Chapter 5
system implementation

5.1 Introduction

This chapter discusses the implementation of the system.

Section 5.2 provides the Mapping design for implementation and information about the system, including the system and software design decisions taken.

Section 5.3 flies and Sample project codes outlines the structure of the system, showing the various directories, file organization and Sample of project code.

Section 5.4 System testing \System Testing is a level of testing that validates the complete and fully integrated software product.

The purpose of a system test is to evaluate the end-to-end system specifications.

Usually, the software is only one element of a larger computer-based system. Ultimately, the software is interfaced with other software/hardware systems.

System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

we will test our system via two Categories of Software Testing

- Black Box Testing

- White Box Testing

System test falls under the black box testing category of software testing.

White box testing is the testing of the internal workings or code of a software application.

In contrast, black box or System Testing is the opposite.

System test involves the external workings of the software from the user's perspective.

Problems such as results delay, errors and human mistakes encountered with results management system in some academic environments is complex and worrisome.

By studying the existing system, it greatly improves the management methods of the organization in which it is applied, and then improves the system, presenting reports, and thus, making important decisions about the results that are presented.

5.2 Mapping design to be implemented

We can find steps to turn the idea into a truly working system with the help of UML diagrams To represent these steps, we can start with the central part, which is the server

1-You need to have a central server that hosts all the data

2-You need a partitioned database according to the picture shown in figure 4.1.

3-You need the hardware device which you can build according to chapter 2 part7.

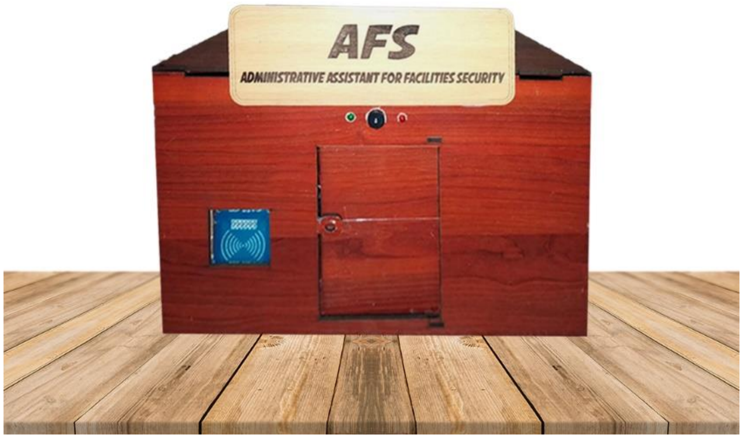
4-You need the Dashboard to enable the admin to control the system's basics, issue reports, and determine the users' permissions.

5-You need tags to represent users and use as a unique tag for each person

5.3 Sample project codes

In this part, we will discuss the parts of the system and explain their functions, which in turn will help to understand the system.

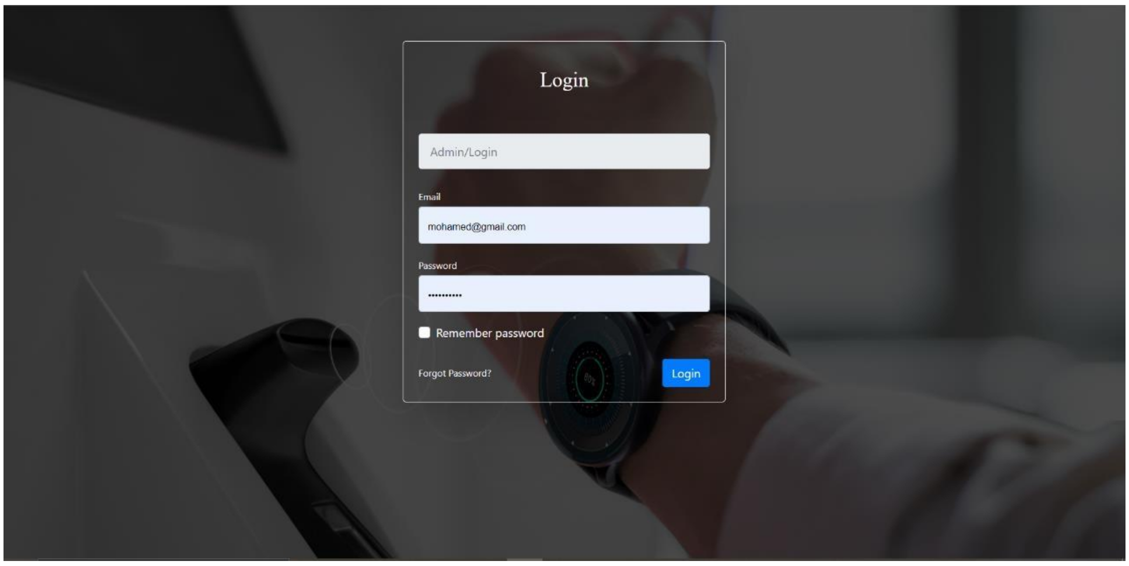
We can start showing a picture of the device Then a sample of the code for the microcontroller Then the code written in Python, which can be considered as a middle layer between the server and Arduino to achieve complete communication between them. Then display pictures of the dashboard and powers of the admin and his activities within the system Then it displays parts of the helpers, which are some codes that are repeated during. Nested implementation in core functions, specific performance.



• Index page in dashboard

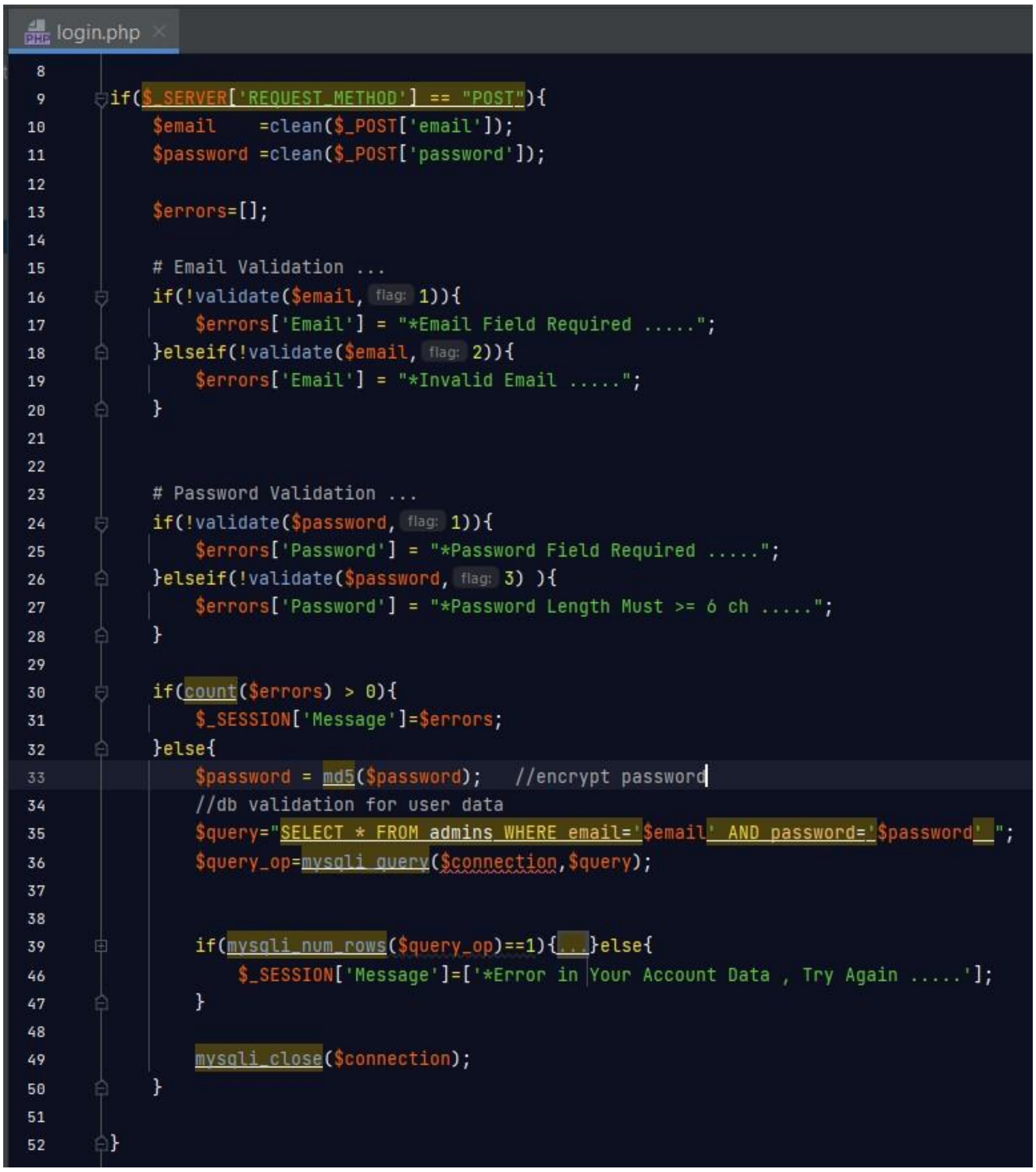
Here we will show some pictures from the final dashboard and some code samples that describe how this dashboard working.

Login page



As shown login from is very simple, it requires email and password to get into the main page of the dashboard.

