encrypted connection between user and server.

## ERD Diagram



***USERS:***

User entity is the service offered by the bank where one may deposit, withdraw or transfer money, check balance, apply loan, check statement in their account information. Every customer has a unique account. The entity set contains the customers username and password to login and the SSN. SSN is the primary key.

***USERINFO:***

The userInfo entity is the entity that holds the information of the user. This entity represents the unique customer information, which is the customer's firstname, lastname, address, phone, SSN, account\_id. The SSN and the account\_id is the primary key. It is child of userInfo so it inherits the attributes from the USERS.

***BALANCE:***

The Balance entity is the entity that represents the total amount of the fund in the user's account. Every user has a balance. It has two attributes the account\_no and the balance which gives them the information. Account\_id is primary key.

***LOAN:***

Loan entity represents lending money to one or more individuals with the required interest rate from the bank. The user can apply for a loan. So the loan entity has the attributes loan number, Loan\_type, account\_id, loan amount where account\_id is primary key.

***DELETEDACCOUNT:***

*This is an entity provided to the user as a service to delete an account. A user can request to delete the account if they no longer are interested in the service. Its attributes are date, username, and account\_no where account number is the primary key.*

***STATEMENTS:***

Statement entity represents a record of the financial activities done by the customer which are also given as the written reports. It basically keeps track of the record list of transactions done by the customer over a certain period of time. User can order the statement any time. So, orderdate and the account\_id are the attributes where account\_id is the primary key.

***CHEQUEBOOK:***

This entity is another service provided to the customer where user can order the cheque book providing their account\_id, and username. OrderDate keeps track when it was ordered.

***DEPOSITS:***

Deposit is entity set which is available to customer to deposit money into their bank account. These deposits can be made through cash, check by the customer. So, for the security of the customer's information. Deposits entity set has the account\_id, Date and the amount as an attribute.

***WITHDRAWALS:***

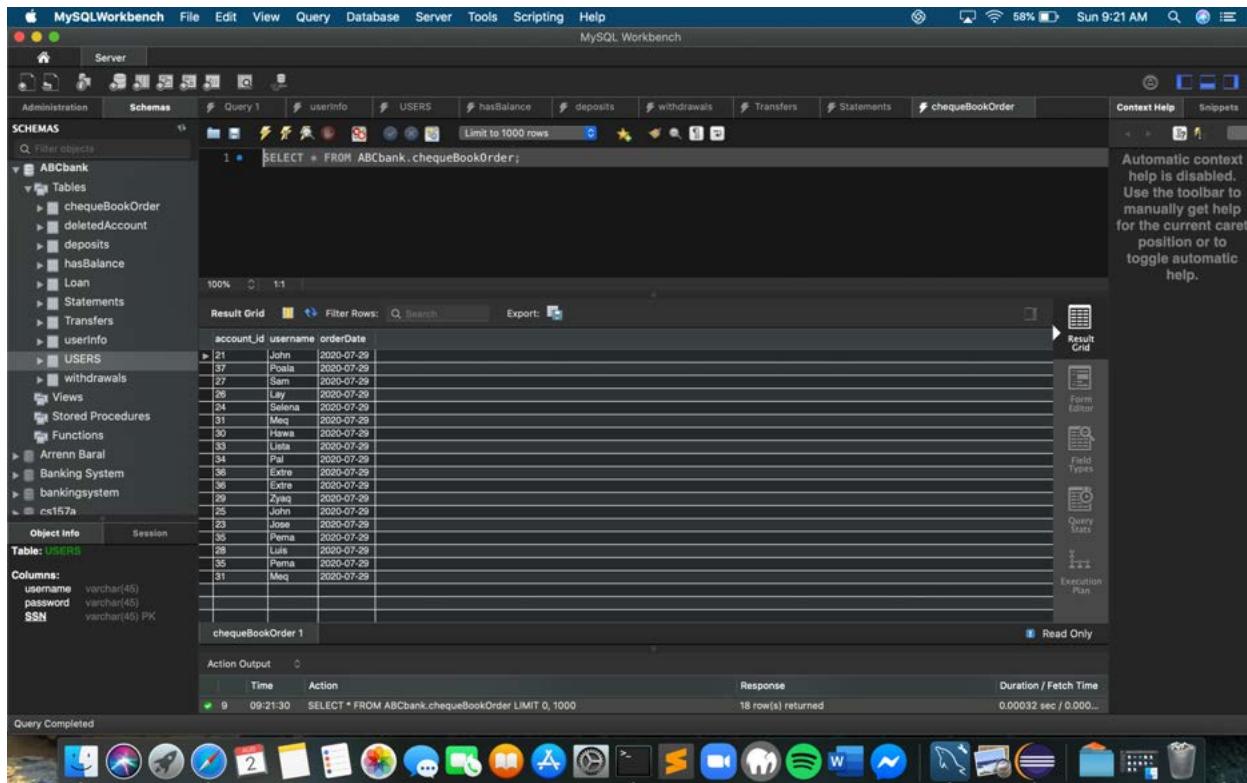
Withdrawal entity set represents giving the customer an option to withdraw money whenever they want. The bank grants access to the customer for the withdrawals except for the unusual activities. This entity set contains withdraw amount, account\_id and the DATE to withdraw from their account as an attribute.

## **TRANSFERS:**

This entity set represents the electronic fund transfer from one person to another to the bank accounts.

It is one of the services provided by the bank so that people can transfer, making people's lives easier.

This entity set has the amount, date, name, phone number and the account\_id so they can transfer directly to their account.



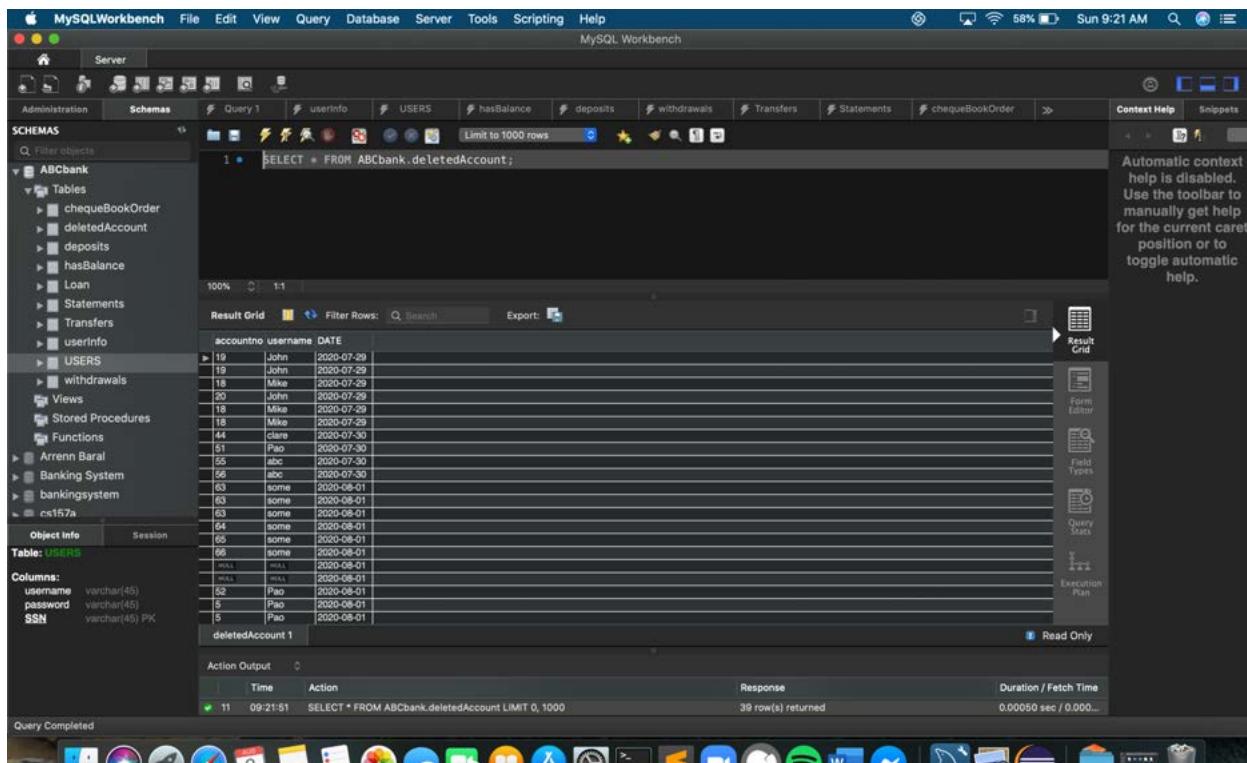
The screenshot shows the MySQL Workbench interface with the 'ABCbank' schema selected. A query window displays the following SQL statement:

```
SELECT * FROM ABCbank.chequeBookOrder;
```

The results grid shows the following data:

account_id	username	orderDate
21	John	2020-07-29
37	Paula	2020-07-29
37	Sam	2020-07-29
26	Lay	2020-07-29
24	Selena	2020-07-29
31	Meq	2020-07-29
30	Hawa	2020-07-29
33	Usta	2020-07-29
34	Pai	2020-07-29
36	Extre	2020-07-29
36	Extre	2020-07-29
29	Zyiaz	2020-07-29
25	John	2020-07-29
23	John	2020-07-29
35	Penna	2020-07-29
28	Luis	2020-07-29
35	Penna	2020-07-29
31	Meq	2020-07-29

The status bar at the bottom indicates the query completed successfully with 18 rows returned in 0.00032 sec / 0.000... ms.



The screenshot shows the MySQL Workbench interface with the 'ABCbank' schema selected. A query window displays the following SQL statement:

```
SELECT * FROM ABCbank.deletedAccount;
```

The results grid shows the following data:

accountno	username	DATE
19	John	2020-07-29
19	John	2020-07-29
18	Mike	2020-07-29
20	John	2020-07-29
18	Mike	2020-07-29
18	Mike	2020-07-29
44	clare	2020-07-30
51	Pao	2020-07-30
55	abc	2020-07-30
56	abc	2020-07-30
63	some	2020-08-01
63	some	2020-08-01
63	some	2020-08-01
64	some	2020-08-01
65	some	2020-08-01
66	some	2020-08-01
52	Pao	2020-08-01
5	Pao	2020-08-01
5	Pao	2020-08-01

The status bar at the bottom indicates the query completed successfully with 39 rows returned in 0.00060 sec / 0.000... ms.

MySQLWorkbench

File Edit View Query Database Server Tools Scripting Help

MySQL Workbench

Administration Schemas

Query 1 userinfo USERS hasBalance deposits

Limit to 1000 rows

Result Grid Filter Rows Search Export

account\_id amount DATE

account_id	amount	DATE
24	400	2020-07-29
25	400	2020-07-29
26	400	2020-07-29
27	400	2020-07-29
28	400	2020-07-29
29	400	2020-07-29
30	400	2020-07-29
31	400	2020-07-29
32	400	2020-07-29
33	120	2020-07-29
34	120	2020-07-29
35	400	2020-07-29
36	400	2020-07-29
37	100	2020-07-29
38	400	2020-07-29
39	120	2020-07-29
40	120	2020-07-29
41	120	2020-07-29
42	120	2020-07-29
43	120	2020-07-29
44	120	2020-07-29
45	120	2020-07-29
46	120	2020-07-29
47	120	2020-07-29
48	120	2020-07-29
49	222	2020-07-29
50	165	2020-07-29

deposit 1

Result Grid Filter Rows Search Export

Time Action Response Duration / Fetch Time

09-20-24 SELECT \* FROM ABCbank.deposits LIMIT 0, 1000 121 row(s) returned 0.00668 sec / 0.00000...

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Object info Session

Table: userinfo

Columns:

- username varchar(45)
- password varchar(45)
- SSN varchar(45) PK

Query Completed

This screenshot shows the MySQL Workbench interface with a query results window. The query executed was 'SELECT \* FROM ABCbank.deposits;'. The results show 121 rows of data with columns account\_id, amount, and DATE. The interface includes a sidebar with schema navigation, a toolbar with various icons, and a status bar at the bottom.

MySQLWorkbench

File Edit View Query Database Server Tools Scripting Help

MySQL Workbench

Administration Schemas

Query 1 userinfo USERS hasBalance

Limit to 1000 rows

Result Grid Filter Rows Search Export/Import

SSN balance

SSN	balance
1111	3781
1112	4740
1113	300
1114	4520
1115	6142
1116	4143
1117	4320
1118	4200
1119	4370
1120	15680
1121	562
1122	47
1123	2760
1124	5740
1125	4720
1126	6004
1127	10000
1128	4200
1129	140
1130	1000

hasBalance 1

Result Grid Filter Rows Search Export/Import

Time Action Response Duration / Fetch Time

09-20-24 SELECT \* FROM ABCbank.hasBalance LIMIT 0, 1000 36 row(s) returned 0.00034 sec / 0.000...

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Object info Session

Table: userinfo

Columns:

- username varchar(45)
- password varchar(45)
- SSN varchar(45) PK

Query Completed

This screenshot shows the MySQL Workbench interface with a query results window. The query executed was 'SELECT \* FROM ABCbank.hasBalance;'. The results show 36 rows of data with columns SSN and balance. The interface includes a sidebar with schema navigation, a toolbar with various icons, and a status bar at the bottom.

MySQLWorkbench

File Edit View Query Database Server Tools Scripting Help

MySQL Workbench

Administration Schemas

Query 1 userinfo USERS hasBalance deposits withdrawals Statements chequeBookOrder

Limit to 1000 rows

Result Grid Filter Rows Search Export/Import

loan\_number account\_id loantype loan\_amount

loan_number	account_id	loantype	loan_amount
123	36	Home	150000
124	37	Car	100000
125	34	Laptop	40000
126	33	Student	60000
127	32	Personal	80000
128	31	TV	30000
129	30	Bike	120000
130	29	Auto	90000
131	28	Watch	500
132	27	Phone	4000
133	26	AirPods	800
134	25	Phone	1000
135	24	Laptop	9000
136	23	Boat	150000
137	22	Holiday	150000
138	21	Auto	90000

Loan 1

Result Grid Filter Rows Search Export/Import

Time Action Response Duration / Fetch Time

09-21-24 SELECT \* FROM ABCbank.Loan LIMIT 0, 1000 16 row(s) returned 0.0026 sec / 0.00001...

