ONLINE BANK MANAGEMENT

Submitted To:

Prof. Mahendra Pratap Singh

Authors:

Jelwin Rodrigues - 17CO218
Parameshwaran Karthik - 17CO228

Table of Contents:

Introduction:

This project will represent the database used in banks. It will contain 6 main tables storing the information regarding Customer details, Account information, Loan information, Card information, Transaction Information, and Profile information.
There are 5 main functionalities developed which are Dashboard, Profile, Transfer_Money, Transactions, and Loan.

The Bank Account Management System is an application for maintaining a person's account in a bank. To develop a project for solving financial applications of a customer in banking environment in order to nurture the needs of an end banking user by providing various ways to perform banking tasks.

Online banking transactions include transferring money from one account to the other and taking loans from the bank. These services allow customers to directly contact the bank to do tasks that would otherwise take up a lot of time. In this web application one can create a new account, enter their details, and receive a new card it's number will be displayed on the dashboard. The customer also has to add a minimum amount of Rs 250 or above for opening a new account. Once logged in,
the user can add more money into his/her account. The user can view the account details and edit them. The user can also take loans from the bank for various reasons such as student loans, house loans, etc where each loan has a different rate of interest. The user can transfer money from one account to another if sufficient balance is available. Also, all the transactions are recorded in a table and will be visible to the user under transaction details.

The main number of database tables that we have used for our project is 6. They are listed below:
1)*access_control* which is used for DAC implementation by storing the read-write accesses for each subject-object pair in the database.
2*)loan_info* which includes all the loan details of the user.
3)*login* which has login info about all users.
4)*register* which has register details for all users, also includes the account number.
5)*mac_user* is the table that implements MAC for our application.
6)*transactions* that contain transaction history for every user.
Frontend = HTML , CSS and JavaScript
Backend = PHP
Database = MySQL

DAC Implementation(Discretionary Access Control):

In computer security, discretionary access control (DAC) is a type of access control defined "as a means of restricting access to objects based on the identity of subjects and/or groups to which they belong”. DAC mechanism controls are defined by user identification with supplied credentials during authentication, such as username and password. DACs are discretionary because the subject (owner) can transfer authenticated objects or information access to other users. In other words, the owner determines object access privileges. In DAC, each system object (file or data object) has an owner, and each initial object owner is the subject that causes its creation. Thus, an object's access policy is determined by its owner.

Implementation:

Our project has 3 types of users: Admin, Employee, and Customer. These are subjects in our case. There are 5 functionalities(objects) we have used. They are Dashboard, Profile, Transfer_Money, Transactions, and Loan. We create an Access Matrix List to give read-write access to the only privileged classes based on the owner. So the owner has the rights over the capability list that we have used. This access matrix is stored in the database as a matrix. Thus only an owner has access to the Access matrix and can change the access of the users accordingly. (In our implementation owner is ‘root’ and is in charge of the database, so he/she can update the records of access matrix). Any other users cannot update the entries in this Access Matrix. In our application whenever a user registers to create an account, there exists a field called ‘type_user’ which allows the one who registers to select the type of field he wants to register as.

Based on the filed registered, if type_user’ is admin, he has access to all the functionalities. If ‘type_user’ is an employee, he is restricted access to transactions page, and if the user is of customer type he has no access to transactions and transfer_money page. All this is implemented using the Access Control Matrix values from the database. The same applies while logging in also, whenever the user logs in, based on his user type, he/she will get redirected to the webpage based on the objects that the particular user can access.

Therefore the admin user has the following functionalities:

- Dashboard, Profile, Transfer_Money, Transactions, Loan, Take Loan, edit profile, and Change Password.

The employee class has the following functionalities:

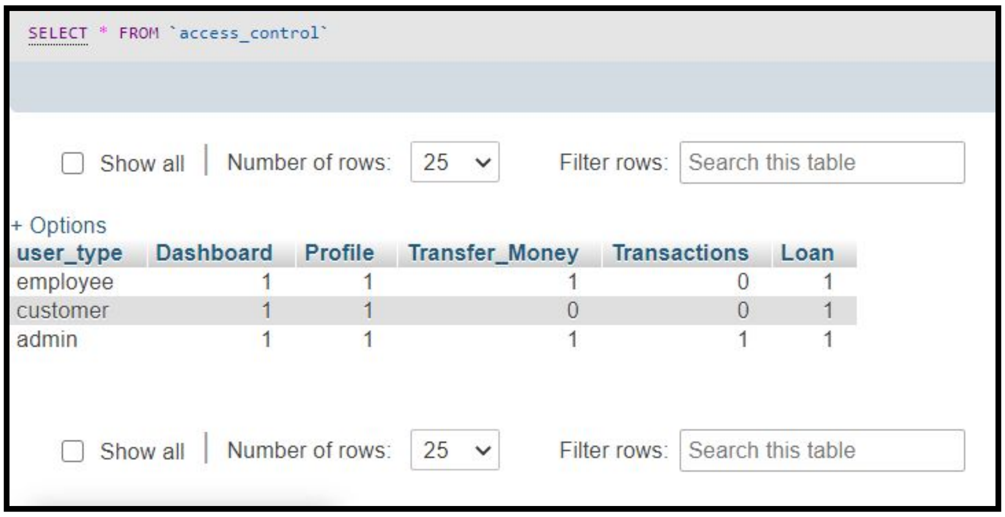
- Dashboard, Profile, Transfer_Money, Loan, Take Loan, edit profile, and Change Password.

And finally, the customer class has the following functionalities:

- Dashboard, Profile, Loan, Take Loan, edit profile, and Change Password.

Only an owner can change the entries of an Access Control Matrix as discussed above. We have not created a specific field for an owner in our application, but the owner we have assumed for our application is in charge of the database and is of type ‘root’. Also, the entries stored in the Access control Matrix are 0 or 1. 0 means read-write access is not permissible and 1 means read-write access is possible.

Results:
Below image shows the access control matrix stored in the Database:(access_control table)



```
SELECT * FROM `access_control`
```

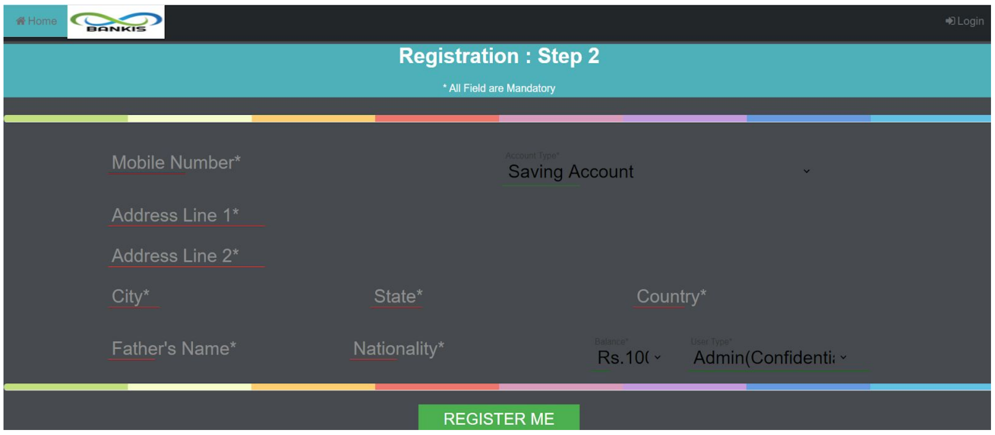
Show all | Number of rows: 25 Filter rows: Search this table