Login code sample

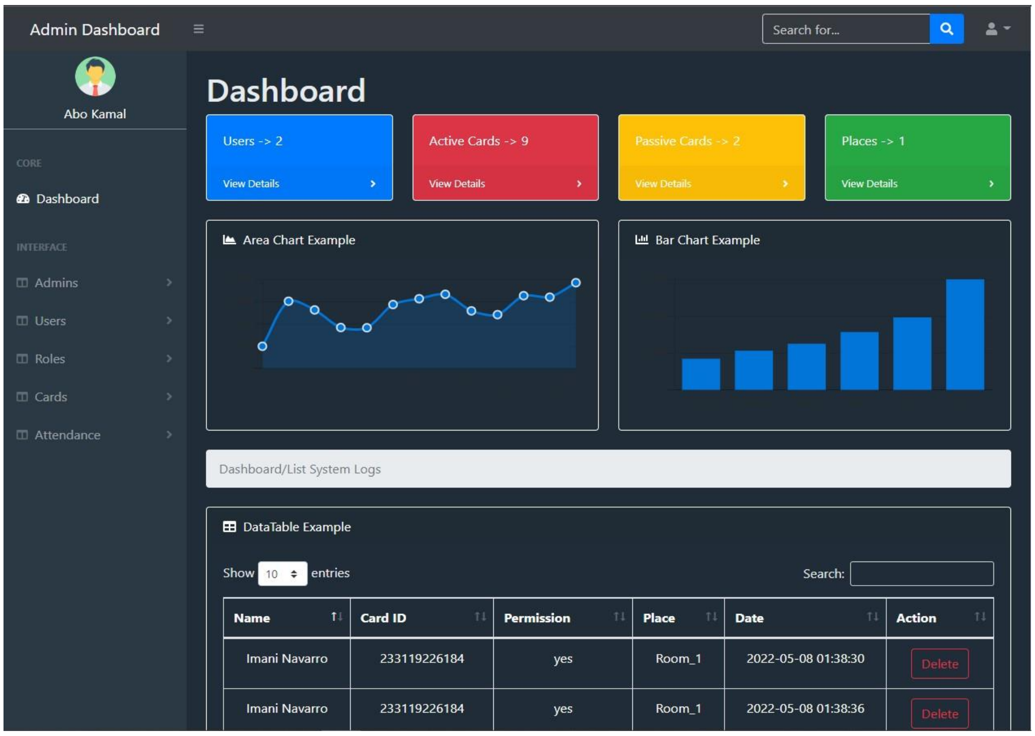


login.php

```php

if($_SERVER['REQUEST_METHOD'] == "POST"){
    $email    =clean($_POST['email']);
    $password =clean($_POST['password']);

    $errors=[];

    # Email Validation ...
    if(!validate($email, flag: 1)){
        $errors['Email'] = "*Email Field Required .....";
    }elseif(!validate($email, flag: 2)){
        $errors['Email'] = "*Invalid Email .....";
    }


    # Password Validation ...
    if(!validate($password, flag: 1)){
        $errors['Password'] = "*Password Field Required .....";
    }elseif(!validate($password, flag: 3) ){
        $errors['Password'] = "*Password Length Must >= 6 ch .....";
    }

    if(count($errors) > 0){
        $_SESSION['Message']=$errors;
    }else{
        $password = md5($password);   //encrypt password
        //db validation for user data
        $query="SELECT * FROM admins WHERE email='$email' AND password='$password' ";
        $query_op=mysqli_query($connection,$query);


        if(mysqli_num_rows($query_op)==1){...}else{
            $_SESSION['Message']=['*Error in Your Account Data , Try Again .....'];
        }

        mysqli_close($connection);
    }

}
```

As shown here, after user press login button form data is sent to this script according to property “action” of the form, then validation rules are applied to the coming data, if there are no errors data will be thrown into the next security level which is matching coming data with its counterpart in the database.

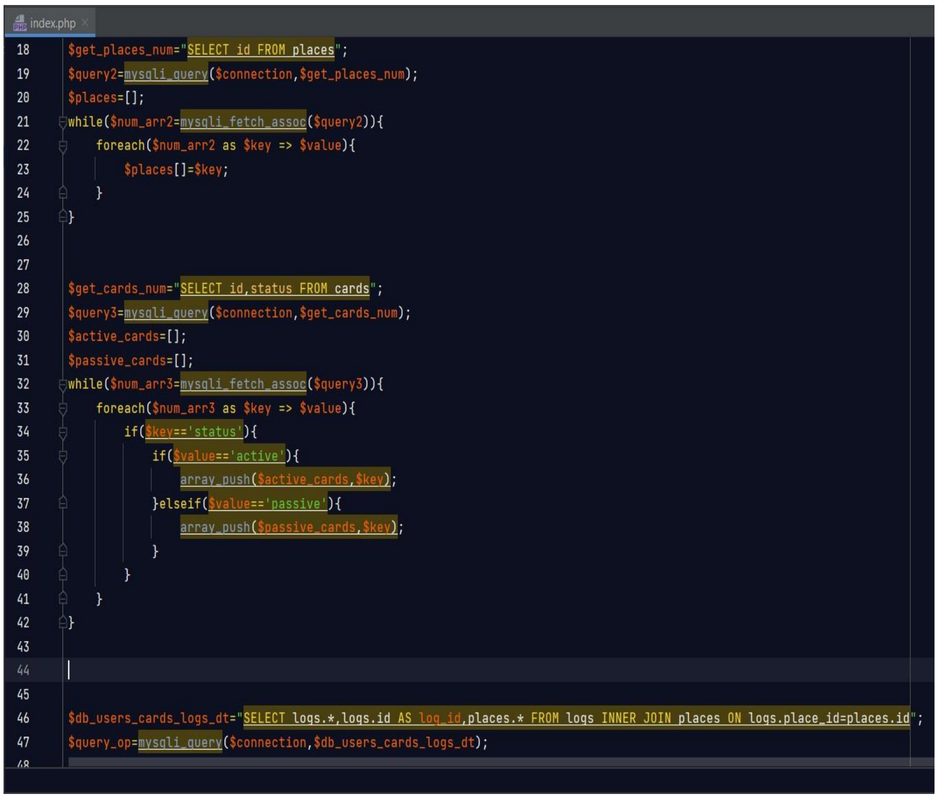
Main dashboard page



Name	Card ID	Permission	Place	Date	Action
Imani Navarro	233119226184	yes	Room_1	2022-05-08 01:38:30	Delete
Imani Navarro	233119226184	yes	Room_1	2022-05-08 01:38:36	Delete

As we can see here is the main page in the dashboard, it contains some important components, in general it has a detailed statistics and information about the whole system.

Dashboard code sample [part 1]

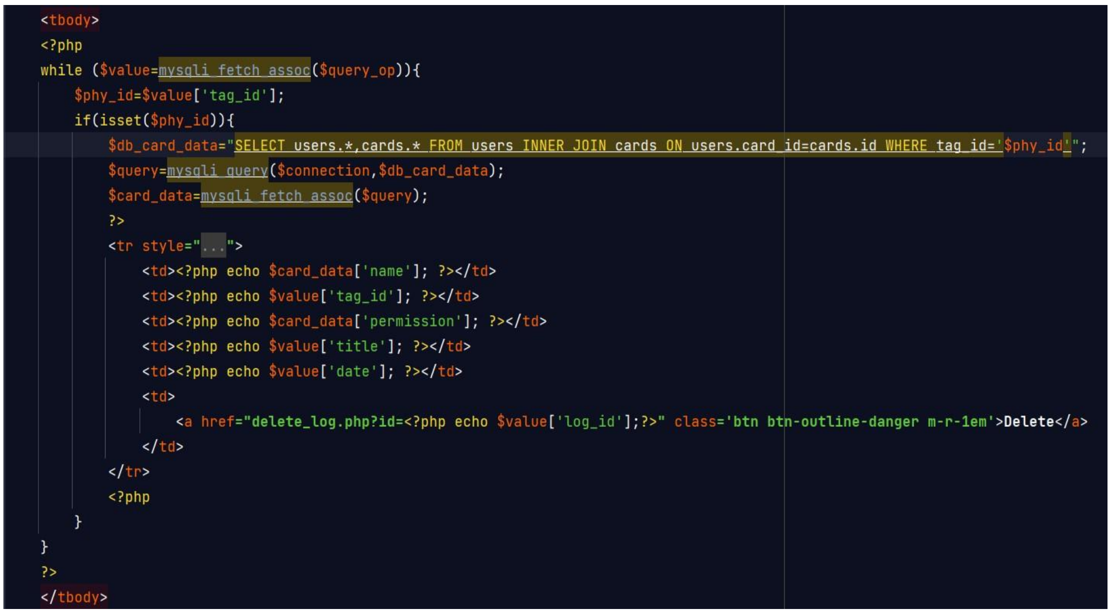


```php
$get_places_num="SELECT id FROM places";
$query2=mysqli_query($connection,$get_places_num);
$places=[];
while($num_arr2=mysqli_fetch_assoc($query2)){
    foreach($num_arr2 as $key => $value){
        $places[]=$key;
    }
}


$get_cards_num="SELECT id,status FROM cards";
$query3=mysqli_query($connection,$get_cards_num);
$active_cards=[];
$passive_cards=[];
while($num_arr3=mysqli_fetch_assoc($query3)){
    foreach($num_arr3 as $key => $value){
        if($key=='status'){
            if($value=='active'){
                array_push($active_cards,$key);
            }elseif($value=='passive'){
                array_push($passive_cards,$key);
            }
        }
    }
}



$db_users_cards_logs_dt="SELECT logs.*,logs.id AS log_id,places.* FROM logs INNER JOIN places ON logs.place_id=places.id";
$query_op=mysqli_query($connection,$db_users_cards_logs_dt);
```

To show statistics about system we must get data about the whole system from database, so in this sample of code we try to perform MYSQL queries to get these data.

First query to get number of users in our system, second one is to get number of cards either active or passive, last one is to get system logs and places in our system.

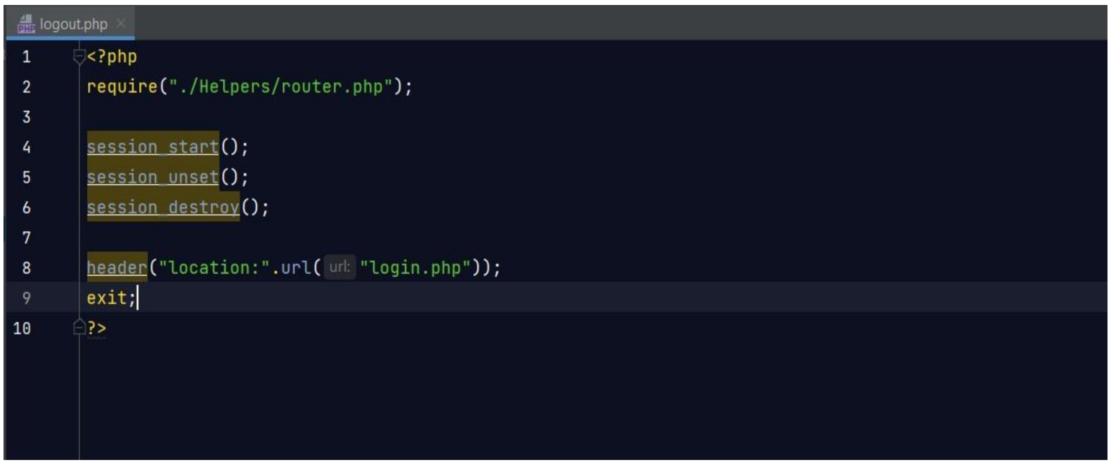
Logs data table



```
<tbody>
<?php
while ($value=mysqli_fetch_assoc($query_op)){
    $phy_id=$value['tag_id'];
    if(isset($phy_id)){
        $db_card_data="SELECT users.*,cards.* FROM users INNER JOIN cards ON users.card_id=cards.id WHERE tag_id='$phy_id'";
        $query=mysqli_query($connection,$db_card_data);
        $card_data=mysqli_fetch_assoc($query);
        ?>
        <tr style="...">
            <td><?php echo $card_data['name']; ?></td>
            <td><?php echo $value['tag_id']; ?></td>
            <td><?php echo $card_data['permission']; ?></td>
            <td><?php echo $value['title']; ?></td>
            <td><?php echo $value['date']; ?></td>
            <td>
                <a href="delete_log.php?id=<?php echo $value['log_id'];?>" class='btn btn-outline-danger m-r-1em'>Delete</a>
            </td>
        </tr>
        <?php
    }
}
?>
</tbody>
```

Here we try to get data from MYSQL object by using **mysql_fetch_assoc()** then will save the return value which will be an array to the variable "$value", finally we will use the values from this array to show them in the data table as shown int the picture.