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Object info Session

Table: userinfo

Columns:

- username varchar(45)
- password varchar(45)
- SSN varchar(45) PK

Query Completed

This screenshot shows the MySQL Workbench interface with a query results window. The query executed was 'SELECT \* FROM ABCbank.Loan;'. The results show 16 rows of data with columns loan\_number, account\_id, loantype, and loan\_amount. The interface includes a sidebar with schema navigation, a toolbar with various icons, and a status bar at the bottom.

MySQLWorkbench

File Edit View Query Database Server Tools Scripting Help

Schemas

Administration Schemas Query 1 userinfo USERS hasBalance deposits withdrawals Transfers Statements

Limit to 1000 rows

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid Filter Rows Search Export

Object Info Session

Table: **USERS**

Columns:

- username varchar(45)
- password varchar(45)
- SSN varchar(45) PK

Statements 1

Action Output Time Action Response Duration / Fetch Time

8 09:21:20 SELECT \* FROM ABCbank.Statements LIMIT 0,1000 55 row(s) returned 0.00032 sec / 0.000...

Query Completed

account_id	orderDate
23	2020-07-29
26	2020-07-29
27	2020-07-29
28	2020-07-29
29	2020-07-29
30	2020-07-29
31	2020-07-29
32	2020-07-29
33	2020-07-29
34	2020-07-29
35	2020-07-29
36	2020-07-29
37	2020-07-29
38	2020-07-29
39	2020-07-29
40	2020-07-29
41	2020-07-29
42	2020-07-29
43	2020-07-29
44	2020-07-29
45	2020-07-29
46	2020-07-30
47	2020-07-30
48	2020-07-30
49	2020-07-30
50	2020-07-30
51	2020-07-30
52	2020-07-30
53	2020-07-30
54	2020-07-30
55	2020-07-30

Statements 1

Action Output Time Action Response Duration / Fetch Time

8 09:21:20 SELECT \* FROM ABCbank.Statements LIMIT 0,1000 55 row(s) returned 0.00032 sec / 0.000...

Query Completed

MySQLWorkbench

File Edit View Query Database Server Tools Scripting Help

Schemas

Administration Schemas Query 1 userinfo USERS hasBalance deposits withdrawals Transfers

Limit to 1000 rows

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid Filter Rows Search Export

Object Info Session

Table: **Transfers**

Columns:

- account\_id recipient\_name recipient\_phon... amount DATE

Transfers 1

Action Output Time Action Response Duration / Fetch Time

7 09:21:07 SELECT \* FROM ABCbank.Transfers LIMIT 0,1000 33 row(s) returned 0.00040 sec / 0.000...

Query Completed

account_id	recipient_name	recipient_phon...	amount	DATE
33	Exte	100-000-0000	30	2020-07-29
30	Selena	200-000-0000	100	2020-07-29
30	Selena	300-000-0000	90	2020-07-29
30	John	400-000-0000	100	2020-07-29
21	Lay	500-000-0000	99	2020-07-29
27	Lay	600-000-0000	2	2020-07-29
27	Sam	700-000-0000	33	2020-07-29
22	Luis	800-000-0000	40	2020-07-29
22	Zyiq	900-000-0000	20	2020-07-29
30	Maria	110-000-0000	55	2020-07-29
31	Meg	120-000-0000	1000	2020-07-29
31	Meg	130-000-0000	920	2020-07-29
37	Poala	140-000-0000	50	2020-07-29
37	Poala	150-000-0000	50	2020-07-29
35	Penna	160-000-0000	100	2020-07-29
35	Penna	170-000-0000	158	2020-07-29
39	Mike Wu	123-456-7890	100	2020-07-30
39	Mike Wu	123-456-7890	100	2020-07-30
39	Mike Wu	123-456-7890	100	2020-07-30
47	Danny	444-444-4444	90	2020-07-30
47	Danny	444-444-4444	90	2020-07-30

Transfers 1

Action Output Time Action Response Duration / Fetch Time

7 09:21:07 SELECT \* FROM ABCbank.Transfers LIMIT 0,1000 33 row(s) returned 0.00040 sec / 0.000...

Query Completed

MySQLWorkbench

File Edit View Query Database Server Tools Scripting Help

Schemas

Administration Schemas Query 1 userinfo

Limit to 1000 rows

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid Filter Rows Search Export/Import

Object Info Session

Table: **userinfo**

Columns:

- idnewUser fname lastname username password address phone amount SSN

userinfo 1

Action Output Time Action Response Duration / Fetch Time

2 09:19:59 SELECT \* FROM ABCbank.userInfo LIMIT 0,1000 38 row(s) returned 0.0014 sec / 0.0006...

idnewUser	fname	lastname	username	password	address	phone	amount	SSN
21	John	Legend	John	John13	0 nowhere st	00010000	3791	1111
22	Sam	Thien	Sam	Sam12	12 nowhere st	00020000	4320	1112
23	Joe	Lene	Joe	Lene02	02 nowhere st	00030000	300	1113
24	Selena	Gomez	Selena	Selena1	03 nowhere st	00040000	4940	1114
25	Christina	Legend	John	John13	04 nowhere st	00050000	4520	1115
26	Lay	Man	Lay	Lay13	05 nowhere st	00060000	5142	1116
27	Sam	Thien	Sam	Sam12	06 nowhere st	00070000	4430	1117
28	Luis	Uziel	Uziel	Uziel02	07 nowhere st	00080000	4318	1118
29	Zyiq	Mari	Zyiq	Zyiq12	08 nowhere st	00090000	5100	1119
30	Rith	Hawa	Hawa	Hawa1	09 nowhere st	00011000	4370	1121
31	Penna	Meo	Meo	Meo12	10 nowhere st	00120000	15680	1122
32	Megan	Alegria	Alegria	Alegria12	11 nowhere st	00013000	592	1123
33	Son	Uziel	Uziel	Uziel12	12 nowhere st	00014000	4718	1124
34	Linda	Uziel	Uziel	Uziel12	13 nowhere st	00015000	2160	1125
35	Paula	Penna	Penna	Paula12	14 nowhere st	00016000	5740	1126
36	Fego	Exte	Exte	Exte12	15 nowhere st	00018000	3820	1127
37	Paco	Arta	Paco	paco12	16 nowhere st	00019000	6604	1128
38	Arrenn	Baral	arrenn...	barail	Nowhere Blvd	409-000	310	1234
39	Iom	Nehru	Iom	Iom12	111-111-1111	00020000	2334	1235
40	Craig	Uziel	Craig	Craig12	Nowhere CT	111-111-1111	200	3456
41	Devon	Chang	devon	chang12	There St	533-333-3333	5000	4567

userinfo 1

Action Output Time Action Response Duration / Fetch Time

2 09:19:59 SELECT \* FROM ABCbank.userInfo LIMIT 0,1000 38 row(s) returned 0.0014 sec / 0.0006...

MySQLWorkbench

File Edit View Query Database Server Tools Scripting Help

MySQL Workbench

Administration Schemas

SCHEMAS ABCbank

Tables: chequeBookOrder, deletedAccount, deposits, hasBalance, Loan, Statements, Transfers, userinfo

1 • SELECT \* FROM ABCbank.USERS;

Result Grid

username	password	SSN
José	Ler112	11112
John	Col1114	11114
John	John13	11115
Lay	Lay13	11116
Sam	Sam2	11117
Lulu	Lulu2	11118
Elys	Zyra1119	11119
James	James1	1121
Meiq	Meiq12	1122
Alegra	Alegra1	1123
Ulfra	Ulfra2	1124
Pat	Pat12	1125
Alma	Alma2	1126
Extra	Extra1	1127
Poolla	poella9	1128
Juan	juan12	11212
Demetrio	berral1	11234
abc	abc1	11236
charles	charles12	11244
mike	wu12	11252
Ivan	ivan1	11434

Object Info Session

Table: USERS

Columns:

username	varchar(45)
password	varchar(45)
SSN	varchar(45) PK

Query Completed

MySQLWorkbench

File Edit View Query Database Server Tools Scripting Help

MySQL Workbench

Administration Schemas

SCHEMAS ABCbank

Tables: chequeBookOrder, deletedAccount, deposits, hasBalance, Loan, Statements, Transfers, userinfo

1 • SELECT \* FROM ABCbank.withdrawals;

Result Grid

account_id	amount	DATE
90	40	2020-07-29
22	120	2020-07-29
21	120	2020-07-29
2	120	2020-07-29
97	323	2020-07-29
22	30	2020-07-29
23	700	2020-07-29
95	300	2020-07-29
98	80	2020-07-29
98	10000	2020-07-29
92	400	2020-07-29
33	796	2020-07-29
34	900	2020-07-29
34	340	2020-07-29
98	150	2020-07-29
40	20	2020-07-30
40	40	2020-07-30
40	450	2020-07-30
40	400	2020-07-30
47	70	2020-07-30

Object Info Session

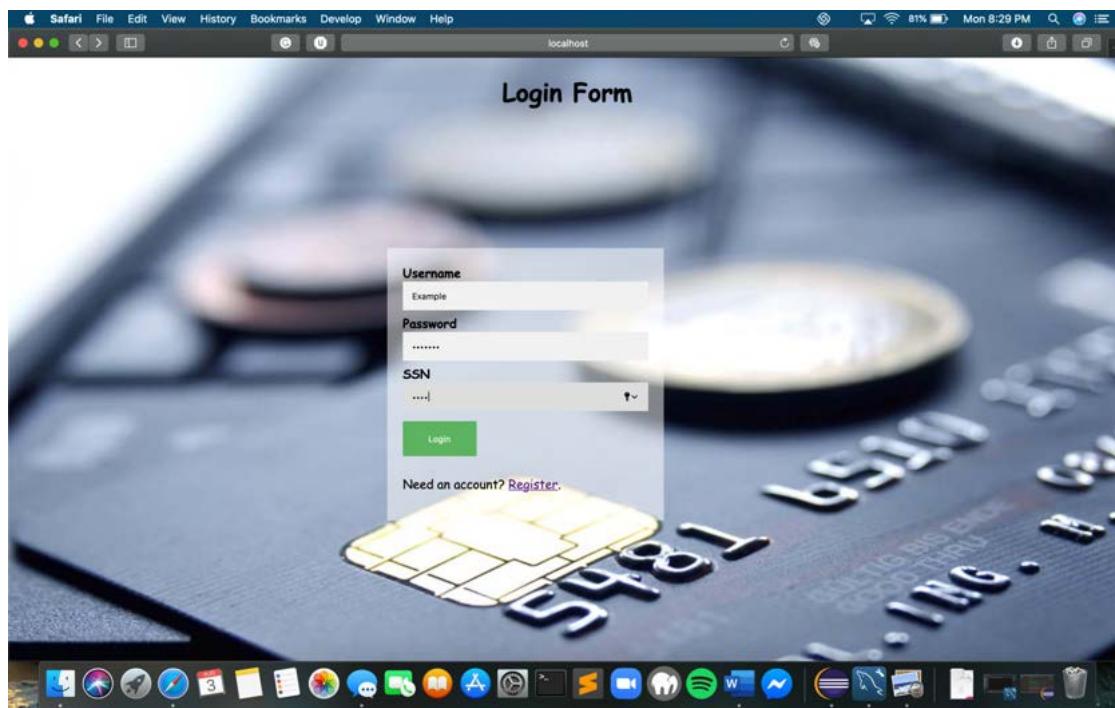
Table: USERS

Columns:

username	varchar(45)
password	varchar(45)
SSN	varchar(45) PK

Query Completed

## IMPLEMENTATION



Front-end for login form.

Any new user can access the Banking System. But to access the system, an user must register beforehand. As you can see in the SS, a user with username "example" is logging in.

Back-end for login form.

This code will check if there an user is registered in the database system. If not, it will prompt that it is not registered.

Part of the login form back-end

The screenshot shows the MySQL Workbench application running on a Mac OS X desktop. The main window displays a query editor with the following content:

```
1 • SELECT * FROM ABCbank.USERS;
```

The results pane shows a table with three columns: username, password, and SSN. The table is currently empty, indicated by three rows with 'NULL' values.

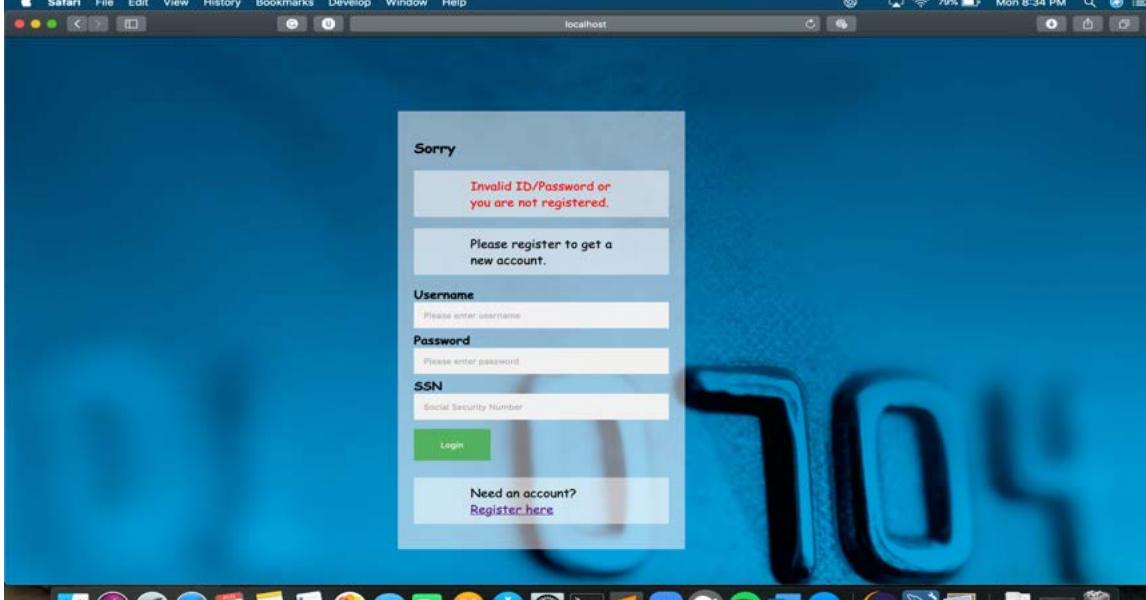
On the left, the Schemas tree shows the 'ABCbank' schema containing tables like 'chequeBookOrder', 'deletedAccount', 'deposits', 'hasBalance', 'Loan', 'Statements', 'Transfers', 'userinfo', and 'USERS'. It also lists views, stored procedures, functions, and two user accounts: 'Arren Baral' and 'Banking System'. A database named 'bankingsystem' is also visible.