+ Options

user_type	Dashboard	Profile	Transfer_Money	Transactions	Loan
employee	1	1	1	0	1
customer	1	1	0	0	1
admin	1	1	1	1	1

Show all | Number of rows: 25 Filter rows: Search this table

Register Page has an entry user_type at the end(last field currently set to admin):

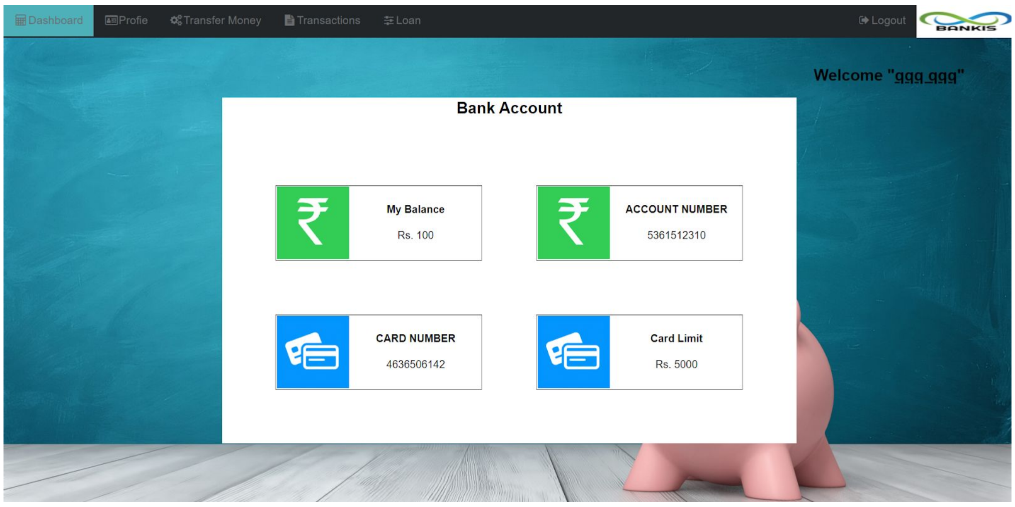


Registration : Step 2

* All Field are Mandatory

Mobile Number*		Account Type*: Saving Account	
Address Line 1*			
Address Line 2*			
City*	State*	Country*	
Father's Name*	Nationality*	Balance*: Rs.10(	User Type*: Admin(Confidenti

Admin Page:(implements all functionalities):

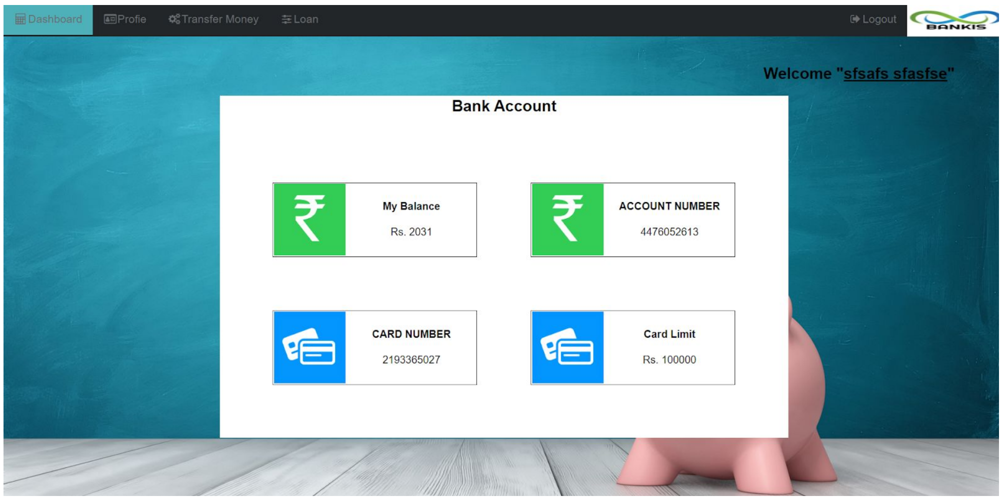


Welcome "qqq qqq"

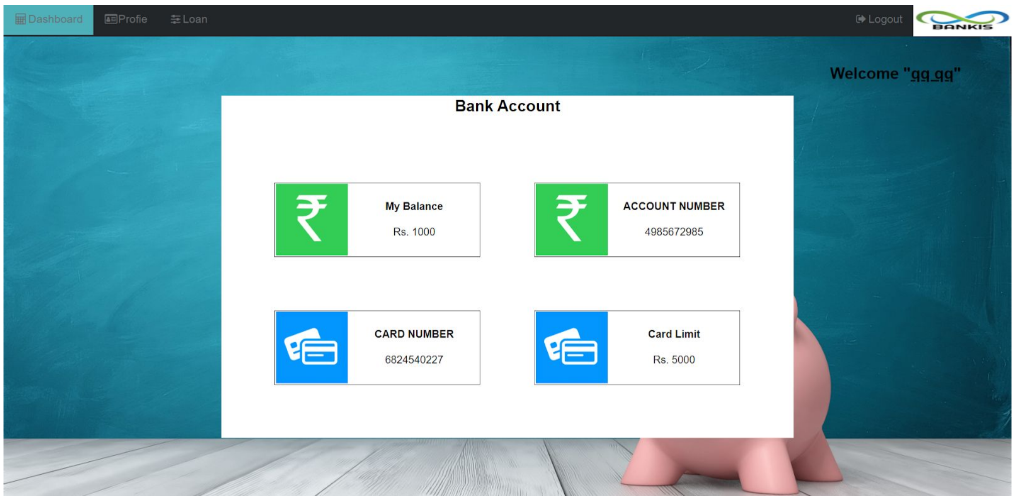
Bank Account

My Balance: Rs. 100	ACCOUNT NUMBER: 5361512310
CARD NUMBER: 4636506142	Card Limit: Rs. 5000