Logout code sample



logout.php

```
<?php
require("./Helpers/router.php");

session_start();
session_unset();
session_destroy();

header("location:".url( url: "login.php"));
exit;
?>
```

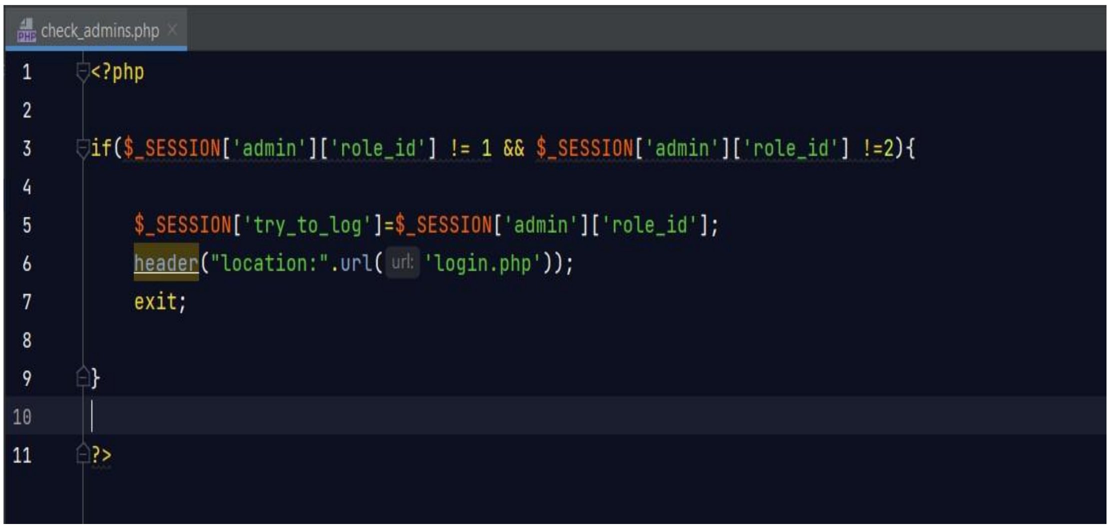
This is the final step in the dashboard user cycle, as we can see there are some built-in functions **session_start()**, **session_unset()**, **session_destroy()**, these built-in function are responsible for terminating user session. Finally, user will be directed to the login page using built-in function **header()**.

Helpers

Helpers as it shown from its name are powerful scripts to help other main scripts, some of them used for checking authority for accessing some main pages, and other used for applying validation and so on.

Here we will dive into some details about these powerful helper scripts…

Check admins



check_admins.php

```php
<?php

if($_SESSION['admin']['role_id'] != 1 && $_SESSION['admin']['role_id'] !=2){

    $_SESSION['try_to_log']=$_SESSION['admin']['role_id'];
    header("location:".url( url: 'login.php'));
    exit;

}

?>
```

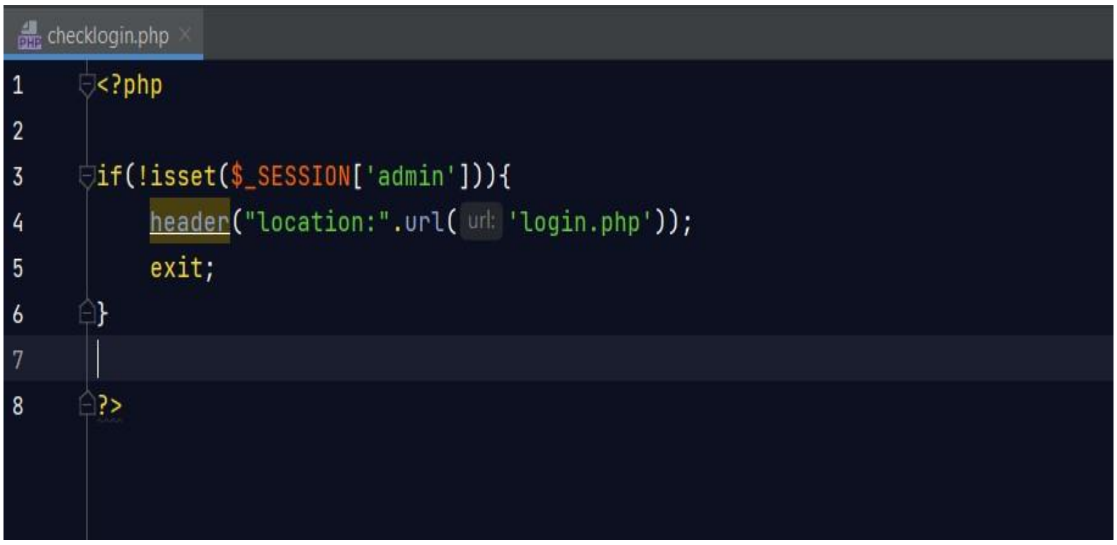
Before we explain this code, I should remind where this current admin data come from?

From the login page when the admin login to the Dashboard his data saved to the powerful Super Global **SESSION** with name “admin”.

Here we call this session to get admin data and try to check if the current system user is not a Super Admin or an admin.

If the condition is true, then error message must be shown to the user and the current system user must be redirected to the login page

Check login

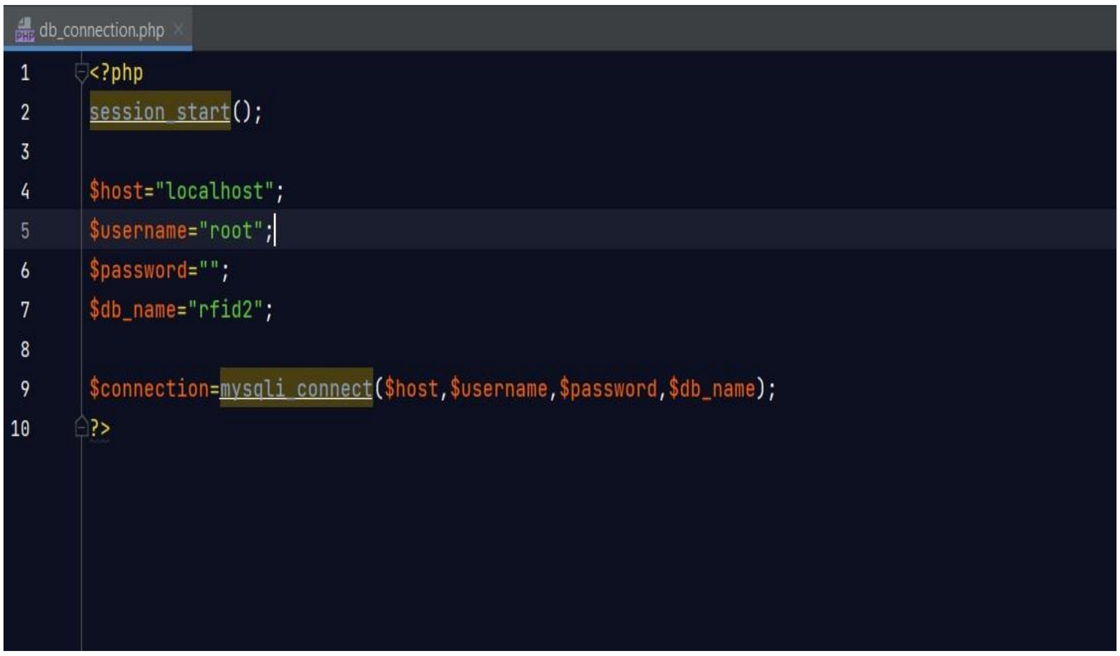


```php
<?php

if(!isset($_SESSION['admin'])){
    header("location:".url( url: 'login.php'));
    exit;
}

?>
```

Here we try to check if the SESSION with name "admin" is existing or not. And we will see if the condition is true automatically user will be redirected to the login page, then what is the benefit of this attribute??

This is one of the most important attributes we use it to control access to all Dashboard pages.

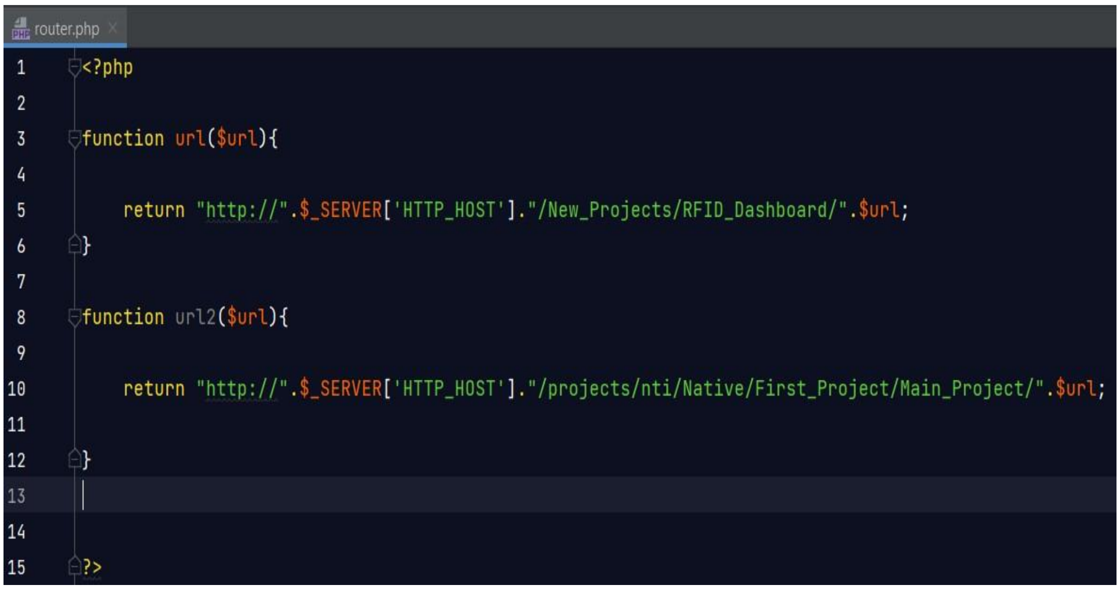
Database Connection



```php
<?php
session_start();

$host="localhost";
$username="root";
$password="";
$db_name="rfid2";

$connection=mysqli_connect($host,$username,$password,$db_name);
?>
```