At the bottom, the Action Output pane shows the execution of the query and its results:

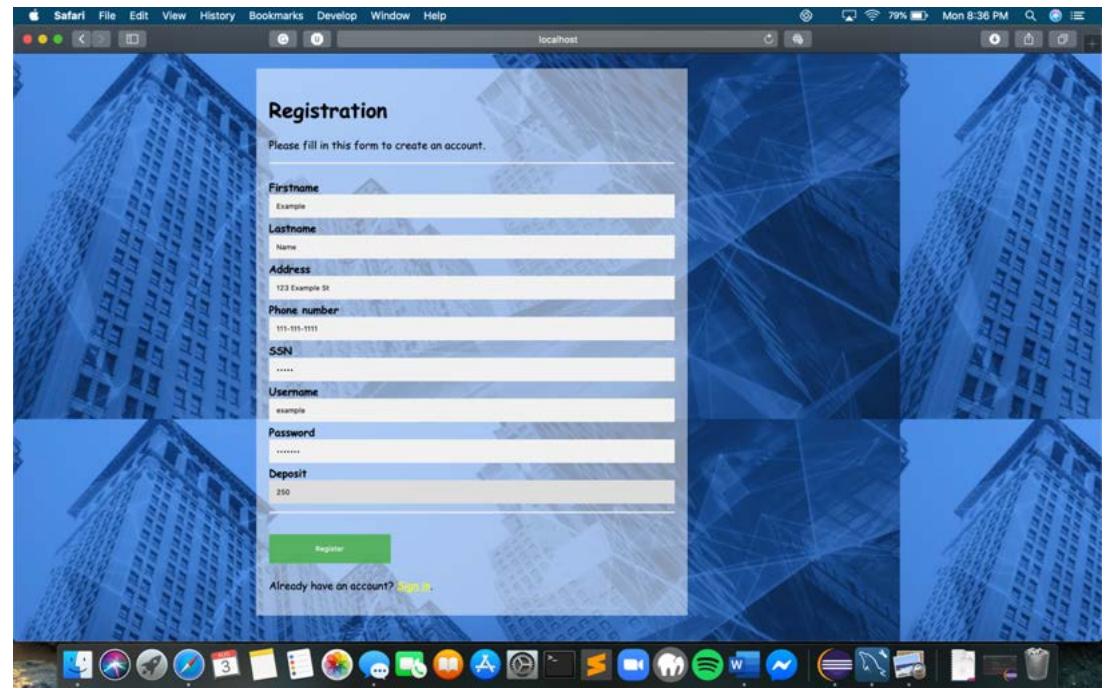
Action	Time	Response	Duration / Fetch Time
16	20:30:14	SELECT * FROM ABCbank.USERS LIMIT 0, 1000 0 row(s) returned	0.00057 sec / 0.0000...

A status bar at the bottom indicates "Query Completed". The system tray at the very bottom shows various application icons.

As we can see, there are no any users registered in the database. So it will prompt the not registered message.



Front-end for login form.  
So, there are no any registered user in the database system. It prompts if the username or password is mistaken or the user is not registered.



### Front-end for Registration

After, the user get the register prompt, user clicks [Register Here](#) link, it redirects to here, so the user can create an account to get an access to the system. Here, user provides his/her information and first deposit.

```

01 String fName = request.getParameter("name");
02 String address = request.getParameter("address");
03 String phone = request.getParameter("phone");
04 String ssn = request.getParameter("ssn");
05 String username = request.getParameter("username");
06 String password = request.getParameter("password");
07 String deposit = request.getParameter("deposit");
08
09 String url ="jdbc:mysql://localhost:3306/ABChankServer?timezone=UTC";
10 Class.forName("com.mysql.jdbc.Driver");
11 try {
12     Connection connection = DriverManager.getConnection(url,"root","");
13     PreparedStatement ps;
14     ps = connection.prepareStatement("Insert into userInfo(firstname, lastname, username, password, address, phone, amount,SSN) values(?,?,?,?,?,?,?,?)");
15
16     ps.setString(1,fName);
17     ps.setString(2,fName);
18     ps.setString(3,username);
19     ps.setString(4,password);
20     ps.setString(5,address);
21     ps.setString(6,phone);
22     ps.setString(7,deposit);
23     ps.setString(8,ssn);
24     ps.executeUpdate();
25
26
27     PreparedStatement psi;
28
29
30     psi = connection.prepareStatement("Insert into USERS(username, password, SSN) values(?, ?, ?)");
31
32     //ps.setNull(1, java.sql.Types.INTEGER);
33     //ps.setString(1,username);
34     ps.setString(2,password);
35     ps.setString(3,ssn);
36     ps.executeUpdate();
37
38
39     PreparedStatement ps2;
40
41
42     ps2 = connection.prepareStatement("Insert into hasBalance(ssn, balance) values(?,?)");
43
44     //ps.setNull(1, java.sql.Types.INTEGER);
45     ps2.setString(1,ssn);
46     ps2.setString(2,deposit);
47
48     ps2.executeUpdate();
49
50
51 }
52 catch (SQLException e) {
53     e.printStackTrace();
54 }
```

```

01 registration.jsp
02
03 <html>
04 <head>
05 <title>Registration</title>
06 </head>
07 <body>
08
09 <form action="registration.jsp" method="post">
10
11     <div style="text-align: center; margin-bottom: 10px;">
12         <input type="text" placeholder="Please enter username" name="username" id="username" required>
13     </div>
14     <div style="text-align: center; margin-bottom: 10px;">
15         <input type="password" name="password" id="password" required>
16     </div>
17     <div style="text-align: center; margin-bottom: 10px;">
18         <input type="password" placeholder="Please enter password" name="password" id="password" required>
19     </div>
20     <div style="text-align: center; margin-bottom: 10px;">
21         <input type="text" placeholder="Social Security Number" name="ssn" id="ssn" required>
22     </div>
23
24     <div style="text-align: center; margin-bottom: 10px;">
25         <input type="submit" class="loginbtn" value="Login">
26     </div>
27
28 </form>
29
30 </body>
31 </html>
```

### Back-end for registration

This code will insert all the information to the userInfo, and USERS tables.

### Back-end for registration Part of the registration

MySQLWorkbench

File Edit View Query Database Server Tools Scripting Help MySQL Workbench

Administration Schemas

ABCbank

Tables

chequeBookOrder deletedAccount deposits hasBalance Loan Statements Transfers userinfo

USERs

withdrawals Views Stored Procedures Functions Arren Baraf Banking System bankingSystem cx157a

Object Info Session

Table: USERs

Columns:

- username varchar(45)
- password varchar(45)
- SSN varchar(45) PK

userinfo

Result Grid Filter Rows Search Edit Export/Import

1 SELECT \* FROM ABCbank.userInfo;

	idnewUser	firstname	lastname	username	password	address	phone	amount	SSN
1	Example	Name	example	example	123 Example St	111-111-1111	250	12345	

Action Output Time Action Response Duration / Fetch Time

18 20:39:03 SELECT \* FROM ABCbank.userInfo LIMIT 0, 1000 1 row(s) returned 0.00024 sec / 0.000...

Query Completed

Tables for registration  
After registration, it successfully adds the user information in the userInfo.

MySQLWorkbench

File Edit View Query Database Server Tools Scripting Help MySQL Workbench

Administration Schemas

ABCbank

Tables

chequeBookOrder deletedAccount deposits hasBalance Loan Statements Transfers userinfo

USERs

withdrawals Views Stored Procedures Functions Arren Baraf Banking System bankingSystem cx157a

Object Info Session

Table: USERs

Columns:

- username varchar(45)
- password varchar(45)
- SSN varchar(45) PK

userinfo

Result Grid Filter Rows Search Edit Export/Import

1 SELECT \* FROM ABCbank.USERs;

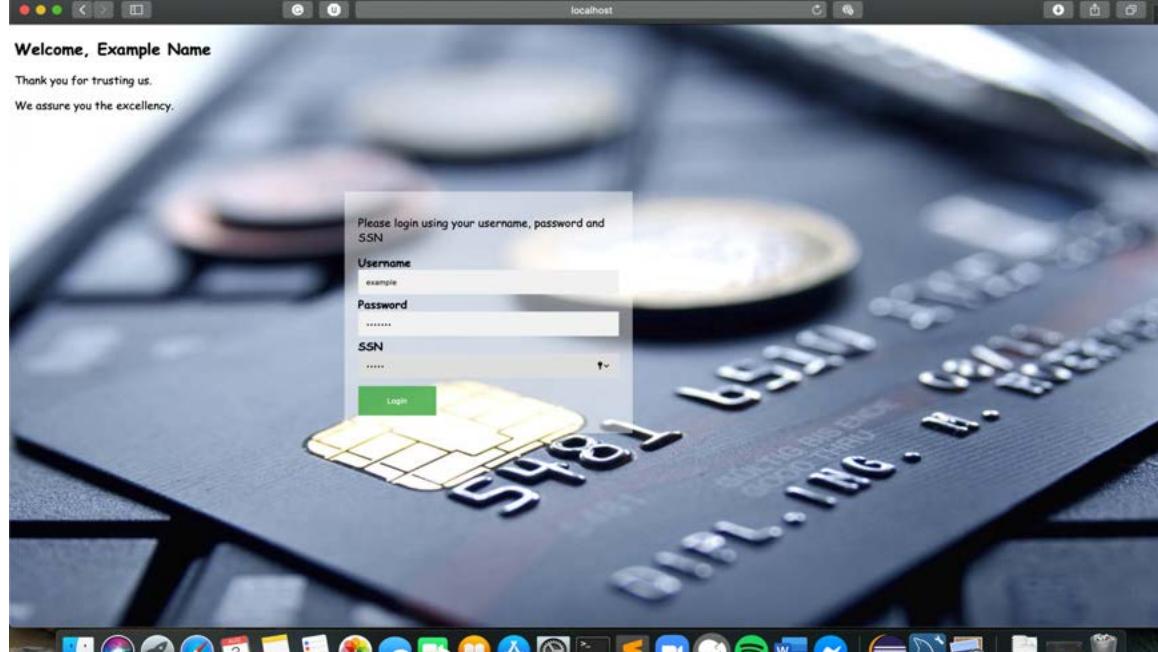
	username	password	SSN
1	example	example	12345

Action Output Time Action Response Duration / Fetch Time

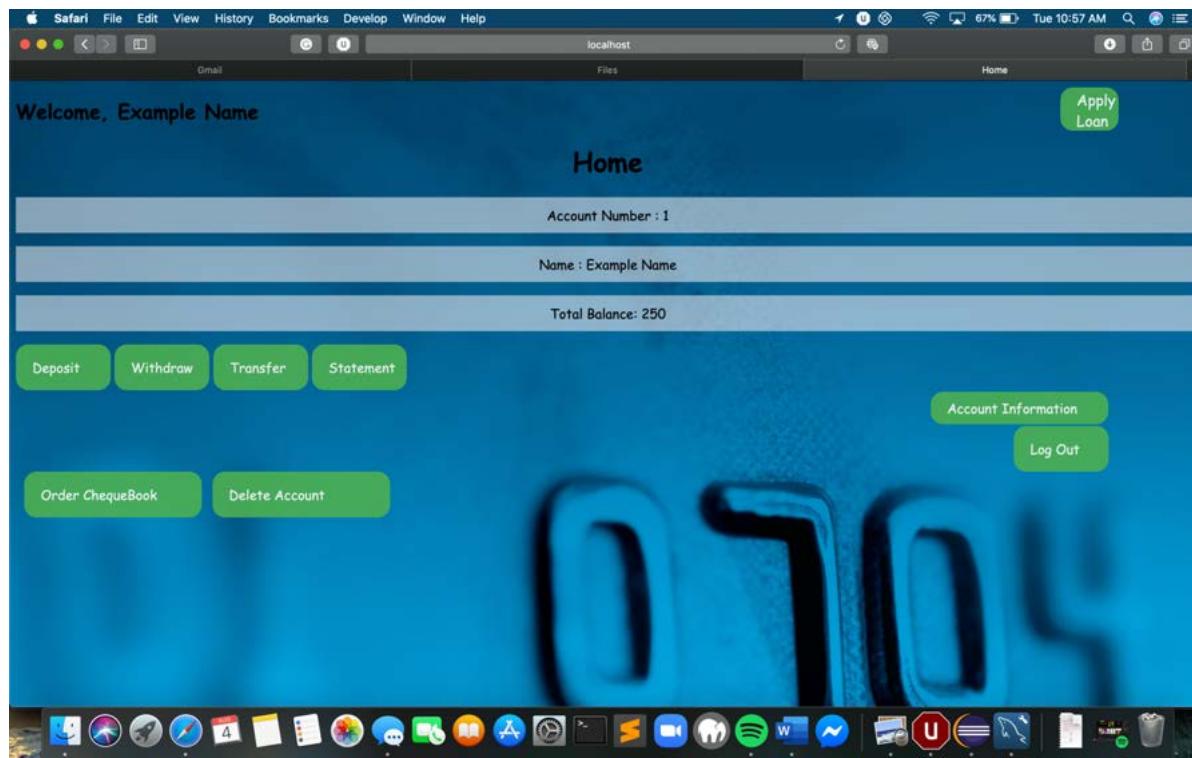
18 20:39:03 SELECT \* FROM ABCbank.userInfo LIMIT 0, 1000 1 row(s) returned 0.00024 sec / 0.000...