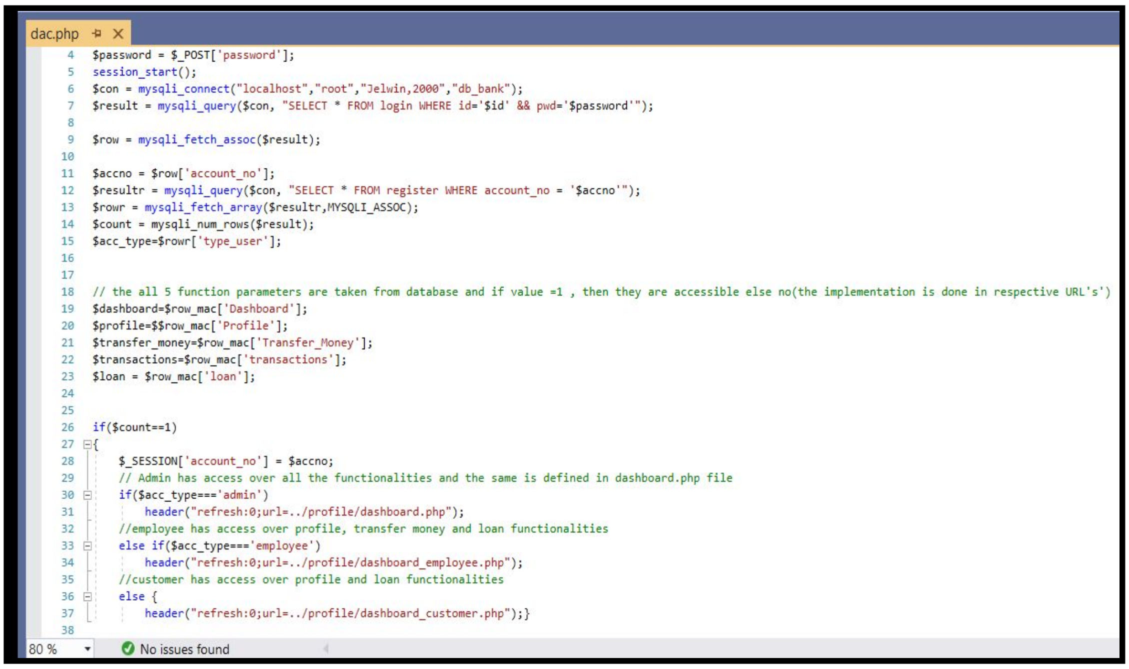
Employee Page:(Does not have transactions object)



Customer Page:(Does not have Transfer money and Transactions functionality)



Code section for generating the read-write access values:(.php file):


dac.php

```php
$password = $_POST['password'];
session_start();
$con = mysqli_connect("localhost","root","Jelwin,2000","db_bank");
$result = mysqli_query($con, "SELECT * FROM login WHERE id='$id' && pwd='$password'");

$row = mysqli_fetch_assoc($result);

$accno = $row['account_no'];
$resultr = mysqli_query($con, "SELECT * FROM register WHERE account_no = '$accno'");
$rowr = mysqli_fetch_array($resultr,MYSQLI_ASSOC);
$count = mysqli_num_rows($result);
$acc_type=$rowr['type_user'];


// the all 5 function parameters are taken from database and if value =1 , then they are accessible else no(the implementation is done in respective URL's')
$dashboard=$row_mac['Dashboard'];
$profile=$$row_mac['Profile'];
$transfer_money=$row_mac['Transfer_Money'];
$transactions=$row_mac['transactions'];
$loan = $row_mac['loan'];


if($count==1)
{
    $_SESSION['account_no'] = $accno;
    // Admin has access over all the functionalities and the same is defined in dashboard.php file
    if($acc_type==='admin')
        header("refresh:0;url=../profile/dashboard.php");
    //employee has access over profile, transfer money and loan functionalities
    else if($acc_type==='employee')
        header("refresh:0;url=../profile/dashboard_employee.php");
    //customer has access over profile and loan functionalities
    else {
        header("refresh:0;url=../profile/dashboard_customer.php");}

```

80 %   No issues found

STEPS TO FOLLOW FOR RUNNING APPLICATION:
Steps for implementations of our project:(Steps 1-2 for running XAMPP, 3 for installing our application, 4 -8 for setting up the database, and 9 for the index page of our application)

1) Install XAMPP with Apache software:(link to download XAMPP below)
   Link: https://www.apachefriends.org/download.html
   Save to directory C:\xampp
2) Open 'xampp-control' application in the 'C:\xampp' folder and Start the Apache and MySQL modules.
3) Download the bankis_bank_dac zip file and extract it to C:\xampp\htdocs folder
4) The database is obtained by the URL: http://localhost/phpmyadmin/
5) Create a new database and name it as 'db_bank' and in the PHPMyAdmin, select this database and click on the 'import' button on the top navigation bar.
6) Import the database from the local file which was extracted, i.e. from C:\xampphtdocs\bankis_bank_dac\SQL\db_bank.sql file.
7) Create a new database and name it as 'db_transactions' and in the PHPMyAdmin, select this database and click on the 'import' button on the top navigation bar.
8) Import the database from the local file which was extracted, i.e. from C:\xampphtdocs\bankis_bank_dac\SQL\db_transactions.sql file.
9) Now we can start our application with the help of the URL:
   http://localhost/bankis_bank_dac/register/register.php
   And now can access the application based on access rights.

MAC Implementation(Mandatory Access Control):

Mandatory Access Control is based on a hierarchical model. The hierarchy is based on the security level. All users are assigned a security or clearance level. All objects are assigned a security label. Users can access only resources that correspond to a security level equal to or lower than theirs in the hierarchy.

In a MAC model, access is controlled strictly by the administrator. The administrator is the one who sets all permissions. Users cannot set permissions themselves, even if they own the object. Because of this, MAC systems are considered very secure.

Subjects and objects have clearances and labels, respectively, such as confidential, secret, and top secret. A subject may access an object only if the subject's clearance is equal to or greater than the object's label. Subjects cannot share objects with other subjects who lack the proper clearance or “write down” objects to a lower classification level.

Implementation:

Our project has 3 types of users: Admin, Employee, and Customer. These are subjects in our case. There are 5 functionalities(objects) we have used. They are Dashboard, Profile, Transfer_Money, Transactions, and Loan. We assign a security level to each of these subjects based on how we want them to access the objects. So, Admin is given the *Top_Secret* security level. Also, Employee is given the *Secret* security level and the customer is given the least security level, i.e. the *confidential* security level. All this is stored in the database and is accessed whenever the user registers or logs in to the application. Now in order to implement MAC, we need to assign security levels to the objects as well so that we can compare the security levels based on integrity models that exist. A subject can have access to an object if it’s security level is higher than compared to the object. So with this in mind, in our application, we assign accessible objects based on the security levels of the objects which are defined in the database separately in the ‘mac_user’ table. In this table, we have defined the security levels of objects as follows:
Dashboard=3, Profile=3, Transactions=1, Transfer_Money=2, Loan=3
Here 3 signifies Confidential level, 2 signifies Secret level, and 1 signifies Top Security level.