Here first thing we did is starting SESSION using built-in function **session_start()**, then we prepare database connection requirements to put them as parameters in the built-in function **mysqli_connect()** which is responsible for connection to database server

Router Functions



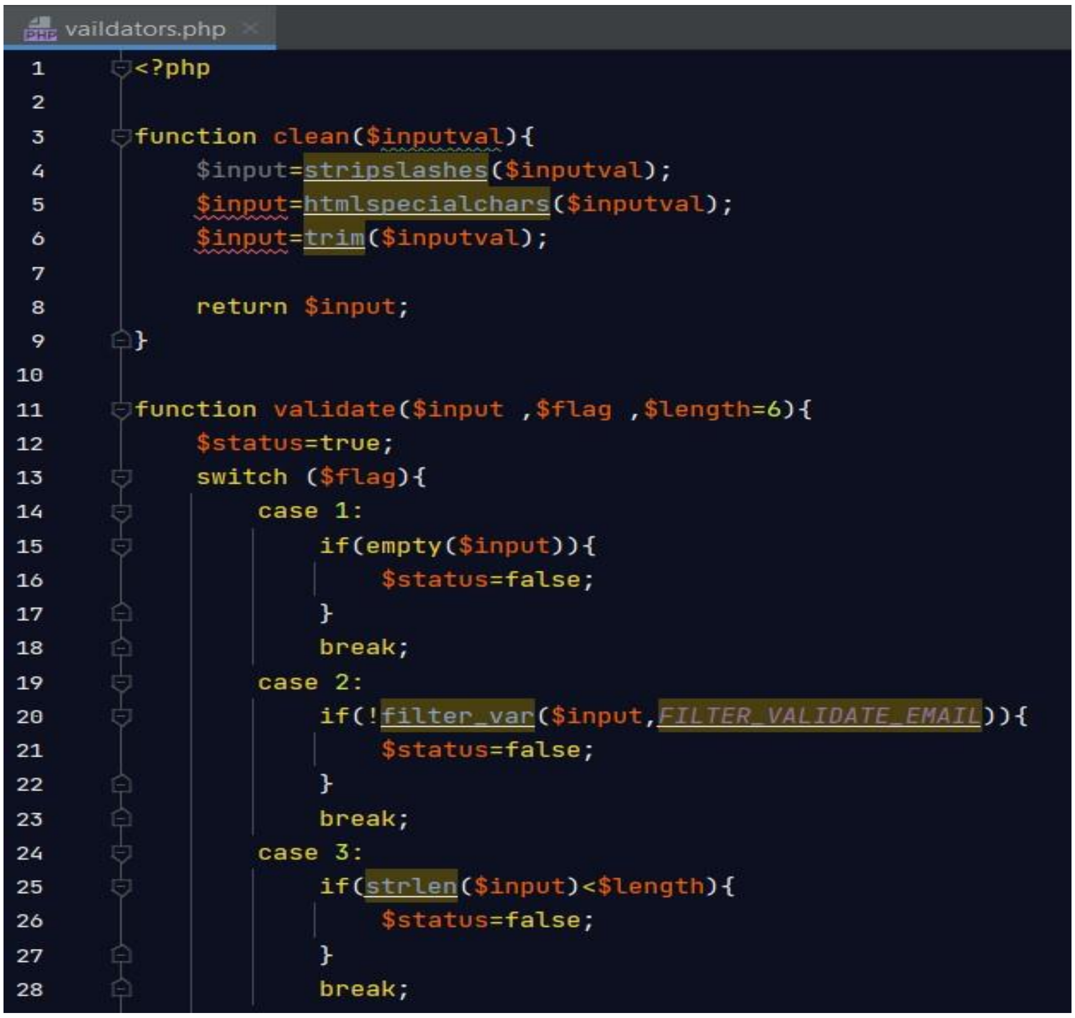
router.php

```php
<?php

function url($url){

    return "http://".$_SERVER['HTTP_HOST']."/New_Projects/RFID_Dashboard/".$url;
}

function url2($url){

    return "http://".$_SERVER['HTTP_HOST']."/projects/nti/Native/First_Project/Main_Project/".$url;

}


?>
```

Here we have built 2 functions, both are used to short paths.

IN details: function will receive file name as a parameter then will concatenate this file name to the complete main path.

Validation rules



vaildators.php

```php
<?php

function clean($inputval){
        $input=stripslashes($inputval);
        $input=htmlspecialchars($inputval);
        $input=trim($inputval);

        return $input;
}

function validate($input ,$flag ,$length=6){
        $status=true;
        switch ($flag){
            case 1:
                if(empty($input)){
                    $status=false;
                }
                break;
            case 2:
                if(!filter_var($input,FILTER_VALIDATE_EMAIL)){
                    $status=false;
                }
                break;
            case 3:
                if(strlen($input)<$length){
                    $status=false;
                }
                break;
```

Here we have 2 user defined functions, by making these two functions we achieve some security measures as we can see, first function “**Clean**” we try to clean the input text from any strange characters by applying three built-in functions [**stripslashes()-htmlspecialchars()-trim()**],

Second function “**Validate**” we try to apply the most powerful filters to the input text:

This function will receive 3 parameters first: input text, second: flag (refers to the filter case number), third: length (refers to the length of the string), after receiving these params the filter with number of the flag number will be applied.

Filters:

1. check if the input text has value or not using **empty()** built-in function.
2. check the validity of the email body using **filter_var()** built-in function.
3. check the length of the string using **strlen()** built-in function.

And some other great filters to raise up the security on the input text.

•Admins

In this part we will show how admin control the whole system parts.

Admin index page

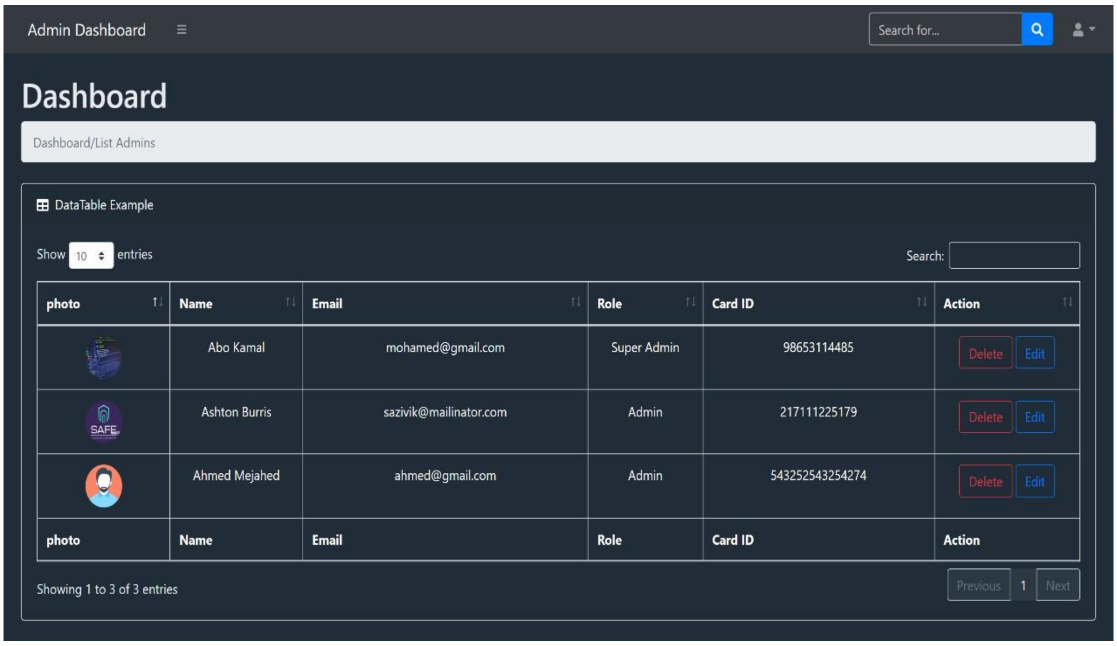
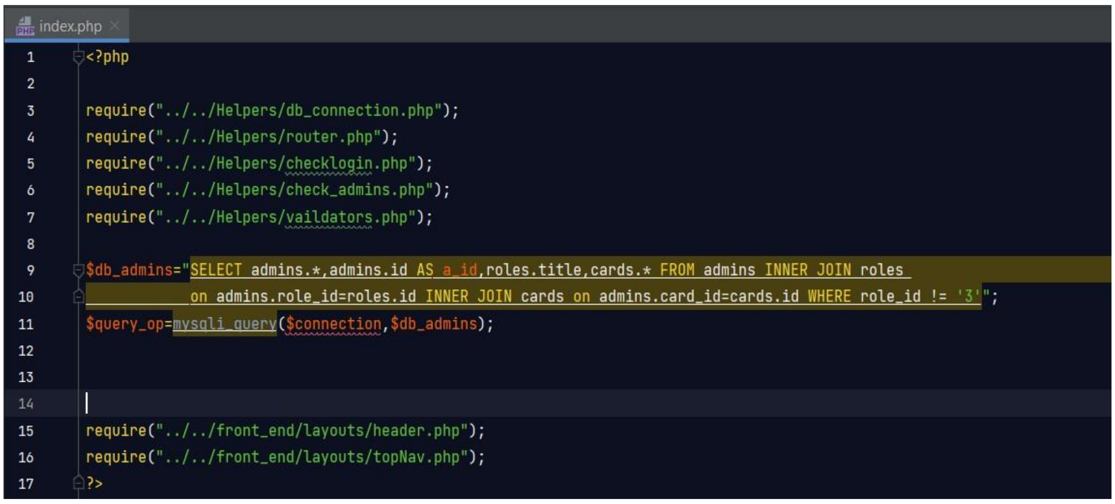


photo	Name	Email	Role	Card ID	Action
	Abo Kamal	mohamed@gmail.com	Super Admin	98653114485	Delete Edit
	Ashton Burris	sazivik@mailinator.com	Admin	217111225179	Delete Edit
	Ahmed Mejahed	ahmed@gmail.com	Admin	543252543254274	Delete Edit
photo	Name	Email	Role	Card ID	Action