Query Completed

Tables for registration  
After registration, it successfully adds the username, password, and SSN in the USERs table.



Front-end after the successful registration  
User will be asked to use their recent username, password, and SSN to login.  
It also welcomes user with the message on the top.



### Front-end Homepage

Once logged in, it will redirect registered user to the Home Page of their account.

User sees the account, name, and total balance with other features like deposit, withdraw, transfer, statement, order cheque book, account information, loan apply, delete account, and log out. Total balance is 250 after the first deposit while registration.

```

Eclipse File Edit Source Refactor Navigate Project Run Window Help
Java Pro - BankingSys/WebContent/home.jsp - Eclipse IDE
Project Explorer
Banking System 2
BankingSys
JAX-WS Web Services
Java Resources
JavaScript Resources
Referenced Libraries
build
WebContent
ib
META-INF
WSB-INF
index.jsp
delete.html
delete.jsp
deposit.jsp
file.jsp
home.jsp
index.html
index.jsp
loan.html
loan.jsp
Registration.html
registration.jsp
RunMeFirst.jsp
statement.jsp
test.html
transfer.jsp
update_address.jsp
update_phone.jsp
withdraw.jsp
onlinesbanking
Servers
Test
Testing
home.jsp
343
344
345
346
347 String ssn=request.getParameter("ssn");
348 String url="jdbc:mysql://localhost:3306/ABCbank";
349 Connection connection = DriverManager.getConnection(url,"root","root");
350 Class.forName("com.mysql.jdbc.Driver");
351
352 Statement querySql = "SELECT * FROM userinfo WHERE SSN="+ssn+"";
353 Statement st=connection.createStatement();
354 ResultSet rs1=st.executeQuery(querySql);
355 while(rs1.next()){
356     String accountno = rs1.getString("idnewUser");
357     String FN = rs1.getString("firstname");
358     String LN = rs1.getString("lastname");
359     String address = rs1.getString("address");
360     String phone = rs1.getString("phone");
361     String deposit = rs1.getString("amount");
362
363
364     <h2>Welcome, <%=FN%> <%=LN%>
365     </h2>
366     <h1 align="center">Home</h1>
367
368     <p align="center">Account Number : <%=accountNo%></p>
369     <p align="center">Name : <%=FN%> <%=LN%> </p>
370
371
372
373 }
374
375 String querySql1 = "SELECT balance FROM hasBalance WHERE SSN="+ssn+"";
376 Statement rs4=connection.createStatement();
377 ResultSet rs4a=rs4.executeQuery(querySql1);
378 while(rs4.next()){
379     String balance = rs4.getString("balance");
380
381
382     <p align="center">Total Balance: <%=balance%></p>
383 }
384
385
386
387

```

### Back-end Homepage

Its selects account\_no, firstname, and last\_name from userInfo. And its selects balance from the hasBalance table.

MySQL Workbench Schemas

Administration Schemas Query 45 USERS userinfo deposits Transfers Statements chequeBookOrder userinfo userinfo > Context Help Snippets

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid Filter Rows Search Edit Export/Import

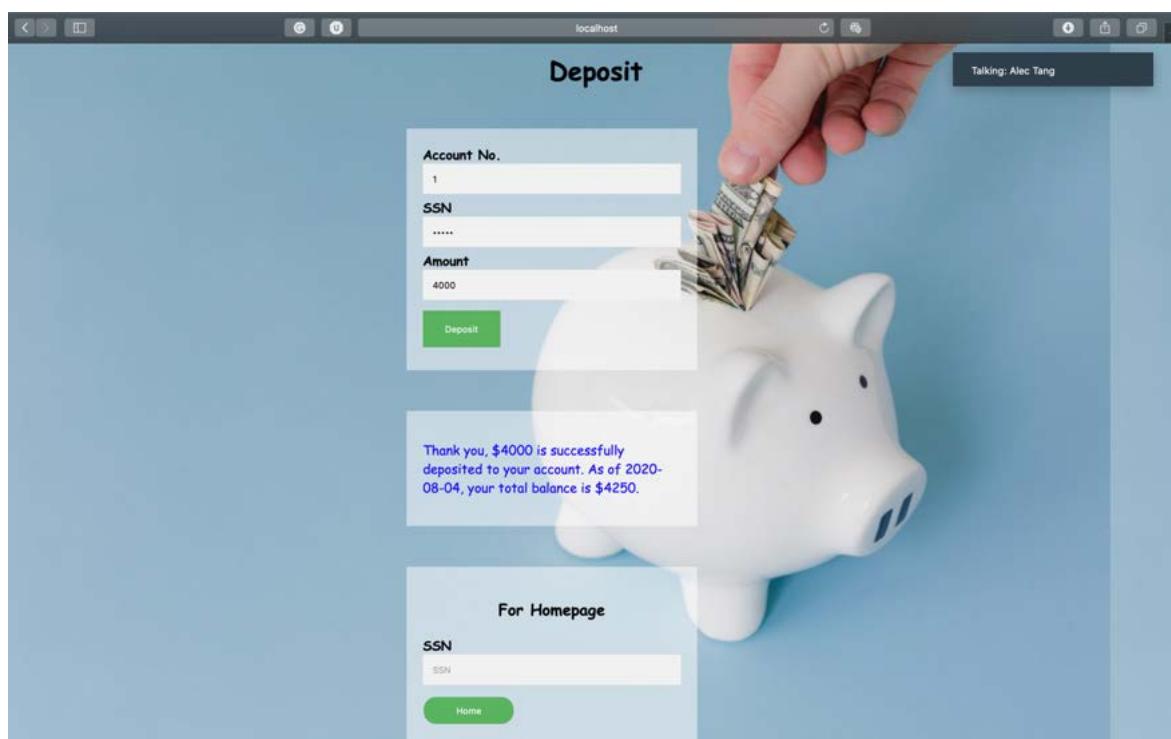
Table: hasBalance

Columns: SSN int PK, balance int

Action Output Time Action Response Duration / Fetch Time

Query Completed

When an user registers, they deposit certain amount that gets stored on the table hasBalance.



From homepage, if user chooses Deposit button, it will redirect to deposit page where user puts account number, ssn and amount. After the deposit, it gives the message that the amount has been deposited and the total amount.

```

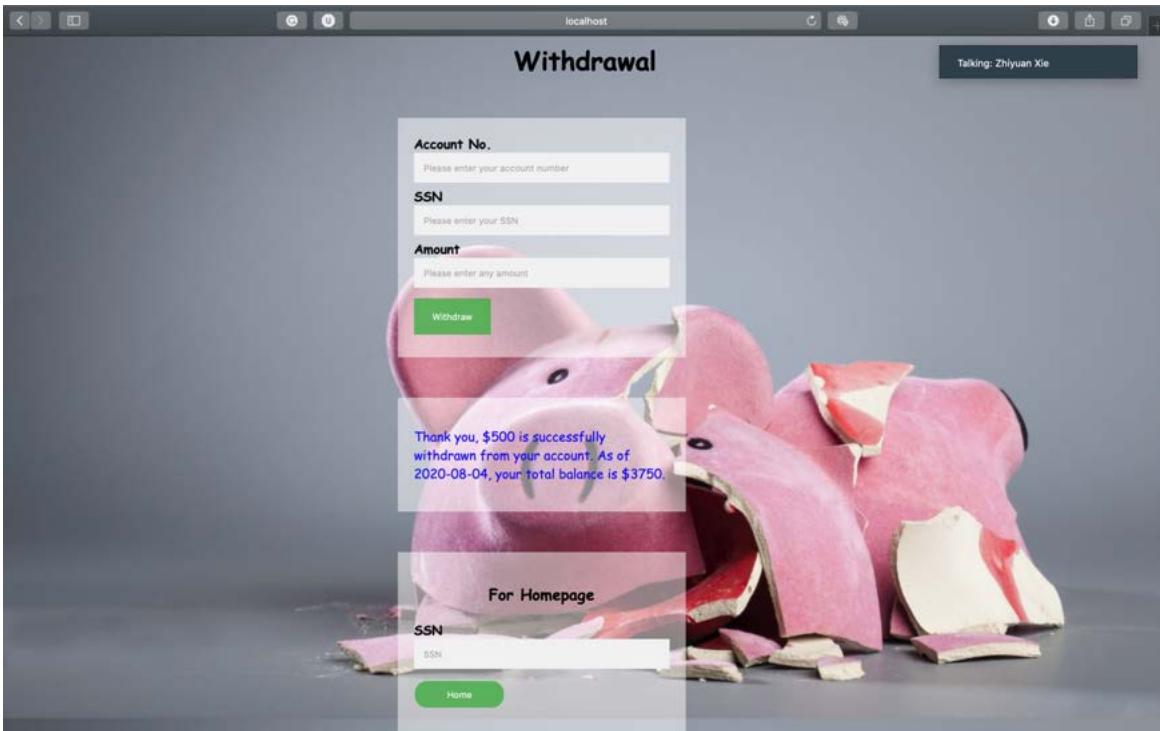
162 <button type="submit" class="loginbtn">Deposit</button>
163 
164 
165 </div></div>
166 
167 <%@ page import="java.sql.*" %>
168 <%@ page import="javax.sql.DataSource" %>
169 
170 
171 <%
172 String account = request.getParameter("account");
173 String ssn = request.getParameter("ssn");
174 String amount = request.getParameter("amount");
175 
176 
177 String url = "jdbc:mysql://localhost:3386/ABCbank?serverTimezone=UTC";
178 Class.forName("com.mysql.jdbc.Driver");
179 try {
180     Connection connection = DriverManager.getConnection(url, "root", "root");
181     java.util.Date now = new java.util.Date();
182     java.sql.Date sqlDate = new java.sql.Date(now.getTime());
183     String insertSql = "INSERT INTO deposits (account_id, amount, DATE) "
184     + "VALUES (?, ?, ?)";
185     Statement statement = connection.createStatement();
186     statement.executeUpdate(insertSql);
187     Statement statement2 = connection.createStatement();
188     statement.execute(insertSql);
189 
190 
191     String updateSql = "UPDATE hasBalance SET balance=balance+? WHERE SSN=?";
192     Statement statement3 = connection.createStatement();
193     statement3.executeUpdate(updateSql);
194 }
195 
196 
197 String querySql = "SELECT * FROM hasBalance WHERE SSN=?";
198 Statement statement4 = connection.createStatement();
199 ResultSet rs3 = statement.executeQuery(querySql);
200 while(rs3.next())
201 {
202     String balance = rs3.getString("balance");
203 
204     <div class="container">
205         <div style="color:BLUE">Thank you, $<%=amount%> is successfully deposited to your account.
206         As of <%=sqlDate%>, your total balance is $<%=balance%>. </div>
207     </div>
208 }
209 
210 
211 </div>
212 </div>
213 </div>
214 </div>
215 </div>
216 </div>
217 </div>
218 
```

## Back-end program

Once the data is submitted, this code will help to store informations on the table deposits table and update hasBalance.

account_id	amount	DATE
1	4000	2020-08-04

As we can see, the deposit record has been recorded on the table.



From homepage, if user chooses Withdraw button, it will redirect to withdraw page where user puts account number, ssn and amount. After the withdrawal, it gives the message that the amount has been withdrawn and the total amount.

## Back-end program

Once the data is submitted, this code will help to store informations on the table withdrawal table and update hasBalance.

MySQLWorkbench

File Edit View Query Database Server Tools Scripting Help

MySQL Workbench

Talking: Zhiyuan Xie

Administration Schemas

SCHEMAS

ABCbank

Tables

- chequeBookOrder
- deletedAccount
- deposits
- hasBalance
- Loan
- Statements
- Transfers
- userinfo
- USERS
- withdrawals

Views

Stored Procedures

Functions

Arren Baral

Banking System

bankingsystem

cs157a

Object Info Session

Table: hasBalance

Columns:

SSN	int PK
balance	int

account\_id amount DATE

HULL	HULL	2020-08-04
1	500	2020-08-04

withdrawals 1

Action Output

Time	Action	Response	Duration / Fetch Time
10 13:30:59	SELECT * FROM ABCbank.withdrawals LIMIT 0, 1000	4 row(s) returned	0.00069 sec / 0.000...

Query Completed

As we can see, the withdraw record has been recorded on the table.

localhost

Talking: Zhiyuan Xie

# Transfer

Account No.

Please enter your account number

SSN

Please enter your SSN

Receiver's Name

Please enter receiver's name

Receiver's Phone Number

Please enter any receiver's phone number

Amount

Please enter any amount

Transfer

Thank you, \$123 is successfully transferred to Someone Lastname from your account. As of 2020-08-04, your total balance is \$3627.