Also, we already know the security level of the subjects,i.e.
Admin=1, Employee=2, Customer=3

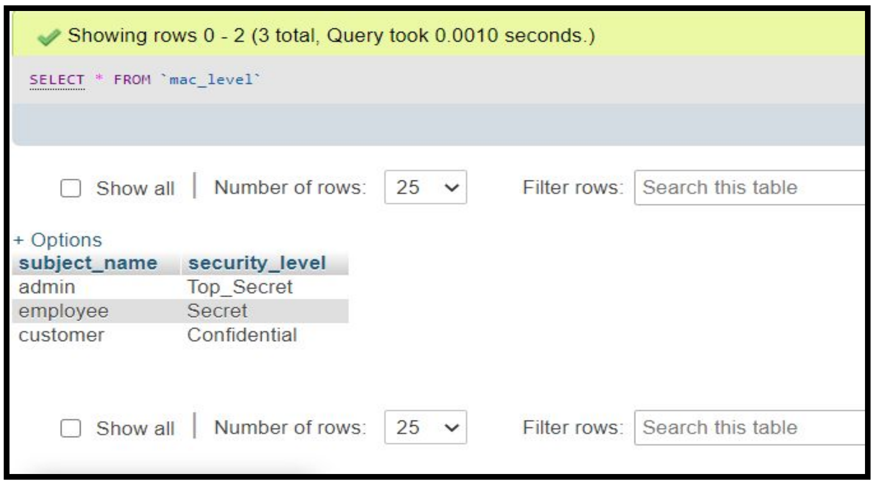
Based on this we get the objects accessible by subjects as follows:(since we have used rank to denote security level, lower value=>higher security level)

- <u>For Admin:</u>
  Security_level(Admin) is greater than all the objects, therefore Admin has access to all the objects.
- <u>For Employee:</u>
  Security_level(Employee)<=Security_level(Transactions), but
  Security_level(Employee) is greater for all other objects, therefore employee has access to all objects bar Transactions.
- <u>For Customer:</u>
  Security_level(Customer)<=Security_level(Transactions)
  Security_level(Customer)<=Security_level(Transfer_Money)
  For other objects, Security_level(Customer) is greater. Therefore, the Customer has access to Dashboard, Profile, and Loan.

Also, we have assumed admins as owners for our MAC implementation. Therefore, admins can view the security rights of all kinds of users. (This is for display purposes).But it’s not possible to edit as it’s stored in database and MAC is a secure implementation,

Results:

Below image shows the security_level table stored in the Database:(Levels discussed above)

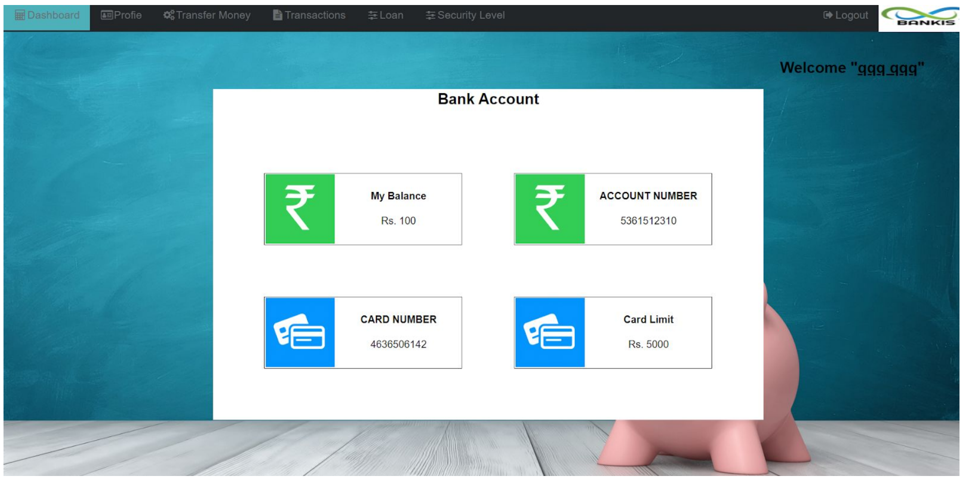


Showing rows 0 - 2 (3 total, Query took 0.0010 seconds.)

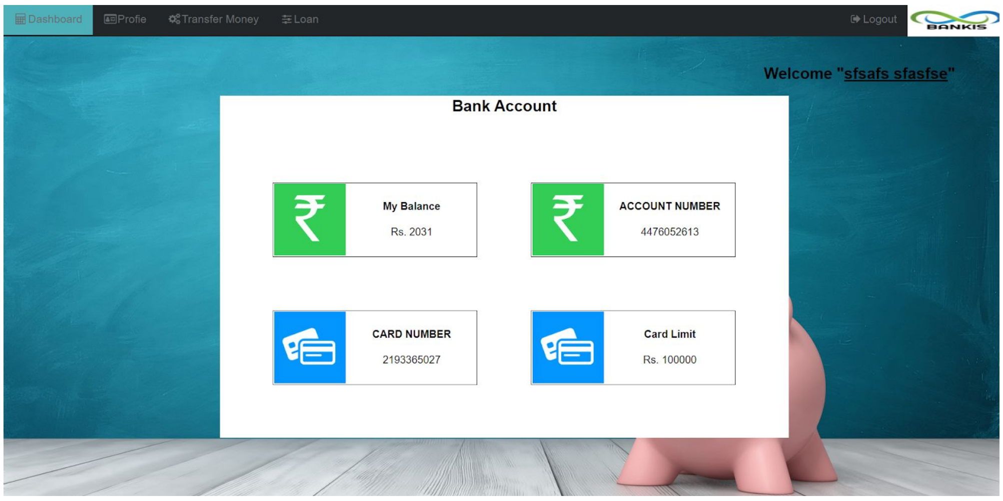
```
SELECT * FROM `mac_level`
```

subject_name	security_level
admin	Top_Secret
employee	Secret
customer	Confidential

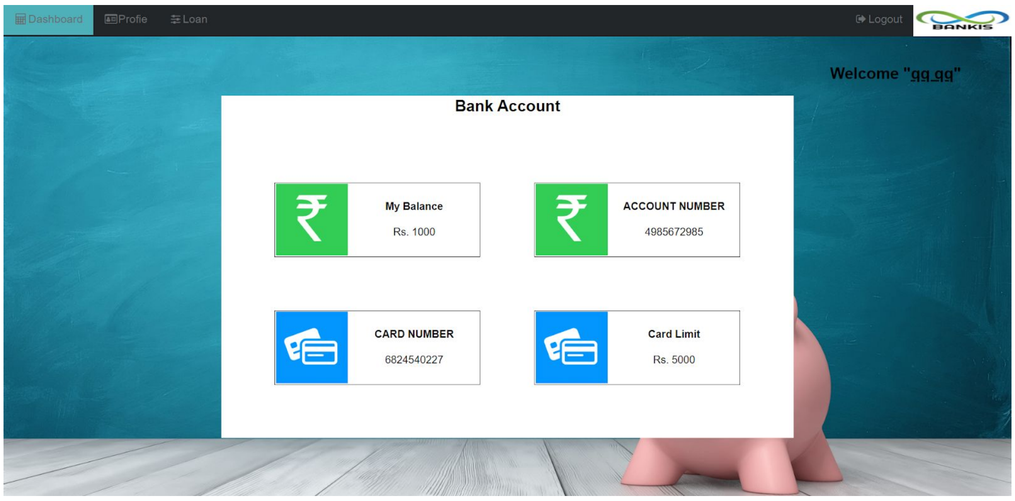
Admin Page:(implements all functionalities. Notice an additional *security level* functionality)



Employee Page:(Does not have transactions object)(Also security level object is missing)



Customer Page:(Does not have Transfer money and Transactions functionality)