Showing 1 to 3 of 3 entries

As shown in this picture, all admins’ data inserted in the detailed data table, it can be deleted or updated and so on

Admin index code

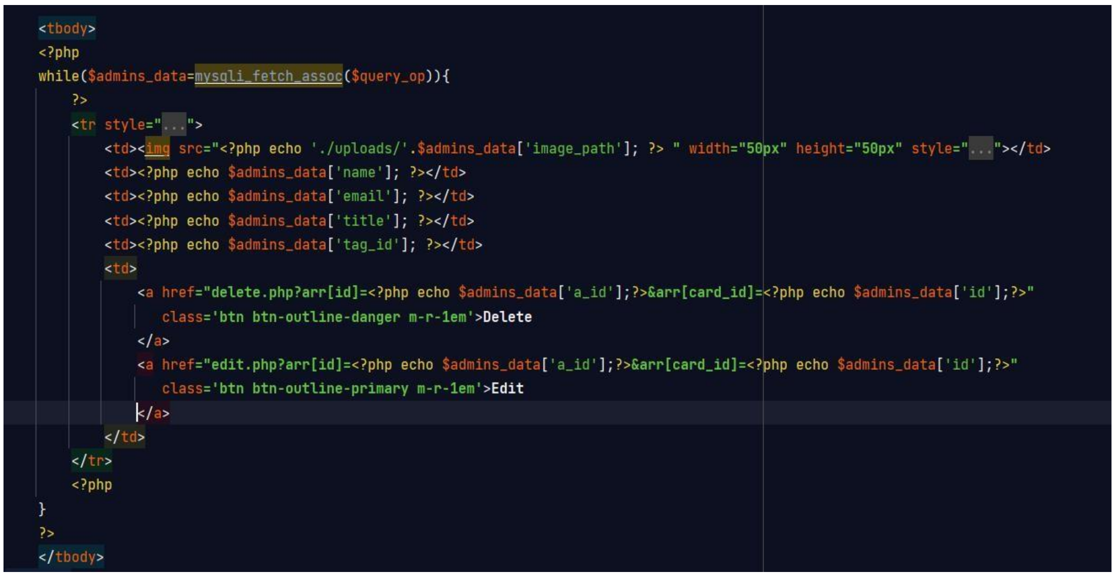


index.php

```
<?php

require("../../Helpers/db_connection.php");
require("../../Helpers/router.php");
require("../../Helpers/checklogin.php");
require("../../Helpers/check_admins.php");
require("../../Helpers/vaildators.php");

$db_admins="SELECT admins.*,admins.id AS a_id,roles.title,cards.* FROM admins INNER JOIN roles
            on admins.role_id=roles.id INNER JOIN cards on admins.card_id=cards.id WHERE role_id != '3'";
$query_op=mysqli_query($connection,$db_admins);



require("../../front_end/layouts/header.php");
require("../../front_end/layouts/topNav.php");
?>
```

In this code sample we have called all helper scripts then we try to get admins data from different tables from database to be shown to the dashboard user.

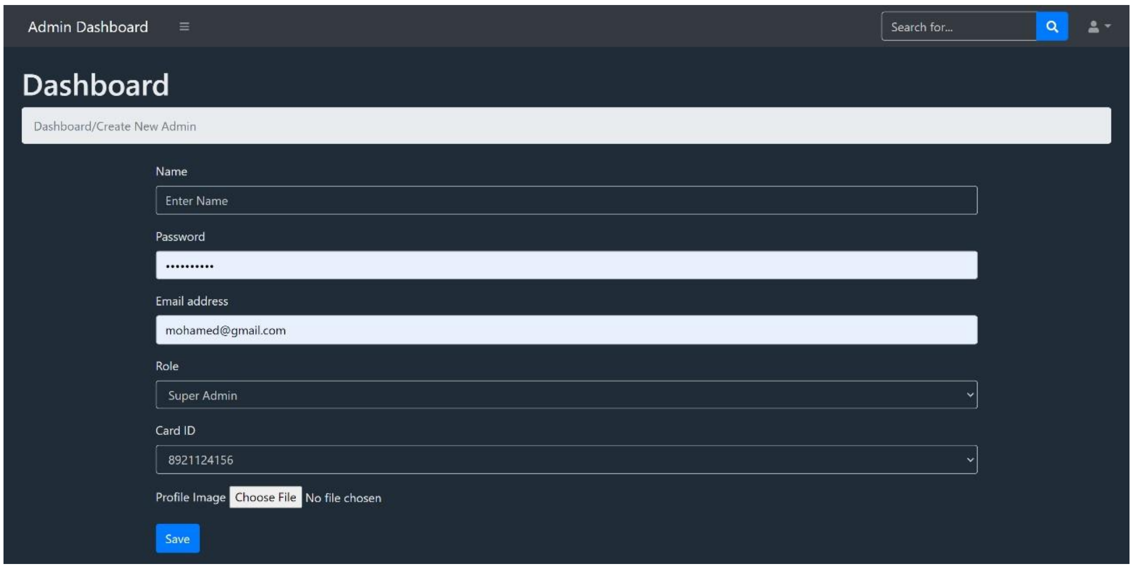
Admins data table



```
<tbody>
<?php
while($admins_data=mysqli_fetch_assoc($query_op)){
    ?>
    <tr style="...">
        <td><img src="<?php echo './uploads/'.$admins_data['image_path']; ?> " width="50px" height="50px" style="..."></td>
        <td><?php echo $admins_data['name']; ?></td>
        <td><?php echo $admins_data['email']; ?></td>
        <td><?php echo $admins_data['title']; ?></td>
        <td><?php echo $admins_data['tag_id']; ?></td>
        <td>
            <a href="delete.php?arr[id]=<?php echo $admins_data['a_id'];?>&arr[card_id]=<?php echo $admins_data['id'];?>"
               class='btn btn-outline-danger m-r-1em'>Delete
            </a>
            <a href="edit.php?arr[id]=<?php echo $admins_data['a_id'];?>&arr[card_id]=<?php echo $admins_data['id'];?>"
               class='btn btn-outline-primary m-r-1em'>Edit
            </a>
        </td>
    </tr>
    <?php
}
?>
</tbody>
```

As we can see in this picture we get data from MySQL object using builtin function **mysql_fetch_assoc((** then the return value of this function will be an associative array with key and value, we will access array values to be shown in the data table.

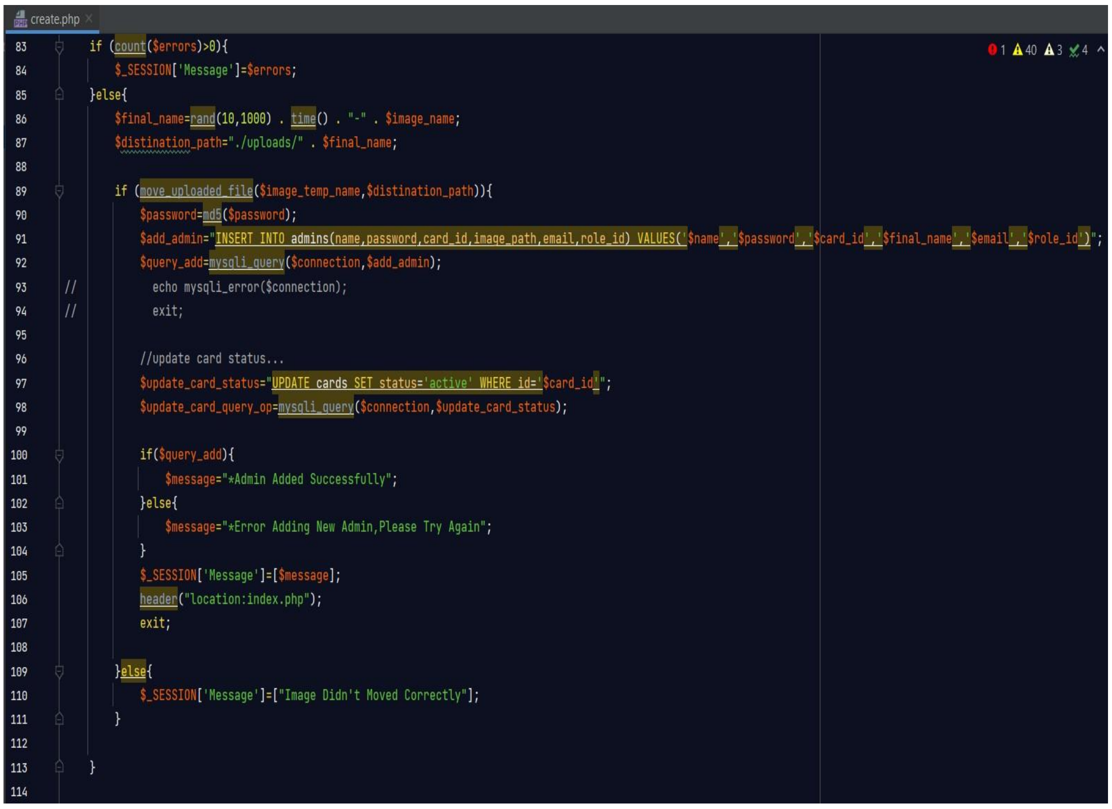
Add new admin form



Name: Enter Name
Password: ••••••••••
Email address: mohamed@gmail.com
Role: Super Admin
Card ID: 8921124156
Profile Image: No file chosen

Here is a form to add new admin to the system, it requires some important information about admin like name, email, role, card id, and personal image.