For Homepage

SSN

From homepage, if user chooses Transfer button, it will redirect to transfer page where user puts account number, ssn, receiver's name, phone, and amount. After the transfer, it gives the message that the amount has been transferred to the receiver and the total amount.

```

183
184
185 String account = request.getParameter("account");
186 String name = request.getParameter("name");
187 String phone = request.getParameter("phone");
188 String amount = request.getParameter("amount");
189 String ssn = request.getParameter("ssn");
190
191 String url = "jdbc:mysql://localhost:3306/ABCbank?serverTimezone=UTC";
192 Class.forName("com.mysql.jdbc.Driver");
193
194 java.util.Date now = new java.util.Date();
195 java.sql.Date sqldate = new java.sql.Date(now.getTime());
196 Connection connection = DriverManager.getConnection(url, "root", "root");
197
198 PreparedStatement ps;
199
200 ps = connection.prepareStatement("Insert into Transfers(account_id, recipient_name, recipient_phone, amount, DATE) values(?, ?, ?, ?, ?)");
201
202
203 ps.setString(1, account);
204 ps.setString(2, name);
205 ps.setString(3, phone);
206 ps.setString(4, amount);
207 ps.setDate(5, sqldate);
208
209 ps.executeUpdate();
210
211
212
213 String updateSql = "UPDATE hasBalance SET balance=balance-? WHERE SSN=?;";  

214 Statement statement2 = connection.createStatement();
215 statement2.execute(updateSql);
216
217 String querySql2 = "SELECT * FROM hasBalance WHERE SSN=?";  

218 Statement st2 = connection.createStatement();
219 ResultSet rs3 = st2.executeQuery(querySql2);
220
221 while(rs3.next())
222 {
223
224     String balance = rs3.getString("balance");
225     <br>
226     <div class="container">
227         <p style="color:BLUE">Thank you, $<?> is successfully transferred to <?> from your account.  

228         As of <?>, your total balance is $<?>. </p>
229     </div>
230 }
231
232
233
234
235 } catch (SQLException e) {
236
237     e.printStackTrace();
238 }
239 
```

Line 1, Column 1      Spaces: 2      Java Server Page (JSP)

## Back-end program

Once the data is submitted, this code will help to store informations on the table transfer table and update hasBalance.

MySQL Workbench

Administration   Schemas   Query 45   USERS   userinfo   deposits   Transfers   Statements   chequeBookOrder   userinfo   userinfo   >

Limit to 1000 rows

Talking: Mike Wu

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid   Filter Rows: Search   Export:

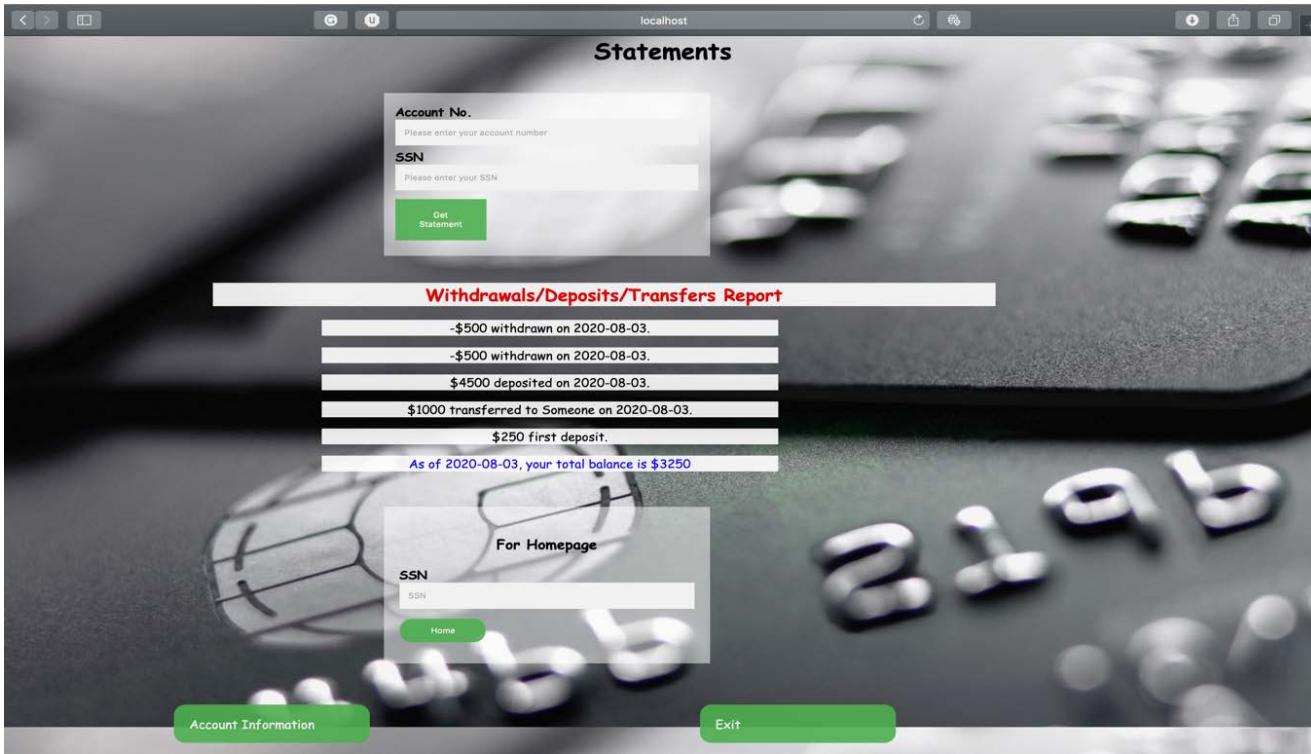
account_id	recipient_name	recipient_ph...	amount	DATE
1	Someone Lastname	333-333-3333	123	2020-08-04

Action Output   Time   Action   Response   Duration / Fetch Time

11 13:36:23   SELECT \* FROM ABCbank.Transfers LIMIT 0, 1000   3 row(s) returned   0.00037 sec / 0.0000...

Query Completed

As we can see, the Transfer record has been recorded on the table.



From homepage, if user chooses Statement button, it will redirect to statement page where user puts account number, and ssn. Once the data is received, it gives the records of recent transactions.

## Back-end program

Once the data is submitted, this code will help to store informations on the table statements.

Part of back-end statement feature.

```
statement.jsp
205
206
207 }
208
209 String querySql = "SELECT * FROM deposits WHERE account_id='account'";
210 Statement st1 con.createStatement();
211 ResultSet rs2 st1.executeQuery(querySql1);
212
213 while(rs2.next()){
214     String amount = rs2.getString("amount");
215     String date = rs2.getString("DATE");
216
217     <div class="container">
218         <align="center">$<amount> deposited on <date>. </></div>
219     </div>
220 }
221
222 String querySql4 = "SELECT * FROM Transfers WHERE account_id='account'";
223 Statement st4 con.createStatement();
224 ResultSet rs4 st4.executeQuery(querySql4);
225
226 while(rs4.next()){
227     String transfer = rs4.getString("amount");
228     String date = rs4.getString("DATE");
229     String receiver = rs4.getString("recipient_name");
230
231     <div class="container">
232         <align="center">$<transfer> transferred to <receiver> on <date>. </></div>
233     </div>
234 }
235
236 String query = "SELECT * FROM userInfo WHERE SSN='ssn'";
237 Statement state con.createStatement();
238 ResultSet result state.executeQuery(query);
239
240 while(result.next()){
241     String amount = result.getString("amount");
242     <div class="container">
243         <align="center">$<amount> first deposit. </></div>
244     </div>
245 }
246
247 java.util.Date now = new java.util.Date();
248 java.sql.Date sqlDate = new java.sql.Date(now.getTime());
249
250 String querySql2 = "SELECT * FROM hasBalance WHERE SSN='ssn'";
251 Statement st2 con.createStatement();
252 ResultSet rs3 st2.executeQuery(querySql2);
253
254 while(rs3.next())
255 {
256     String amount = rs3.getString("balance");
257     <div class="container">
258         <align="center" style="color:blue"> As of <sqlDate>, your total balance is $<amount> </></div>
259     </div>
260 }
261
262 ///////////////////////////////////////////////////////////////////
263 PreparedStatement ps;
264 ps = con.prepareStatement("INSERT INTO Statements(account_id,orderDate) VALUES" +
265                         "(?,?)");
266 ps.setString(1,account);
267 ps.setDate(2,sqlDate);
268 ps.executeUpdate();
269
270 ///////////////////////////////////////////////////////////////////
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
```

Line 238, Column 2 Tab Size: 4 Java Server Page (JSP)

MySQL Workbench

Administration   Schemas   Query 45   USERS   userinfo   deposits   Transfers   Statements   chequeBookOrder   userinfo   userinfo   >   Context Help   Snippets

Server

Talking: Min-Yuan Lee

Automatic context help is disabled. Use the toolbar to manually get help for the current care position or to toggle automatic help.

ABCbank

Tables

- chequeBookOrder
- deletedAccount
- deposits
- hasBalance
- Loan
- Statements
- Transfers
- userinfo
- USERS
- withdrawals

Views

Stored Procedures

Functions

Arrenn Baral

Banking System

bankingsystem

cs157a

Object Info   Session

Table: hasBalance

Columns:

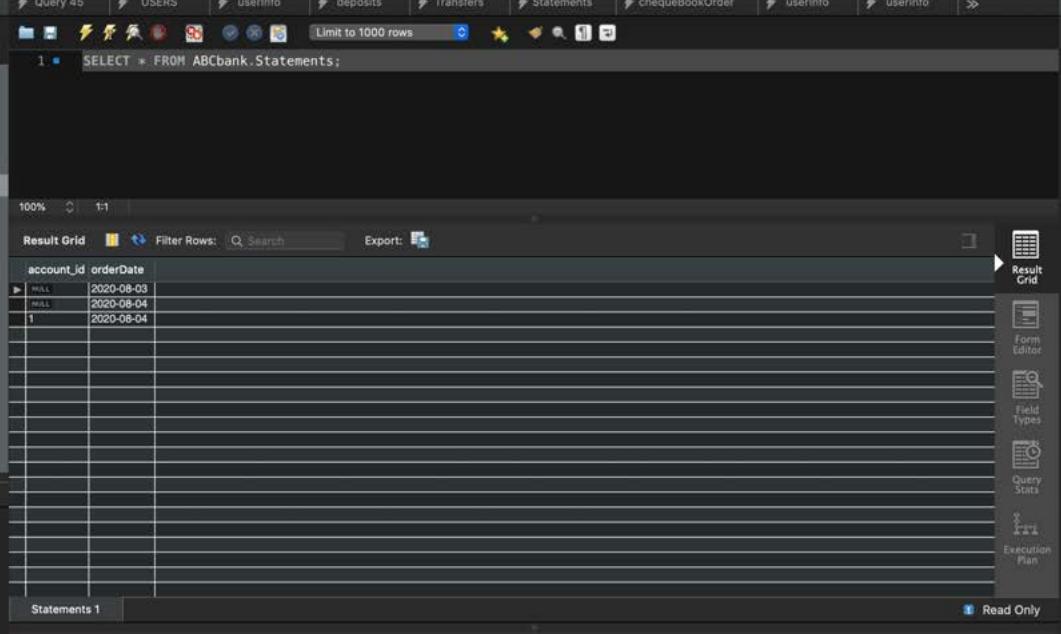
SSN	int PK
balance	int

Statements 1

Action Output

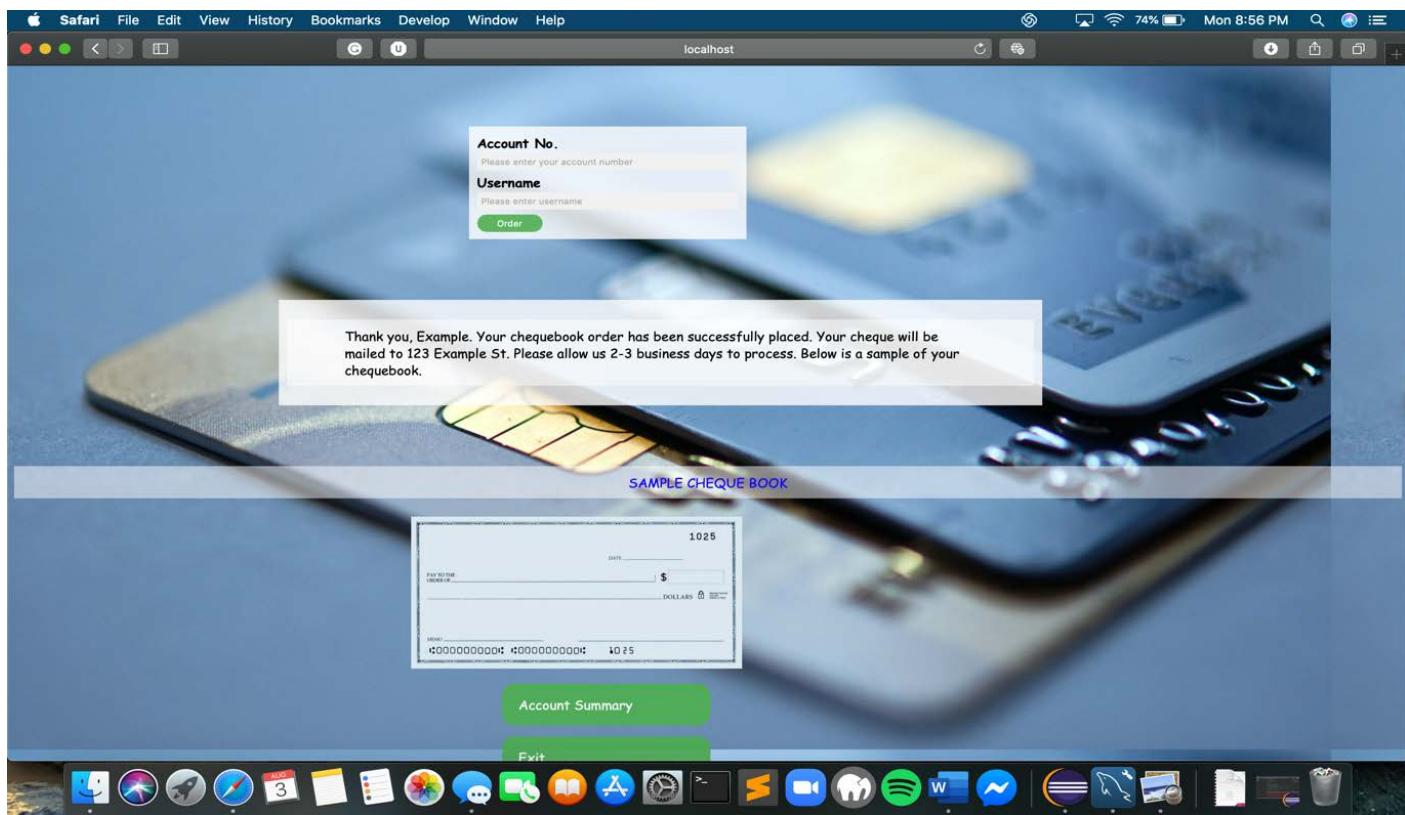
Time	Action	Response	Duration / Fetch Time
13:47:06	SELECT * FROM ABCbank.Statements LIMIT 0, 1000	3 row(s) returned	0.00036 sec / 0.000...