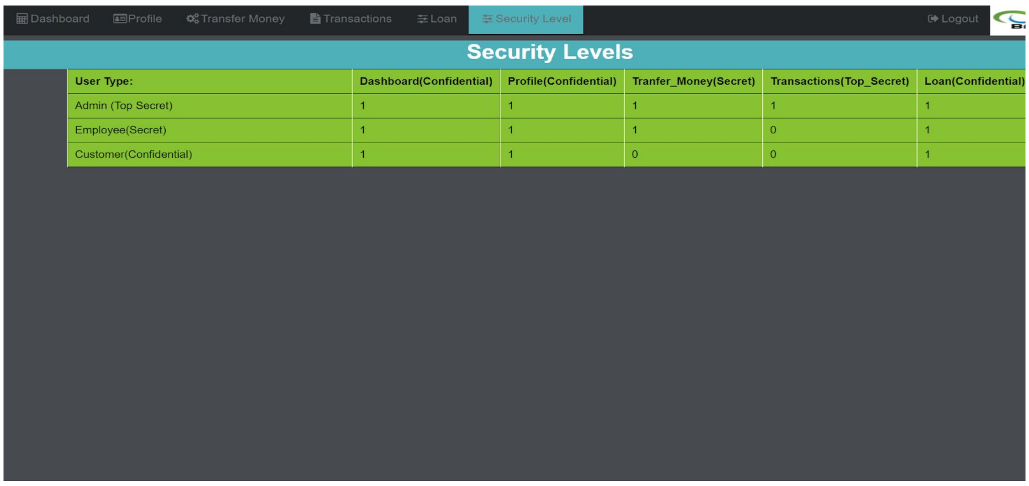


Welcome "gg gg"

Bank Account

My Balance	Rs. 1000
ACCOUNT NUMBER	4985672985
CARD NUMBER	6824540227
Card Limit	Rs. 5000

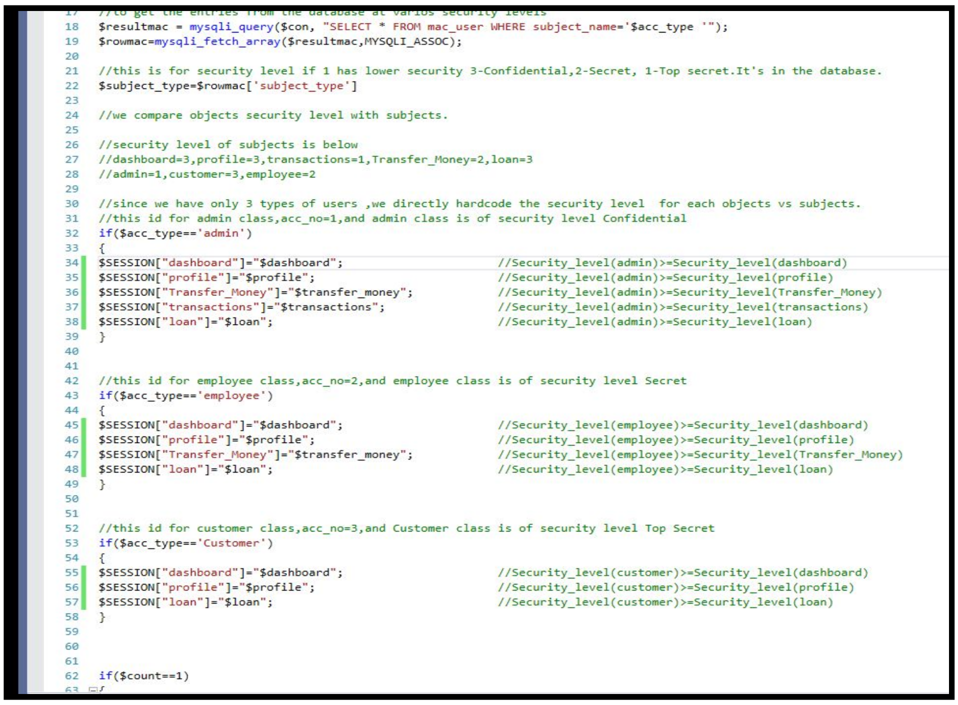
Security level:(Only available for admins as discussed above)(Cannot edit)



Security Levels

User Type:	Dashboard(Confidential)	Profile(Confidential)	Tranfer_Money(Secret)	Transactions(Top_Secret)	Loan(Confidential)
Admin (Top Secret)	1	1	1	1	1
Employee(Secret)	1	1	1	0	1
Customer(Confidential)	1	1	0	0	1

Code section for generating the security level values and necessary access rights:(.php file):



```
//to get the entries from the database at varios security levels
$resultmac = mysqli_query($con, "SELECT * FROM mac_user WHERE subject_name='$acc_type '");
$rowmac=mysqli_fetch_array($resultmac,MYSQLI_ASSOC);

//this is for security level if 1 has lower security 3-Confidential,2-Secret, 1-Top secret.It's in the database.
$subject_type=$rowmac['subject_type']

//we compare objects security level with subjects.

//security level of subjects is below
//dashboard=3,profile=3,transactions=1,Transfer_Money=2,loan=3
//admin=1,customer=3,employee=2

//since we have only 3 types of users ,we directly hardcode the security level  for each objects vs subjects.
//this id for admin class,acc_no=1,and admin class is of security level Confidential
if($acc_type=='admin')
{
$SESSION["dashboard"]="$dashboard";                    //Security_level(admin)>=Security_level(dashboard)
$SESSION["profile"]="$profile";                        //Security_level(admin)>=Security_level(profile)
$SESSION["Transfer_Money"]="$transfer_money";          //Security_level(admin)>=Security_level(Transfer_Money)
$SESSION["transactions"]="$transactions";              //Security_level(admin)>=Security_level(transactions)
$SESSION["loan"]="$loan";                              //Security_level(admin)>=Security_level(loan)
}


//this id for employee class,acc_no=2,and employee class is of security level Secret
if($acc_type=='employee')
{
$SESSION["dashboard"]="$dashboard";                    //Security_level(employee)>=Security_level(dashboard)
$SESSION["profile"]="$profile";                        //Security_level(employee)>=Security_level(profile)
$SESSION["Transfer_Money"]="$transfer_money";          //Security_level(employee)>=Security_level(Transfer_Money)
$SESSION["loan"]="$loan";                              //Security_level(employee)>=Security_level(loan)
}


//this id for customer class,acc_no=3,and Customer class is of security level Top Secret
if($acc_type=='Customer')
{
$SESSION["dashboard"]="$dashboard";                    //Security_level(customer)>=Security_level(dashboard)
$SESSION["profile"]="$profile";                        //Security_level(customer)>=Security_level(profile)
$SESSION["loan"]="$loan";                              //Security_level(customer)>=Security_level(loan)
}



if($count==1)
{
```

STEPS TO FOLLOW FOR RUNNING APPLICATION:
Steps for implementations of our project:(Steps 1-2 for running XAMPP, 3 for installing our application, 4 -8 for setting up the database, and 9 for the index page of our application)

1) Install XAMPP with Apache software:(link to download XAMPP below)
   Link: https://www.apachefriends.org/download.html
   Save to directory C:\xampp
2) Open 'xampp-control' application in the 'C:\xampp' folder and Start the Apache and MySQL modules.
3) Download the bankis_bank_mac zip file and extract it to C:\xampp\htdocs folder
4) The database is obtained by the URL: http://localhost/phpmyadmin/
5) Create a new database and name it as 'db_bank' and in the PHPMyAdmin, select this database and click on the 'import' button on the top navigation bar.
6) Import the database from the local file which was extracted, i.e. from C:\xampphtdocs\bankis_bank_mac\SQL\db_bank.sql file.
7) Create a new database and name it as 'db_transactions' and in the PHPMyAdmin, select this database and click on the 'import' button on the top navigation bar.
8) Import the database from the local file which was extracted, i.e. from C:\xampphtdocs\bankis_bank_mac\SQL\db_transactions.sql file.
9) Now we can start our application with the help of the URL:
   http://localhost/bankis_bank_mac/register/register.php
   And now can access the application based on security levels.