Add new admin form



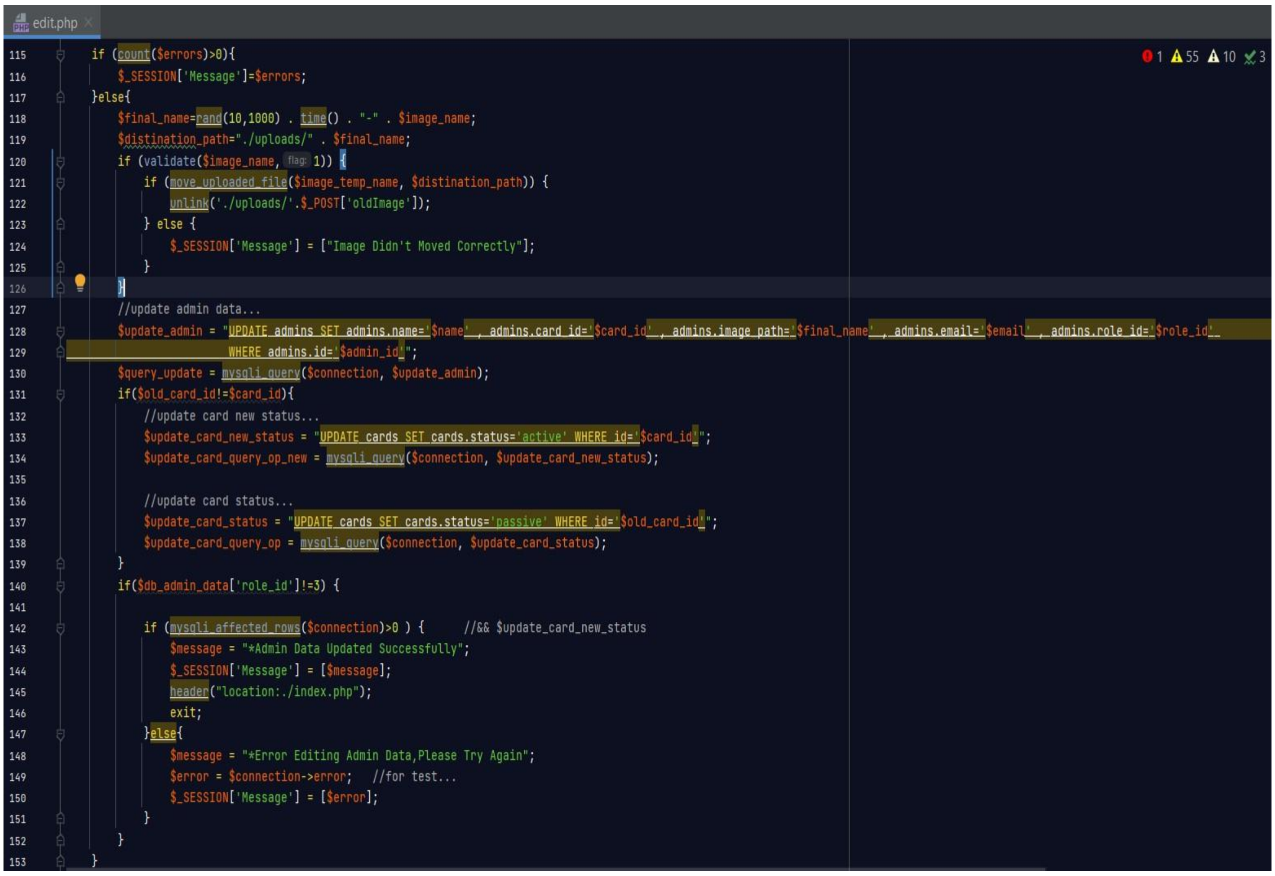
```php
    if (count($errors)>0){
        $_SESSION['Message']=$errors;
    }else{
        $final_name=rand(10,1000) . time() . "-" . $image_name;
        $distination_path="./uploads/" . $final_name;

        if (move_uploaded_file($image_temp_name,$distination_path)){
            $password=md5($password);
            $add_admin="INSERT INTO admins(name,password,card_id,image_path,email,role_id) VALUES('$name','$password','$card_id','$final_name','$email','$role_id')";
            $query_add=mysqli_query($connection,$add_admin);
//            echo mysqli_error($connection);
//            exit;

            //update card status...
            $update_card_status="UPDATE cards SET status='active' WHERE id='$card_id'";
            $update_card_query_op=mysqli_query($connection,$update_card_status);

            if($query_add){
                $message="*Admin Added Successfully";
            }else{
                $message="*Error Adding New Admin,Please Try Again";
            }
            $_SESSION['Message']=[$message];
            header("location:index.php");
            exit;

        }else{
            $_SESSION['Message']=["Image Didn't Moved Correctly"];
        }

    }

```

Here as shown there is a code sample for adding new admin to the system.

First, we try to check if there is any error coming from validation rules, if there is an error, then it will be appeared to the dashboard user as an error message. If there is not any error, then it will go on.

Second, we will perform data processing: move uploaded image to a specific folder, hashing new admin password, insert new admin data into database, update card status from passive to active, finally we will check if all these data processes have been completed successfully or not

Update admin data



edit.php

```
if (count($errors)>0){
    $_SESSION['Message']=$errors;
}else{
    $final_name=rand(10,1000) . time() . "-" . $image_name;
    $distination_path="./uploads/" . $final_name;
    if (validate($image_name, flag: 1)) {
        if (move_uploaded_file($image_temp_name, $distination_path)) {
            unlink('./uploads/'.$_POST['oldImage']);
        } else {
            $_SESSION['Message'] = ["Image Didn't Moved Correctly"];
        }
    }
    //update admin data...
    $update_admin = "UPDATE admins SET admins.name='$name' , admins.card_id='$card_id' , admins.image_path='$final_name' , admins.email='$email' , admins.role_id='$role_id' 
                    WHERE admins.id='$admin_id'";
    $query_update = mysqli_query($connection, $update_admin);
    if($old_card_id!=$card_id){
        //update card new status...
        $update_card_new_status = "UPDATE cards SET cards.status='active' WHERE id='$card_id'";
        $update_card_query_op_new = mysqli_query($connection, $update_card_new_status);

        //update card status...
        $update_card_status = "UPDATE cards SET cards.status='passive' WHERE id='$old_card_id'";
        $update_card_query_op = mysqli_query($connection, $update_card_status);
    }
    if($db_admin_data['role_id']!=3) {

        if (mysqli_affected_rows($connection)>0 ) {      //&& $update_card_new_status
            $message = "*Admin Data Updated Successfully";
            $_SESSION['Message'] = [$message];
            header("location:./index.php");
            exit;
        }else{
            $message = "*Error Editing Admin Data,Please Try Again";
            $error = $connection->error;   //for test...
            $_SESSION['Message'] = [$error];
        }
    }
}
```

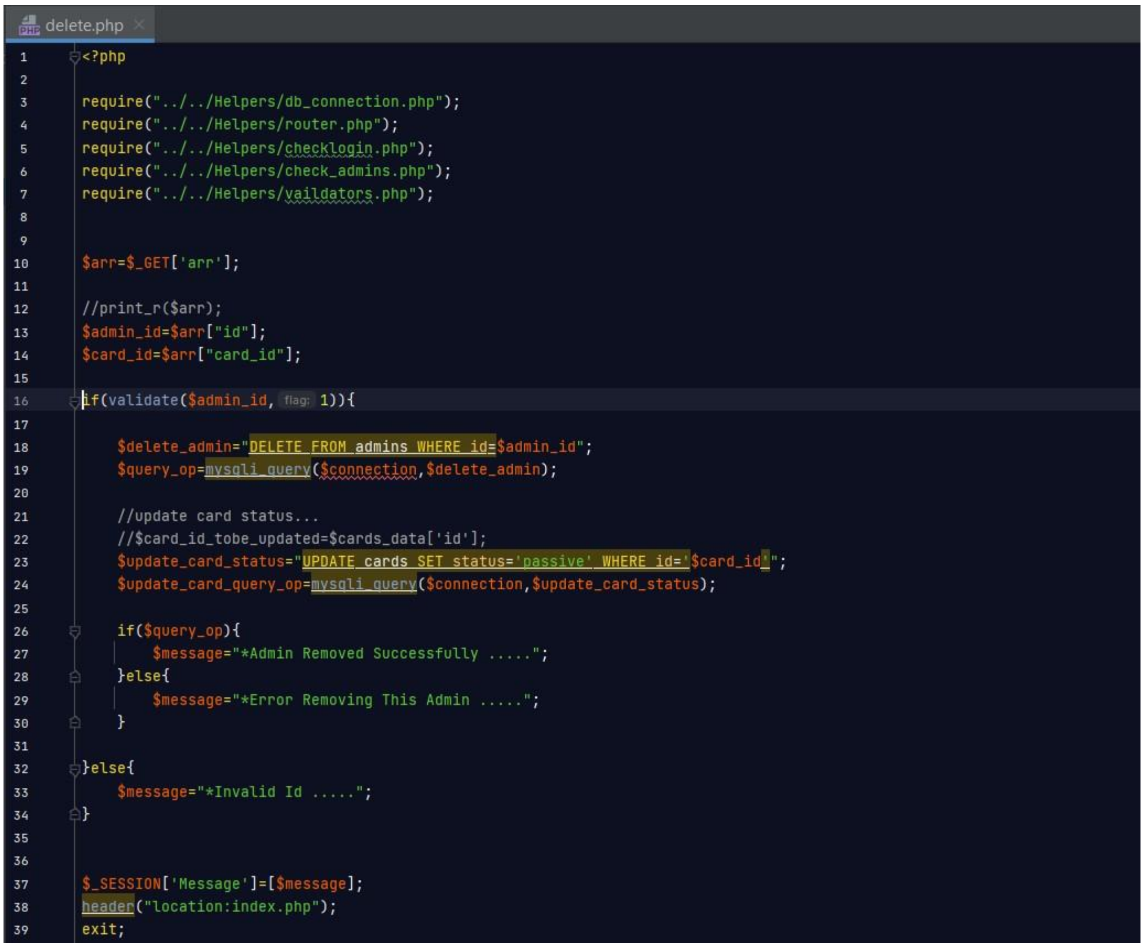
Here we will perform some data processing to update admin data.

First, we will upload new image to the admin uploads folder then the old image must be deleted.

Second, we will perform MySQL query to update data in the database and if the card id has been updated the new chosen card status must be updated to ‘active’, and the old one must be changed to ‘passive’.

Finally, we will check if all these data processes have been completed or not.