Query Completed



account_id	orderDate
NULL	2020-08-03
NULL	2020-08-04
1	2020-08-04

As we can see, the statement record has been recorded on the table.



From homepage, if user chooses Order Checkbook button, it will redirect to checkboxorder page where user puts account number, and ssn. Once the data is received, it gives the message that the order is placed and it will be mailed the stored address.

```

Apple Safari File Edit View History Bookmarks Develop Window Help
localhost
Account No.
Please enter your account number
Username
Please enter username
Order

Thank you, Example. Your chequebook order has been successfully placed. Your cheque will be mailed to 123 Example St. Please allow us 2-3 business days to process. Below is a sample of your chequebook.

SAMPLE CHEQUE BOOK

Account Summary
Exit

Sublime Text File Edit Selection Find View Goto Tools Project Window Help
cheque.jsp UNREGISTERED
127
128
129
130
131 String account = request.getParameter("account");
132 String name = request.getParameter("name");
133 String url ="jdbc:mysql://localhost:3306/ABCbank?serverTimezone=UTC";
134 Class.forName("com.mysql.jdbc.Driver");
135
136 try {
137
138 Connection connection = DriverManager.getConnection(url,"root","root");
139 java.util.Date now = new java.util.Date();
140 java.sql.Date sqdate = new java.sql.Date(now.getTime());
141
142
143
144 PreparedStatement ps;
145 ps = connection.prepareStatement("INSERT INTO chequeBookOrder(account_id,username, orderDate) VALUES (?, ?, ?)");
146 ps.setString(1,account);
147 ps.setString(2,name);
148 ps.setDate(3,sqdate);
149 ps.executeUpdate();
150
151 String querySql = "SELECT * FROM userInfo WHERE idnewUser=? account ";
152 Statement st = connection.createStatement();
153 ResultSet rs1 = st.executeQuery(querySql);
154 while(rs1.next()){
155     String FN = rs1.getString("firstname");
156     String address = rs1.getString("address");
157 }
158
159 <div class="mcontainer">
160
161 <!--> Thank you, <!-->. Your chequebook order has been successfully placed. Your cheque will be mailed to <!-->address <!-->.
162 Please allow us 2-3 business days to process.
163 Below is a sample of your chequebook.
164 </div>
165 <div align="center" style="color:blue">SAMPLE CHEQUE BOOK</div>
166 
167
168
169
170
171
172
173
174
175
176
177
178
179 catch (SQLException e) {
180     e.printStackTrace();
181 }
182
183
184
185
186
187 <a href="file.jsp" class="admin">Account Summary</a>
188 <a href="index.html" class="admin">Exit</a>
Line 1, Column 1
Spaces: 2 Java Server Page (JSP)

```

## Back-end program

Once the data is submitted, this code will help to store informations on the table chequeBookOrder.

MySQL Workbench

File Edit View Query Database Server Tools You are viewing Tracy Ho's screen View Options 96% Tue 2:00 PM MySQL Workbench

Administration Schemas Query 45 chequeBookOrder

Limit to 1000 rows

Automatic context help is disabled. Use the toolbar to manually get help for the current care position or to toggle automatic help.

**SCHEMAS**

- ABCbank
  - Tables
    - chequeBookOrder
    - deletedAccount
    - deposits
    - hasBalance
    - Loan
    - Statements
    - Transfers
    - userinfo
    - USERS
    - withdrawals
  - Views
  - Stored Procedures
  - Functions
- Arren Baral
- Banking System
- bankingsystem
- cs157a

Object Info Session

Table: hasBalance

Columns:

SSN	int PK
balance	int

chequeBookOrder 1

Action Output

Time	Action	Response	Duration / Fetch Time
19 14:00:04	SELECT * FROM ABCbank.chequeBookOrder LIMIT 0, 1000	1 row(s) returned	0.00033 sec / 0.000...

Read Only

Query Completed

As we can see, the check book order record has been recorded on the table.

localhost Account Information

SSN  
Please enter your SSN Go

Account Number : 1  
First Name : Example  
Last Name : Name  
Address : 123 Example St  
Phone Number : 111-111-1111

Homepage  
SSN  
Home

Update Address  
Update Phone

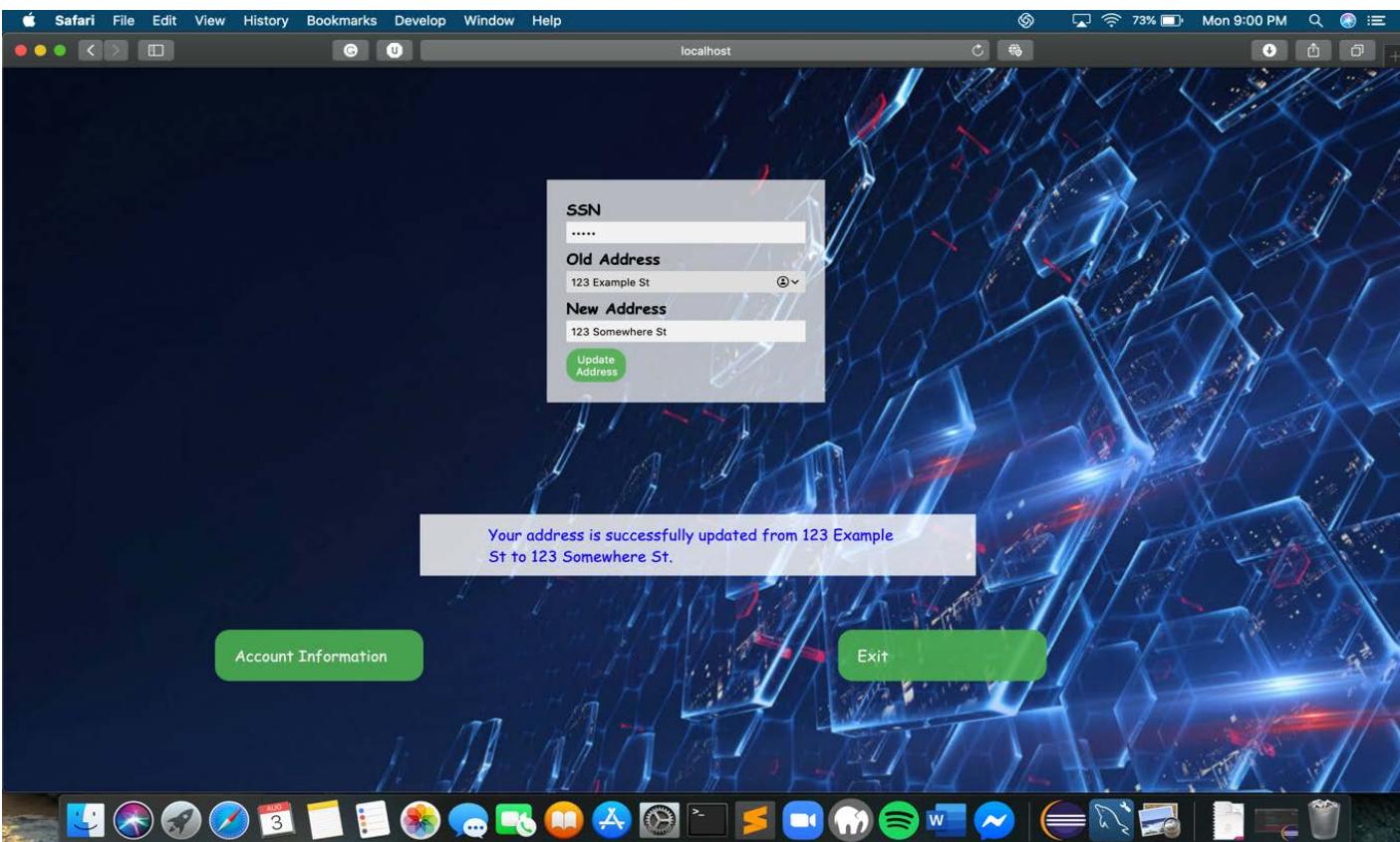
From homepage, if user chooses Account Information button, it will redirect to account information page where user puts ssn. Once the data is received, it displays the account information of the user.

```
Sublime Text File Edit Selection Find View Goto Tools Project Window Help file.jsp 100% Mon 11:27 PM UNREGISTERED  
file.jsp  
351  
352  
353  
354  
355 <html action="home.jsp" method="post">  
356 <div class="container">  
357 <div align="center">Homepage</div>  
358 <form for="ssn"><br/>  
359 <input type="password" placeholder="SSN" name="ssn1" id="ssn1" required>  
360  
361 <button type="submit" class="btn">Home</button>  
362 </form>  
363  
364  
365 <%@ page import="java.sql.*" %>  
366 <%@ page import="javax.sql.*" %>  
367 <%@ page import="java.util.*" %>  
368  
369 <%  
370 String ssn1 = request.getParameter("ssn1");  
371 Statement st1 = connection.createStatement();  
372 String query1 = "SELECT * FROM userinfo WHERE SSN='"+ssn1+"'";  
373 Statement st1 = connection.createStatement();  
374 ResultSet rs1 = st1.executeQuery(query1);  
375 while(rs1.next()) {  
376     String accountno = rs1.getString("idnumber");  
377     String FN = rs1.getString("firstname");  
378     String LN = rs1.getString("lastname");  
379     String address = rs1.getString("address");  
380     String phone = rs1.getString("phone");  
381     String deposit = rs1.getString("amount");  
382     <br>  
383     <div align="center">Account Number : <%=accountNo%</div>  
384     <div align="center">Name : <%=FN%> <%=LN%>  
385     <br>  
386 }  
387  
388  
389 <%  
390 String query1 = "SELECT balance FROM hasBalance WHERE SSN='"+ssn1+"'";  
391 Statement st2 = connection.createStatement();  
392 ResultSet rs2 = st2.executeQuery(query1);  
393 while(rs2.next()) {  
394     String balance = rs2.getString("balance");  
395     <br>  
396     <div align="center">Total Balance: <%=balance%</div>  
397 }  
398  
399  
400  
401  
402  
403  
404 <a href="update_address.jsp" class="depositbtn">Update Address</a>  
405 <a href="update_phone.jsp" class="depositbtn">Edit Update Phone</a>  
406 <a href="index.html" class="depositbtn">Exit</a>  
407  
408  
409 </body>  
410 </html>
```

Line 1, Column 1      Spaces: 2      Java Server Page (JSP)

## Back-end program

Once the data is submitted, this code will help to display information of user with SSN given.



## Front end for update address

If user wishes to update their address, then they can click on update address button from account information. They will be asked to put ssn, old address and new address. Message will be prompted to the user about the changed address.

```

1 update_address.jsp
2 <%@page import="java.sql.*" %>
3 <%@page import="javax.sql.*" %>
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174

```

Line 1, Column 1      Spaces: 2      Java Server Page (JSP)

## Back-end program

Once the data is submitted, this code will help to display information of user with SSN given with updated address.

MySQL Workbench

Administration   Schemas   Query 45   deletedAccount   hasBalance   userinfo

Limit to 1000 rows

Result Grid   Filter Rows: Search: Edit: Export/Import:

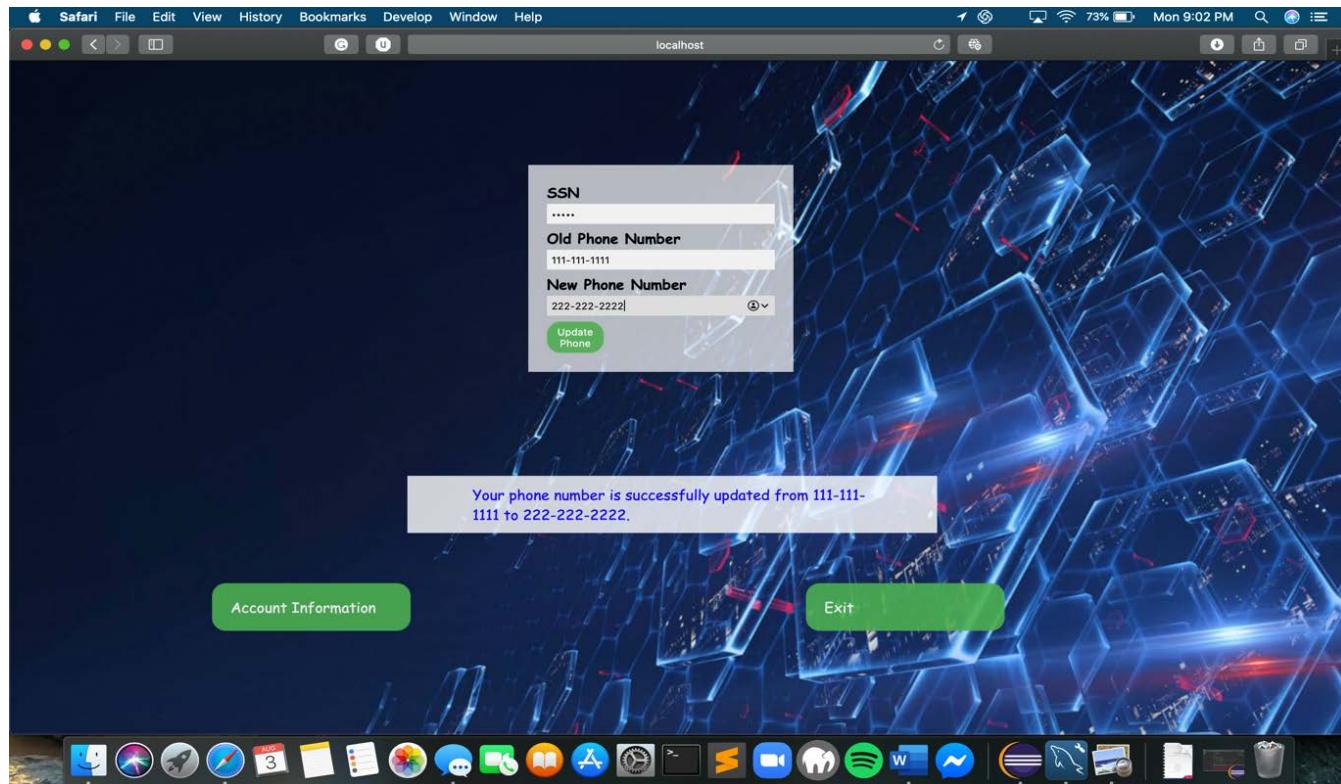
	id	newUser	firstname	lastname	username	password	address	phone	amount	SSN
1	WILL	Example	Name	MAIL	MAIL	MAIL	123 Somewhere St	111-111-1111	250	12345

Action Output:

Time	Action	Response	Duration / Fetch Time
27 15:59:53	SELECT * FROM ABCBank.userInfo LIMIT 0, 1000	1 row(s) returned	0.0011 sec / 0.00001...