RBAC(ROLE-BASED ACCESS CONTROL):

The Role-based Access Control (RBAC) Model has garnered great interest in the security community due to the flexible and secure nature of its applicability to the complex and sophisticated information system. But there are few research results about implementation analysis on RBAC. With the help of partition number theory, this paper provides a method of RBAC to implement data access permissions control in the major application system. One of the important research areas in this model is that the agent mechanism is introduced to implement RBAC. Secondly, when a user enters the access system and gets a session, the user will be activated and the permission is granted simultaneously to ensure safe and optimal operation in his session. Finally, the implementation is analyzed for verifying the applicable value of the mode

Implementation:
This project has 32types of users: Admin, and Customer. These are subjects in our case. There are 5 functionalities(objects) we have used. They are Dashboard, Profile, Transfer_Money, Transactions, and Loan. We assign a security level to each of these subjects based on how we want them to access the objects. So, Admin is given the *Top_Secret* security level. The customer is given the least security level, i.e. the *confidential* security level. We have different types of customers, i.e, students, non loan accounts, and others. Each one of them has different access priorities in the application. During the registration of each user the roles are assigned to them. This can further only be changed by the admin. All this is stored in the database and is accessed whenever the user registers or logs in to the application. Now in order to implement RBAC, we need to assign security levels to the objects as well so that we can compare the security levels based on integrity models that exist. A subject can have access to an object if it's security level is higher compared to the object. So with this in mind, in our application, we assign accessible objects based on the security levels of the objects which are defined in the database separately in the 'role' table. In this table, we have defined the security levels of objects as follows:
Dashboard=all, Profile=all, Transactions=all, Transfer_Money=all, Loan=1,2
The security level of the subjects are
Student = 1, Others=2, No Loan=3

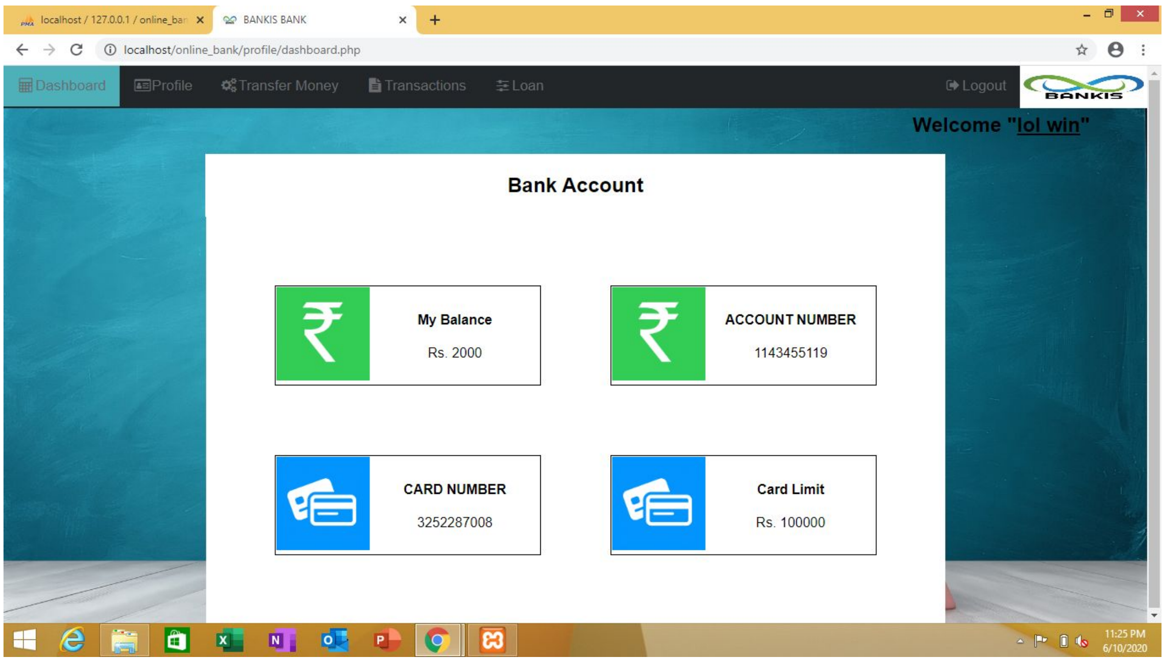
Based on this we get the objects accessible by subjects as follows:

- <u>For Student:</u>
  Security_level(Student)<=Security_level(Transactions)
  Security_level(Student)<=Security_level(Transfer_Money)
  Security_level(Student)<=Security_level(Student_Loan_only)
  For other objects, Security_level(Student) is greater. Therefore, the Student has access to everything bar normal loans.
- <u>For Others:</u>
  Security_level(Others)<=Security_level(Transactions)
  Security_level(Others)<=Security_level(Transfer_Money)
  Security_level(Others)<=Security_level(No_Student_Loan)
  For other objects, Security_level(Others) is greater. Therefore, the Student has access to everything bar Student loans.
- <u>For No Loan:</u>
  Security_level(No_loan)<=Security_level(Transactions)
  Security_level(No_loan)<=Security_level(Transfer_Money)

Security_level(No_loan)<=Security_level(No_loan)
For other objects, Security_level(No_loan) is greater. Therefore, the Student has access to everything bar loans.

Also, we have assumed admins as owners for our RBAC implementation. Therefore, admins can view the security rights of all kinds of users. (This is for display purposes).But it's not possible to edit as it's stored in database and RBAC is a secure implementation,

Results:


localhost / 127.0.0.1 / online_ban
BANKIS BANK
localhost/online_bank/profile/dashboard.php
Dashboard
Profile
Transfer Money
Transactions
Loan
Logout
BANKIS

Welcome "lol win"