Delete admin



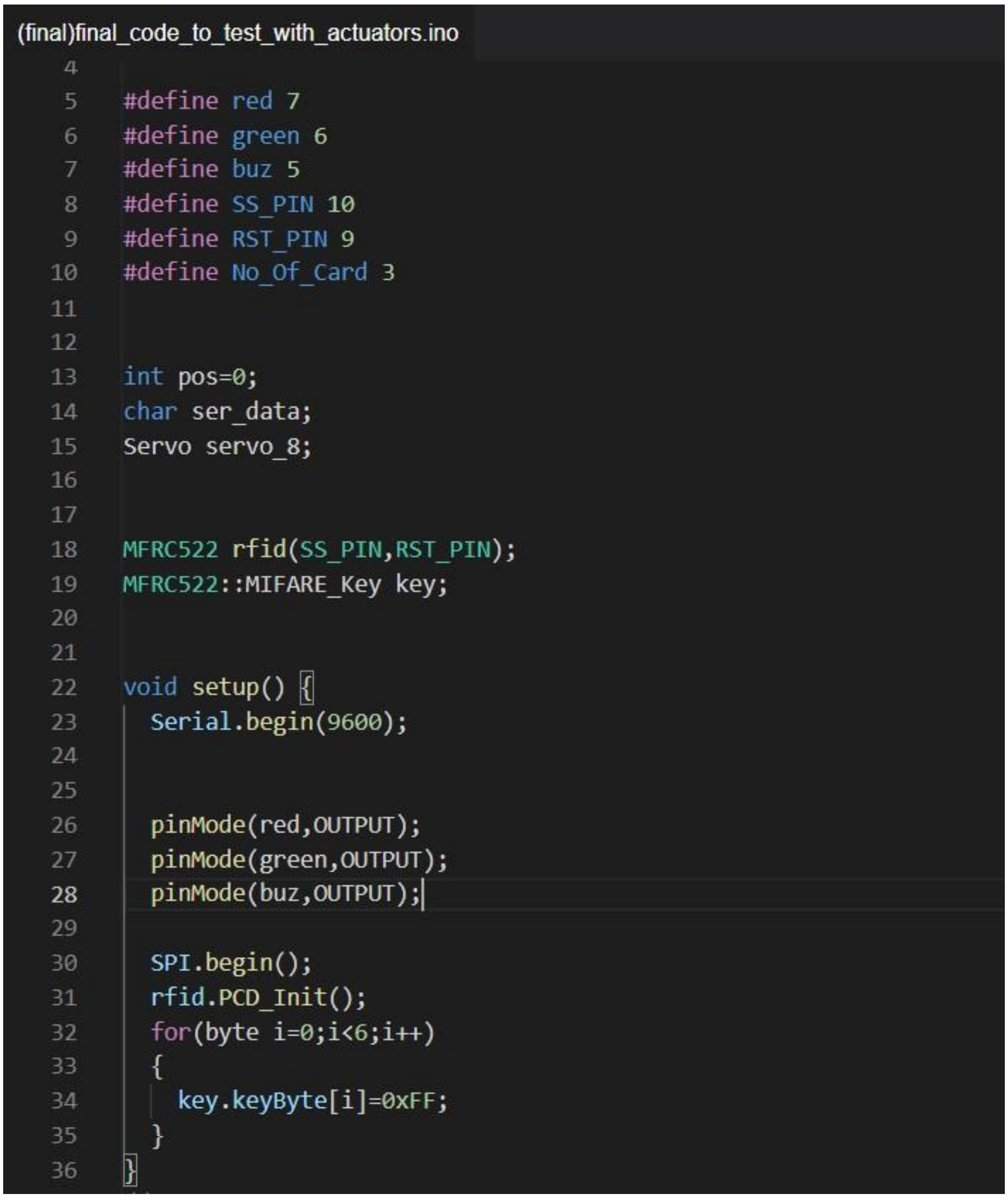
```php
<?php

require("../../Helpers/db_connection.php");
require("../../Helpers/router.php");
require("../../Helpers/checklogin.php");
require("../../Helpers/check_admins.php");
require("../../Helpers/vaildators.php");


$arr=$_GET['arr'];

//print_r($arr);
$admin_id=$arr["id"];
$card_id=$arr["card_id"];

if(validate($admin_id, flag: 1)){

    $delete_admin="DELETE FROM admins WHERE id=$admin_id";
    $query_op=mysqli_query($connection,$delete_admin);

    //update card status...
    //$card_id_tobe_updated=$cards_data['id'];
    $update_card_status="UPDATE cards SET status='passive' WHERE id='$card_id'";
    $update_card_query_op=mysqli_query($connection,$update_card_status);

    if($query_op){
        $message="*Admin Removed Successfully .....";
    }else{
        $message="*Error Removing This Admin .....";
    }

}else{
    $message="*Invalid Id .....";
}


$_SESSION['Message']=[$message];
header("location:index.php");
exit;
```

From the previous page there is a delete button, after clicking on this button the admin id will be send as a value in a get method, according to that here we will receive this id then perform MYSQL query to delete admin data from database

• Arduino code samples

Here we will show Arduino code samples to know how Arduino work.

Predefined and Setup



(final)final_code_to_test_with_actuators.ino

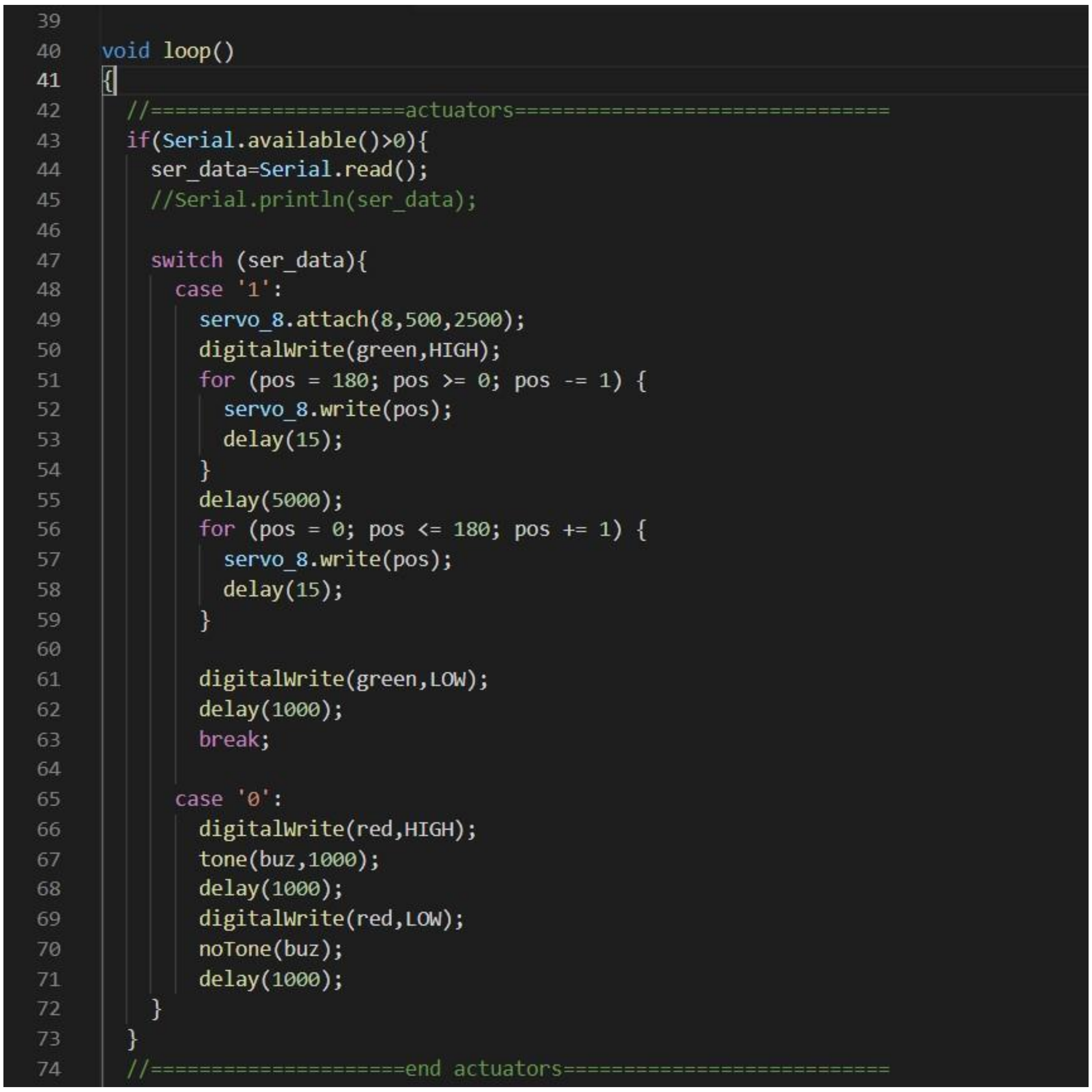
```

#define red 7
#define green 6
#define buz 5
#define SS_PIN 10
#define RST_PIN 9
#define No_Of_Card 3


int pos=0;
char ser_data;
Servo servo_8;


MFRC522 rfid(SS_PIN,RST_PIN);
MFRC522::MIFARE_Key key;


void setup() {
  Serial.begin(9600);


  pinMode(red,OUTPUT);
  pinMode(green,OUTPUT);
  pinMode(buz,OUTPUT);

  SPI.begin();
  rfid.PCD_Init();
  for(byte i=0;i<6;i++)
  {
    key.keyByte[i]=0xFF;
  }
}
```

Here as we can see we have defined some variable to specific pins for example, we specified pin number 7 to variable with name “red” and so on.

We also called and initialized **MFRCC522** which is responsible for initializing **RFID** reader.

Second part, we initialize leds using built-in function **pinMode()** to be used in the loop part, then we initialize **RFID** reader to receive serial number of any card close to it.

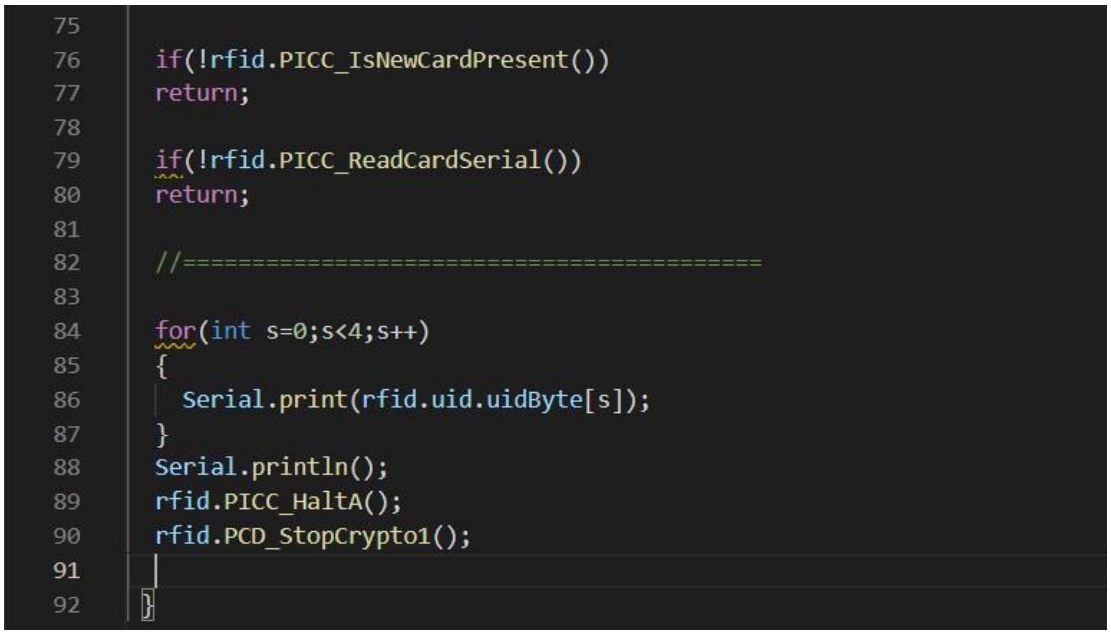
Loop part



```

void loop()
{
  //===================actuators=============================
  if(Serial.available()>0){
    ser_data=Serial.read();
    //Serial.println(ser_data);

    switch (ser_data){
      case '1':
        servo_8.attach(8,500,2500);
        digitalWrite(green,HIGH);
        for (pos = 180; pos >= 0; pos -= 1) {
          servo_8.write(pos);
          delay(15);
        }
        delay(5000);
        for (pos = 0; pos <= 180; pos += 1) {
          servo_8.write(pos);
          delay(15);
        }

        digitalWrite(green,LOW);
        delay(1000);
        break;

      case '0':
        digitalWrite(red,HIGH);
        tone(buz,1000);
        delay(1000);
        digitalWrite(red,LOW);
        noTone(buz);
        delay(1000);
    }
  }
  //===================end actuators=========================
```