Query Completed

As we can see, the updated address record has been updated on the table.



### Front end for update phone

If user wishes to update their phone, then they can click on update phone button from account information. They will be asked to put ssn, old phone and new phone. Message will be prompted to the user about the changed phone.

idnewUser	firstname	lastname	username	password	address	phone	amount	SSN
1	Example	Name	example	example	123 Somewhere St	222-222-2222	250	12345

As we can see, the updated phone record has been updated on the table.

Sublime Text File Edit Selection Find View Goto Tools Project Window Help update\_phone.jsp UNREGISTERED

```

328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380

```

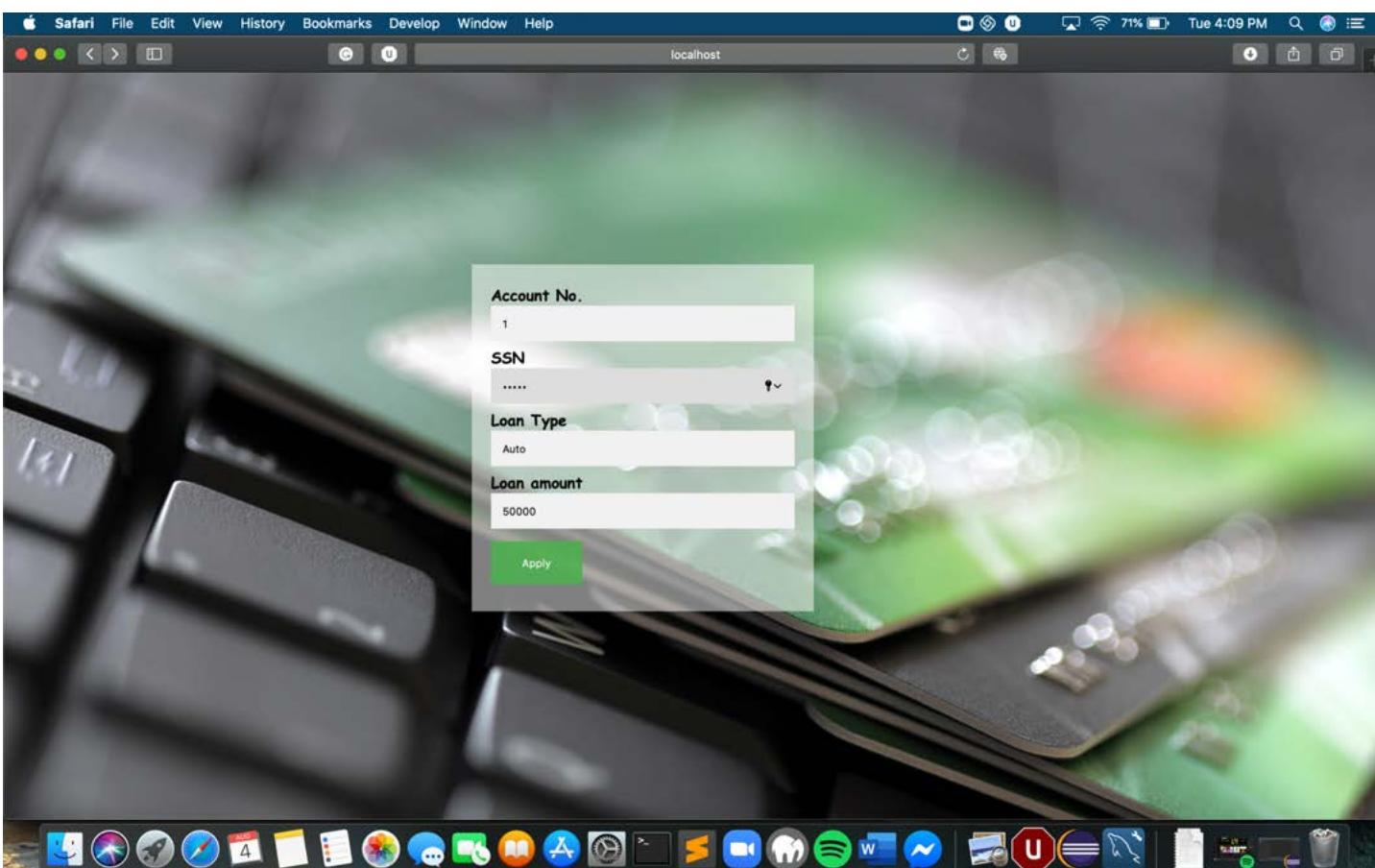
String ssn = request.getParameter("ssn");  
String oldphone = request.getParameter("phone");  
String newphone = request.getParameter("newphone");  
  
String url ="jdbc:mysql://localhost:3306/ABCbank?serverTimezone=UTC";  
Class.forName("com.mysql.jdbc.Driver");  
try {  
Connection connection = DriverManager.getConnection(url,"root","root");  
PreparedStatement ps;  
//  
ps = connection.prepareStatement("UPDATE userInfo SET phone=? WHERE SSN=?");  
//  
//ps.setString(1,newphone);  
ps.setString(1,ssn);  
  
ps.executeUpdate();  
  
String querySql = "SELECT phone FROM userInfo WHERE SSN='"+ssn+"'";  
Statement st1 = connection.createStatement();  
ResultSet rs1 = st1.executeQuery(querySql);  
rs1.next();  
  
String phone = rs1.getString("phone");  
  
//  
//<div class="mcontainer">  
//<div style="color:#blue"> Your phone number is successfully updated from <%oldphone %> to <%phone %>.  
//</div>  
//</div>  
  
}
catch (SQLException e) {
e.printStackTrace();
}
  
[Account Information](file.jsp)>  
[Exit](index.html)>

Line 1, Column 1 Spaces: 2 Java Server Page (JSP)

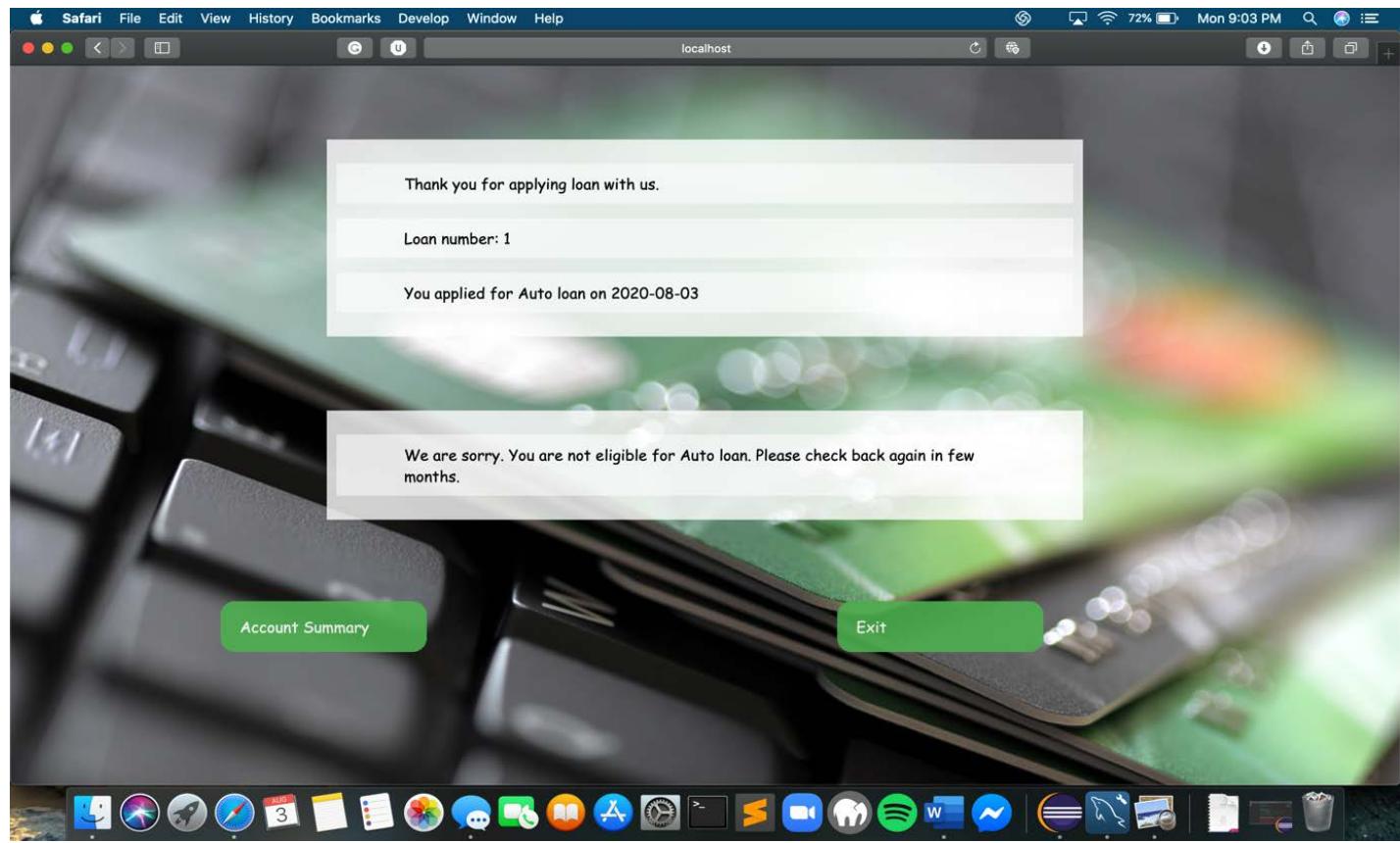


## Back-end program

Once the data is submitted, this code will help to display information of user with SSN given with updated phone.



From homepage, if user chooses Apply Loan button from the top right, it will redirect to loan page where user puts account number, and ssn.



Once they apply for loan, it will give the instance result. In the back end program, I have the requirement that if a user has more than 4000 balance, they get accepted for the loan.

A screenshot of MySQL Workbench. The left sidebar shows the schema "ABCbank" with tables like chequeBookOrder, deletedAccount, deposits, hasBalance, Loan, Statements, Transfers, userinfo, and USERS. The central pane shows a query results grid for a SELECT statement: "SELECT \* FROM ABCbank.Loan;". The result grid contains one row of data: loan\_number (1), account\_id (HULL), loantype (Auto), loan\_amount (50000), and DATE (2020-08-03). The right sidebar includes a note about context help being disabled and links to Form Editor, Field Types, Query Stats, and Execution Plan.