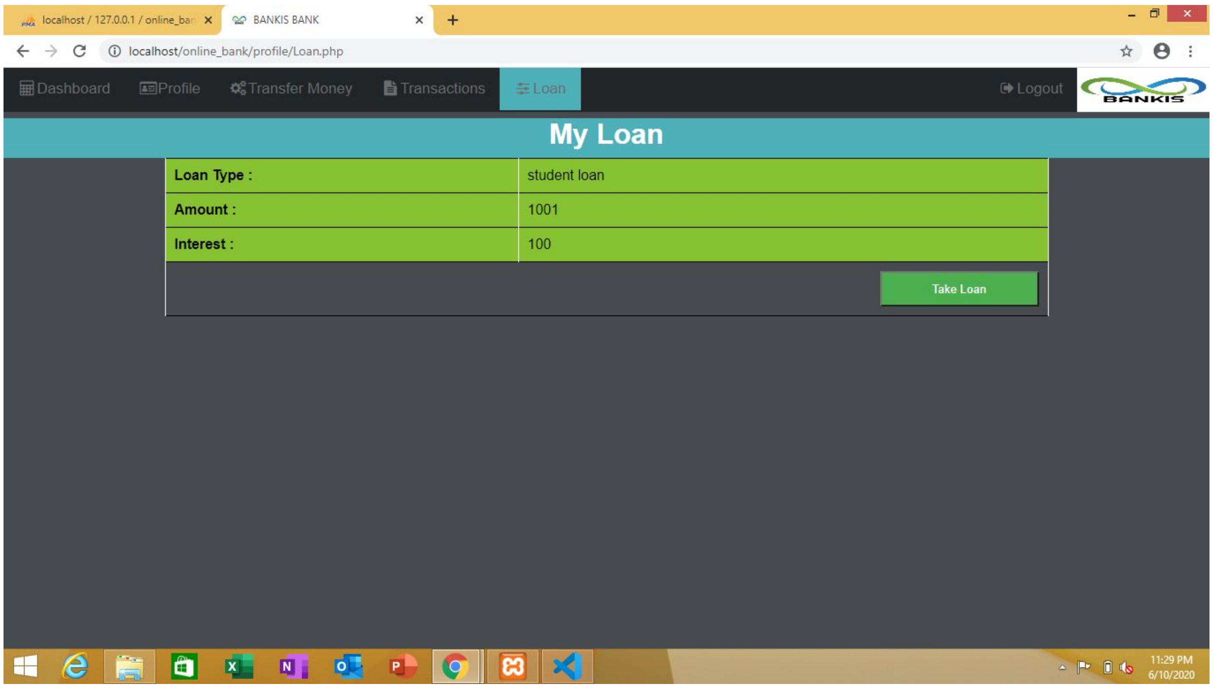
Bank Account

My Balance
Rs. 2000

ACCOUNT NUMBER
1143455119

CARD NUMBER
3252287008

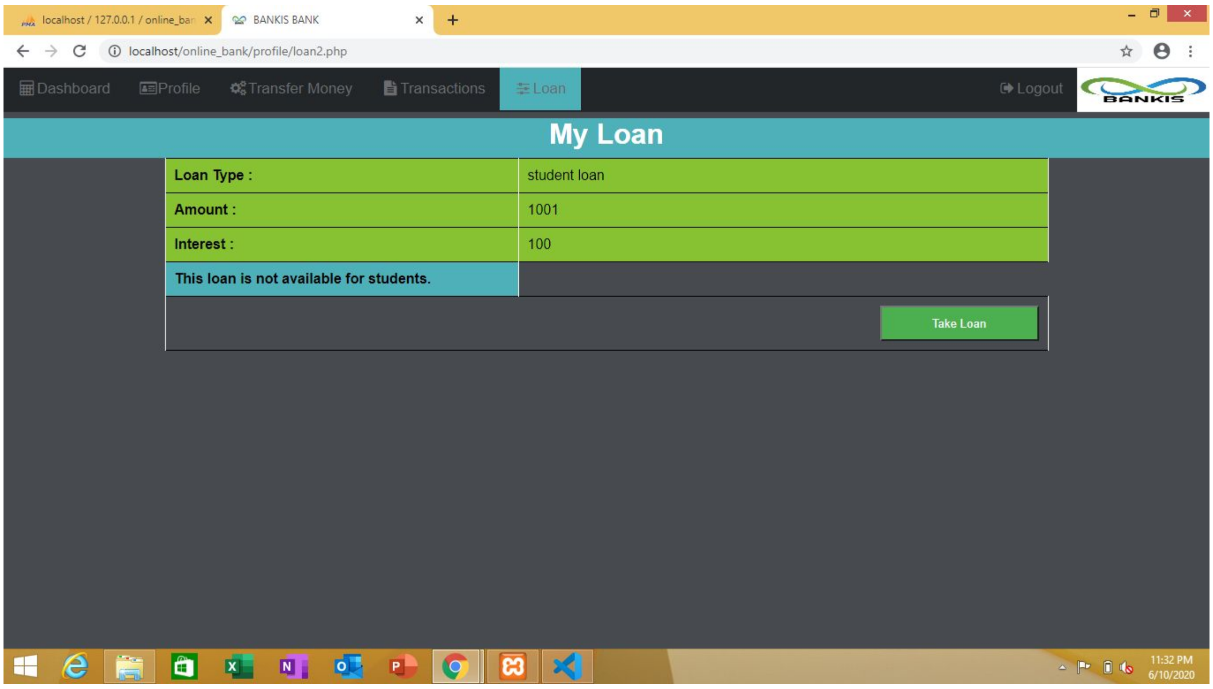
Card Limit
Rs. 100000



localhost/online_bank/profile/Loan.php

My Loan

Loan Type :	student loan
Amount :	1001
Interest :	100

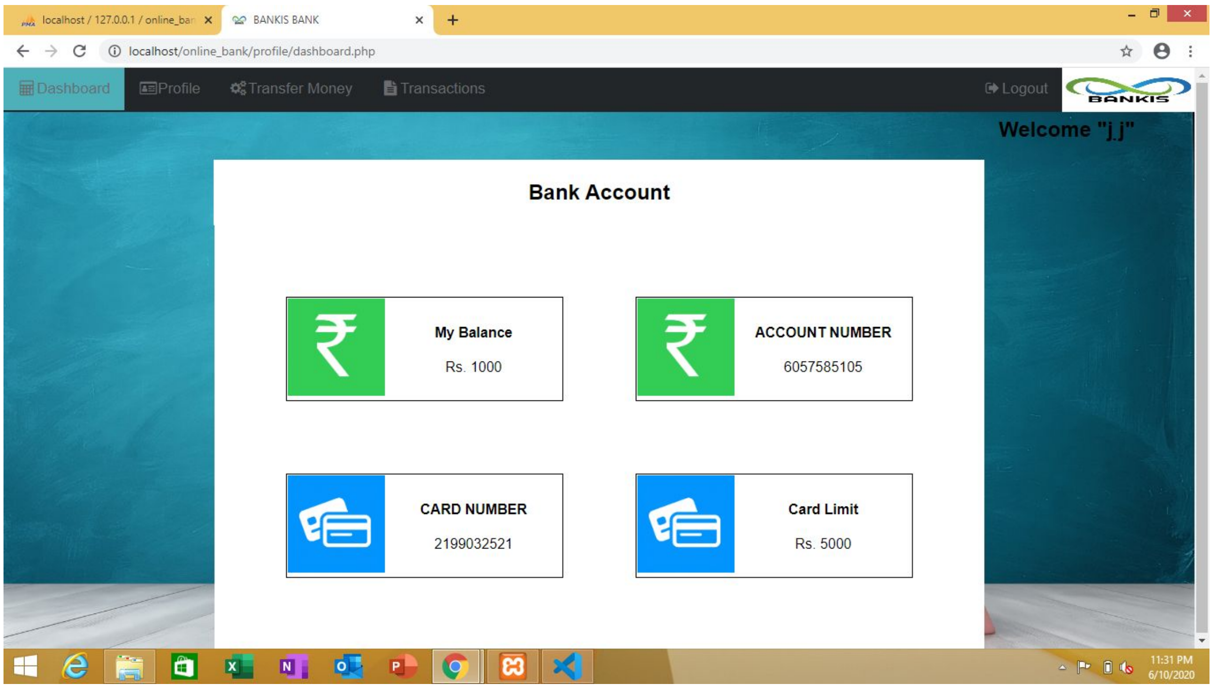


localhost/online_bank/profile/loan2.php

My Loan

Loan Type :	student loan
Amount :	1001
Interest :	100
This loan is not available for students.	