loan_number	account_id	loantype	loan_amount	DATE
1	HULL	HULL	50000	2020-08-03

As we can see, the loan record has been recorded on the table.

```

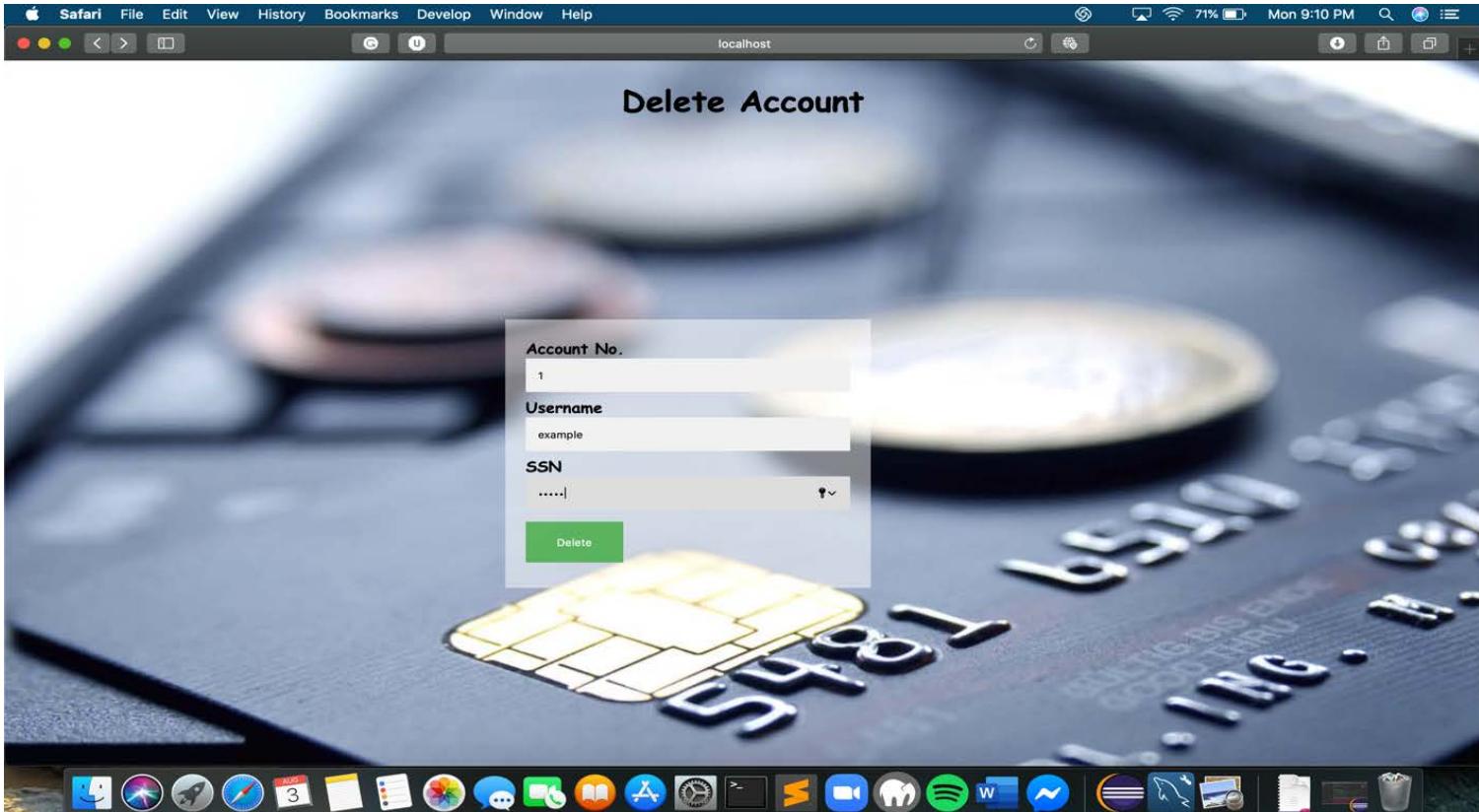
126 String url="jdbc:mysql://localhost:3306/ABCbank?serverTimezone=UTC";
127 Class.forName("com.mysql.jdbc.Driver");
128 try {
129     java.util.Date now = new java.util.Date();
130     java.sql.Date sqDate = new java.sql.Date(now.getTime());
131     Connection connection = DriverManager.getConnection(url,"root","root");
132 
133     PreparedStatement ps;
134     ps = connection.prepareStatement("INSERT INTO Loan(account_id, loan_type, loan_amount, DATE) VALUES" +
135         "(?, ?, ?, ?)");
136     ps.setString(1,account);
137     ps.setString(2,loanType);
138     ps.setDouble(3,loanAmount);
139     ps.setDate(4, sqDate);
140     ps.executeUpdate();
141     String querySql = "SELECT * FROM Loan WHERE account_id=?";
142     Statement st = connection.createStatement();
143     ResultSet rs = st.executeQuery(querySql);
144     while(rs.next())
145     {
146         // rs.getString(1); //or rs.getString("column name");
147         String loanNo = rs.getString("loan_number");
148         String loanType = rs.getString("loan_type");
149         String date = rs.getString("DATE");
150 
151         <div class="mcontainer">
152             <p> Thank you for applying loan with us.</p>
153             <p> Loan number: <code>loanNo</code></p>
154             <p> You applied for <code>loanType</code> loan on <code>date</code> </p>
155         </div>
156     }
157 
158     String querySql1 = "SELECT balance FROM hasBalance WHERE SSN=?";
159     Statement st1 = connection.createStatement();
160     ResultSet rs1 = st1.executeQuery(querySql1);
161     while(rs1.next())
162     {
163         // rs.getString(1); //or rs.getString("column name");
164         int loan_status = rs1.getInt("balance");
165         if(loan_status<4000)
166         {
167             <div class="mcontainer">
168                 <p>Congratulations. You are eligible for <code>loan_type</code> loan. </p>
169             </div>
170             break;
171         }
172         else
173         {
174             <div class="mcontainer">
175                 <p>We are sorry. You are not eligible for <code>loan_type</code> loan.  
Please check back again in few months. </p>
176             </div>
177             break;
178         }
179     }
180 
181 }
182 
183 </body>
184 </html>
185 
```

Line 142, Column 28

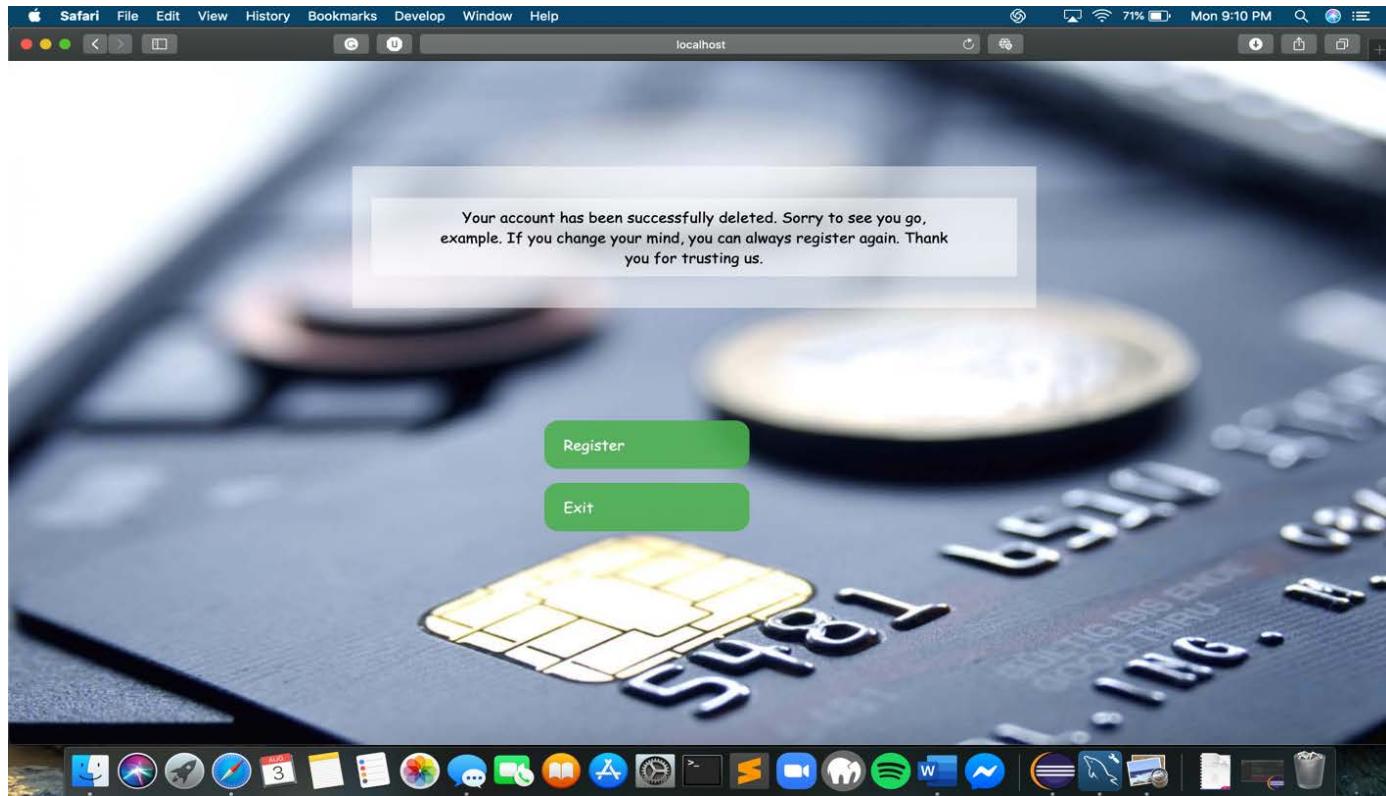
Tab Size: 4 Java Server Page (JSP)

## Back-end program

Once the data is submitted, this code will help to record loan information from user with SSN given, and display the result.



From homepage, if user chooses delete account button from the bottom left, it will redirect to delete account page where user puts account number, ssn and username..



Once the data is received, it gives the message that the account has been deleted. Also, given the option to register again.

A screenshot of the Sublime Text code editor. The active file is named "delete.jsp". The code is written in Java Server Page (JSP) and uses JDBC to interact with a MySQL database. It performs several delete operations: deleting from the "deletedAccount" table, the "userInfo" table, the "deposits" table, the "withdrawals" table, the "Transfers" table, and the "hasBalance" table. The code includes imports for java.sql.\*, javax.servlet.\*, and javax.servlet.http.\*. It uses PreparedStatement objects to execute SQL statements and Connection objects to get database connections. The code is annotated with line numbers from 56 to 116.

```

117
118    PreparedStatement ps6;
119    ps6 = connection.prepareStatement("DELETE FROM Statements WHERE account_id=?");
120
121    ps6.setString(1,account);
122
123    ps6.executeUpdate();
124
125    PreparedStatement ps7;
126    ps7 = connection.prepareStatement("DELETE FROM chequeBookOrder WHERE account_id=?");
127
128    ps7.setString(1,account);
129
130    ps7.executeUpdate();
131
132    PreparedStatement ps8;
133    ps8 = connection.prepareStatement("DELETE FROM USERS WHERE SSN=?");
134
135    ps8.setString(1,ssn);
136
137    ps8.executeUpdate();
138
139    PreparedStatement ps9;
140    ps9 = connection.prepareStatement("DELETE FROM Statements WHERE account_id=?");
141
142    ps9.setString(1,account);
143
144    ps9.executeUpdate();
145
146    PreparedStatement ps10;
147    ps10 = connection.prepareStatement("DELETE FROM chequeBookOrder WHERE account_id=?");
148
149    ps10.setString(1,account);
150
151    ps10.executeUpdate();
152
153
154    <div class="container">
155        <div style="text-align:center">Your account has been successfully deleted.
156        Sorry to see you go, <username %>.
157        If you change your mind, you can always register again.
158        Thank you for trusting us.
159    </div>
160
161    </div>
162
163
164
165
166
167 } catch (SQLException e) {
168     e.printStackTrace();
169 }
170
171
172
173
174    <a href="#">Registration.html</a> <button>Register</button>
175    <a href="#">Index.html</a> <button>Exit</button>
176
177
178
179
179

```

Line 1, Column 1      Tab Size: 4      Java Server Page (JSP)

## Back end

This chunk of code will help to delete user's information from all the tables where the given SSN is

MySQL Workbench

Administration   Schemas   Query 45   USERS   userinfo   deposits   Transfers   Statements   chequeBookOrder   userinfo   userinfo   >

SCHEMAS

ABCbank

Tables

chequeBookOrder   deletedAccount   deposits   hasBalance   Loan   Statements   Transfers   userinfo   USERS   withdrawals

Views

Stored Procedures

Functions

Arren Baral

Banking System

bankingsystem

cs152a

Object Info   Session

Table: deletedAccount

Columns:

accountno	username	DATE
1	example	2020-08-03

Action Output

Time	Action	Response	Duration / Fetch Time
29	21:11:55	SELECT * FROM ABCbank.deletedAccount LIMIT 0, 1000	1 row(s) returned 0.000038 sec / 0.000...

Result Grid

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

As we can see, the deleted account record has been recorded on the table.

MySQLWorkbench

File Edit View Query Database Server Tools Scripting Help

MySQL Workbench

Administration Schemas Query 45 deletedAccount hasBalance userinfo hasBalance

Context Help Snippets

Automatic context help is disabled. Use the toolbar to manually get help for the current care position or to toggle automatic help.

SCHEMAS

Filter objects ABCbank Tables chequeBookOrder deletedAccount deposits hasBalance Loan Statements Transfers userinfo USERS Withdrawals Views Stored Procedures Functions Arrenn Baral Banking System bankingsystem cs157a

Object Info Session

Table: deletedAccount

Columns:

accountno	int
username	varchar(45)
DATE	date

hasBalance 1

Action Output

Time	Action	Response	Duration / Fetch Time
29 16:20:47	SELECT * FROM ABCbank.hasBalance LIMIT 0, 1000	1 row(s) returned	0.00025 sec / 0.0000...

Result Grid Filter Rows: Search Export/Import: Result Grid Form Editor Field Types Query Stats Execution Plan

Result Grid

SSN balance

SSN	balance
12345	3627
HULL	HULL

Query Completed



So, after all the transaction, the total balance of this user is 3627. After the delete account option, every record will be cleared out. They cannot recover.

## **Project Conclusion**

*Arrenn Baral*

Throughout the summer semester, I got an opportunity to get exposed to database management system which was completely new topic. We learned from the basics of database management system to the complex one. Learning DB management is one of the important skills in the tech industry nowadays.

We started from the design of the project and ended with great implementation at the end of the summer session. We learned to use Github, Node JS, CSS and HTML to implement our design. These are very powerful tool to work on project. It was worth learning. This is my first web-based application which I feel pretty great about it.

One of the great lessons from this project is working with team member. Later in the future, we have to work with the team and this project helped to us to learn with the team. Collaboration is one of the hard things to implement, but it can be if worked together with teaming mindset.

*Prakriti Basyal*

Implementing and designing the banking database application of the project I learned multiple ways to handle the software system. Through our web application project, I get to learn more about the Mysql codes, HTML, CSS, and its implementations. This is pretty new to me to develop a web application, so I got a hand on experience dealing with ide and the server as well as developing the application. I have a more sense of databases now through this project. Collaborating with my teammate was one of the crucial things to finish the project. Time management was needed so I learned that it could help me in the future to manage my time for successful results. I had a little knowledge of using GitHub but through this project, I get to explore more about Github.

Finally, in this short amount of time this project, I have gained more knowledge and also have more ideas about the database management systems and the other tools to complete the project. It was challenging to learn all at once but learning this is going to be helpful for my future.

## Future Implementation

For the future improvements, we are planning on extra work on design and convenience to provide great user interface and experience. Right now, user needs to put their social security number and account number to make any kind of transactions or features. To reduce that redundancy, we will be implementing dynamic programming where user can update and see right away. Also, features like:

### Recover account

Right now, user cannot recover their account once deleted. We will add this feature in future.

### Better home page

User might experience redundancy with our system right now. So, we will design in a way that user experience easy interface.

### Activity log

User will be able to see activity log from the certain date.

### Joint Account

If user wishes to get a joint account like husband and wife, then they can request for join account.

Overall, in the future, our banking system will sure have great user experience. We try to work from user's perspectives. We need to think from user point of view to make our system to the easy and convenient. Since we chose banking system as our project and it deals with money, we need to win users trust and provide transparency with tiny details. It was a great opportunity to get to know the back work of banking system. Now we know how bank deals with the system and makes our life easy.