The above 3 are Student implementations where the loan forms are specific.


localhost/online_bank/profile/dashboard.php

Dashboard | Profile | Transfer Money | Transactions | Logout

Welcome "j.j"

Bank Account

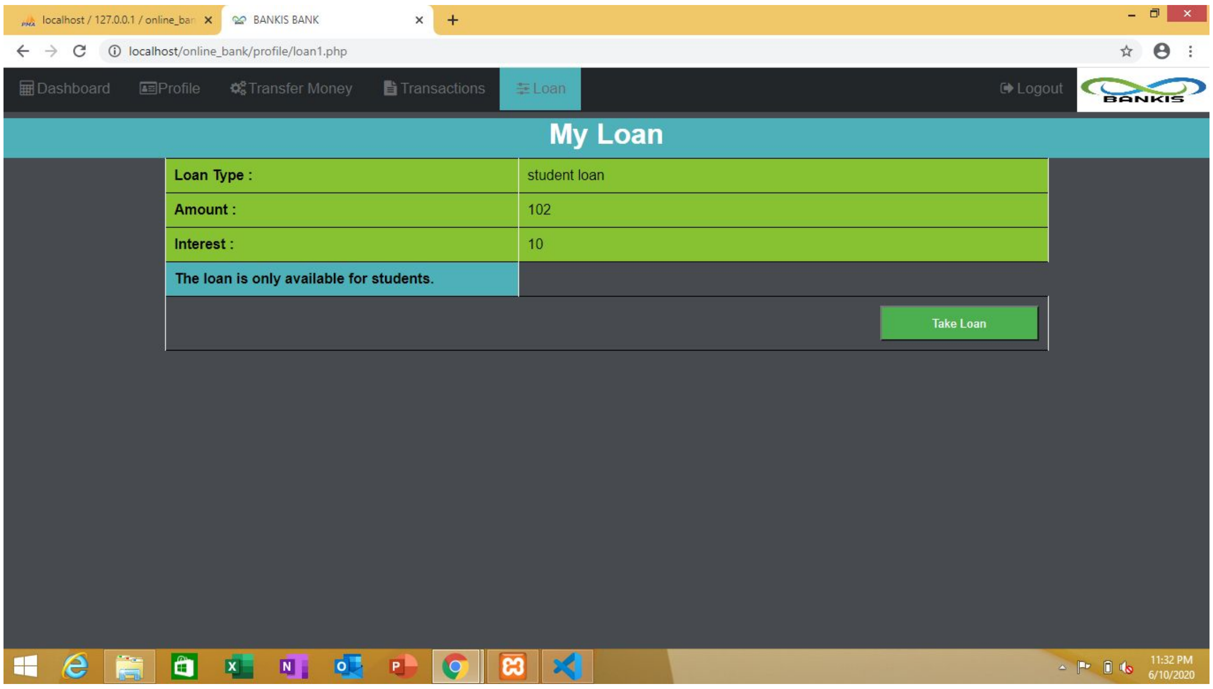
My Balance
Rs. 1000

ACCOUNT NUMBER
6057585105

CARD NUMBER
2199032521

Card Limit
Rs. 5000

This is the implementation of No loan account where the loan header is not accessible.


localhost/online_bank/profile/loan1.php

Dashboard | Profile | Transfer Money | Transactions | Loan | Logout

My Loan

Loan Type :	student loan
Amount :	102
Interest :	10
The loan is only available for students.	

Take Loan

This is the ‘others’ implementation, where student loans are not available.

STEPS TO FOLLOW FOR RUNNING APPLICATION:

Steps for implementations of our project:(Steps 1-2 for running XAMPP, 3 for installing our application, 4 -8 for setting up the database, and 9 for the index page of our application)

1) Install XAMPP with Apache software:(link to download XAMPP below)
   Link: https://www.apachefriends.org/download.html
   Save to directory C:\xampp
2) Open 'xampp-control' application in the 'C:\xampp' folder and Start the Apache and MySQL modules.
3) Download the bankis_bank_rbac zip file and extract it to C:\xampp\htdocs folder
4) The database is obtained by the URL: http://localhost/phpmyadmin/
5) Create a new database and name it as 'db_bank' and in the PHPMyAdmin, select this database and click on the 'import' button on the top navigation bar.
6) Import the database from the local file which was extracted, i.e. from C:\xampphtdocs\bankis_bank_rbac\SQL\db_bank.sql file.
7) Create a new database and name it as 'db_transactions' and in the PHPMyAdmin, select this database and click on the 'import' button on the top navigation bar.
8) Import the database from the local file which was extracted, i.e. from C:\xampphtdocs\bankis_bank_rbac\SQL\db_transactions.sql file.
9) Now we can start our application with the help of the URL:
   http://localhost/bankis_bank_rbac/register/register.php
   And now can access the application based on security levels.

Conclusion and Future Work:

In these implementations we have tried to use partition numbers to describe the rules of our three access control. We have implemented DAC, MAC, RBAC for an online banking system and briefly explained the working of all the 3 models. Each model is very useful based on the systems they are going to be used on. The results we have obtained from the project shows us that technology can be utilized in the security domain with clear semantics by providing sharable and reasonable security measures.

Many of the benefits of doing our banking online are obvious: You don't have to wait in line. You can look at your balance whenever you want, not just when you get a statement. There are some other benefits too. As a young bank customer, you're just learning how to manage your money and observe your spending patterns. This project is developed to nurture the needs of a user in the banking sector including almost all the tasks of transactions taking place in a bank. Online banking is an innovative tool that is fast becoming a necessity. It is a successful strategic weapon for banks to remain profitable today. Security is a very important and difficult issue for every area. Security is mainly about protecting possessions or resources(the subjects as in our case)

References:

1)https://www.sciencedirect.com/topics/computer-science/mandatory-access-control
2)Jinyu, C., & Tong, Z. (2009). *Research and Implementation of Role-Based Access Control Model Based on Partition Number. 2009 Second International Symposium on Computational Intelligence and Design.* doi:10.1109/iscid.2009.150
3)https://www.techopedia.com/definition/229/discretionary-access-control-dac
4)https://www.techotopia.com/index.php/Mandatory,_Discretionary,_Role_and_Rule_Based_Access_Control
5)https://www.scs.stanford.edu/05au-cs240c/notes/l13.pdf