In this part we check if the serial has any data or not which considered as a flag coming from python script as a flag, if data is available then we will check if it equal “1” the actuators [green led & servo motor] will begin working, and if the available data equal “0” the actuators [res led & buzzer] will begin working.

Final step



```
if(!rfid.PICC_IsNewCardPresent())
return;

if(!rfid.PICC_ReadCardSerial())
return;

//==========================================

for(int s=0;s<4;s++)
{
  Serial.print(rfid.uid.uidByte[s]);
}
Serial.println();
rfid.PICC_HaltA();
rfid.PCD_StopCrypto1();

}
```

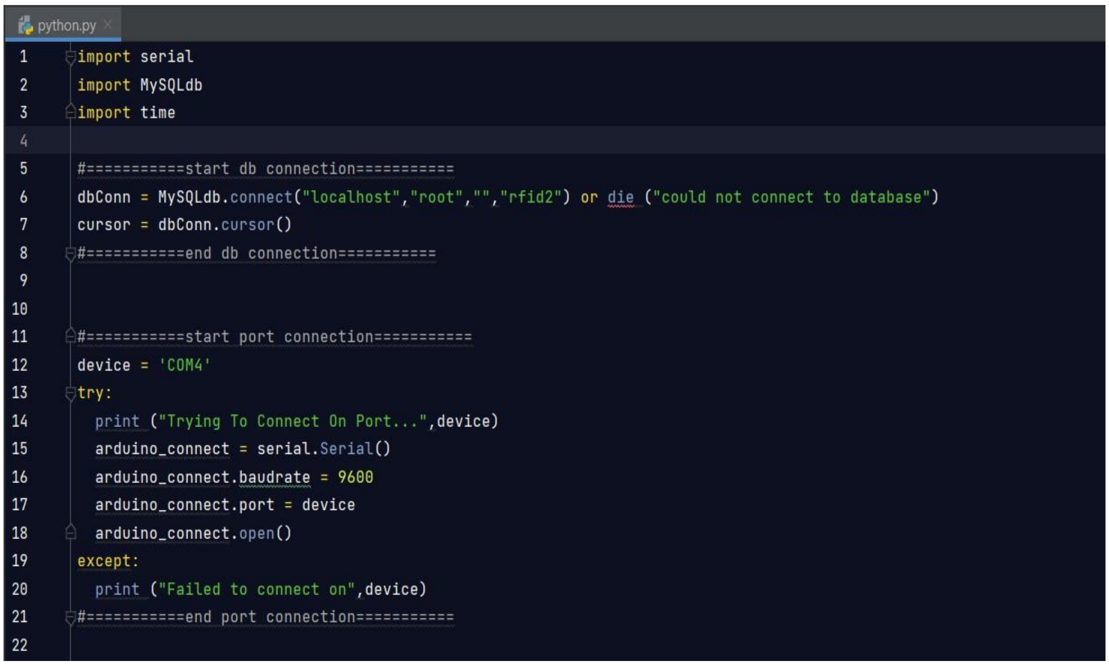
Here we tell the reader if there are no cards close to it, then do nothing.

Finally, we will send card serial number to the serial communication using built-in function **Serial.print()**, then we will terminate RFID using built-in function **rfid.PICC_HaltA()**.

• Python script

This part can be considered as middle layer between hardware and dashboard.

Initialize python code



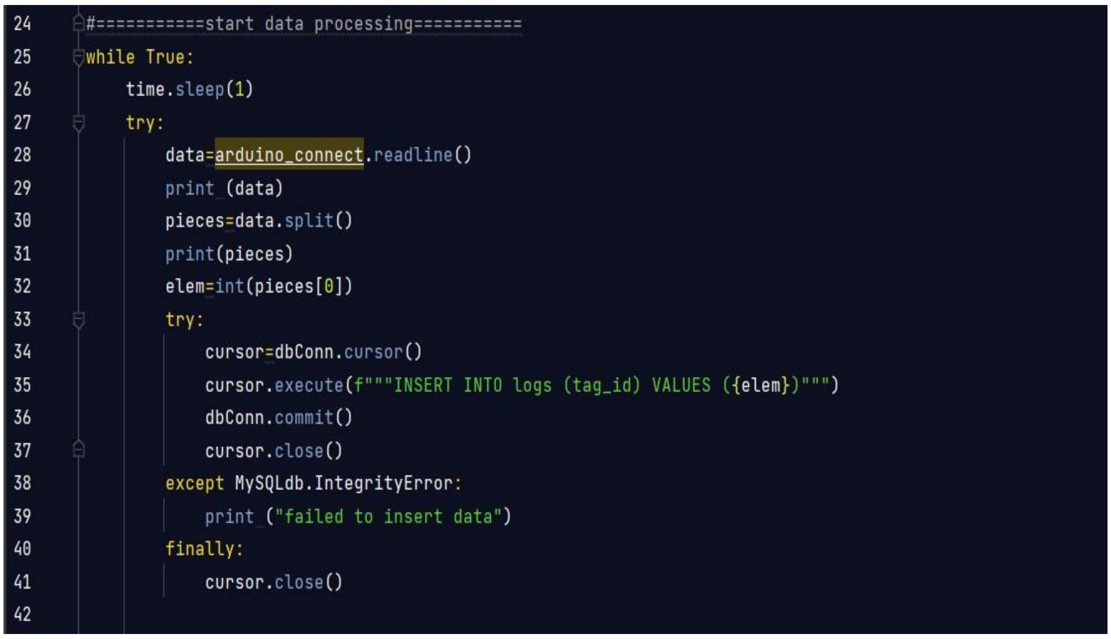
```
import serial
import MySQLdb
import time

#===========start db connection===========
dbConn = MySQLdb.connect("localhost","root","","rfid2") or die ("could not connect to database")
cursor = dbConn.cursor()
#===========end db connection==========


#===========start port connection===========
device = 'COM4'
try:
  print ("Trying To Connect On Port...",device)
  arduino_connect = serial.Serial()
  arduino_connect.baudrate = 9600
  arduino_connect.port = device
  arduino_connect.open()
except:
  print ("Failed to connect on",device)
#===========end port connection==========

```

First thing we have called all libraries that we need like **serial**, **MySQLdb**, and **time**.

Second, we have begun opening connection channel with database server using function **connect()**, also we have initialized cursor using function **cursor()** which is responsible for executing database queries.

In this part finally, we will try to connect with serial using **serial()** function

Insert nto DB logs table

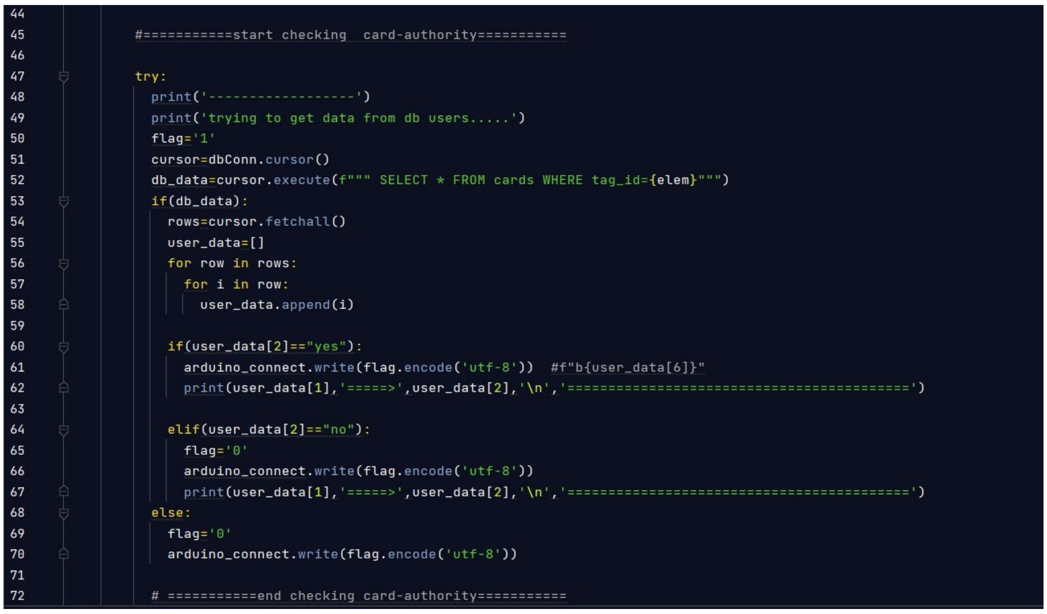


```python
#===========start data processing==========
while True:
    time.sleep(1)
    try:
        data=arduino_connect.readline()
        print (data)
        pieces=data.split()
        print(pieces)
        elem=int(pieces[0])
        try:
            cursor=dbConn.cursor()
            cursor.execute(f"""INSERT INTO logs (tag_id) VALUES ({elem})""")
            dbConn.commit()
            cursor.close()
        except MySQLdb.IntegrityError:
            print ("failed to insert data")
        finally:
            cursor.close()

```

To enforce python script to listen for any new data we use while loop, then we try to read data from serial if it is available.

If data is available we get it using function **arduino_connect()** then insert it into database.

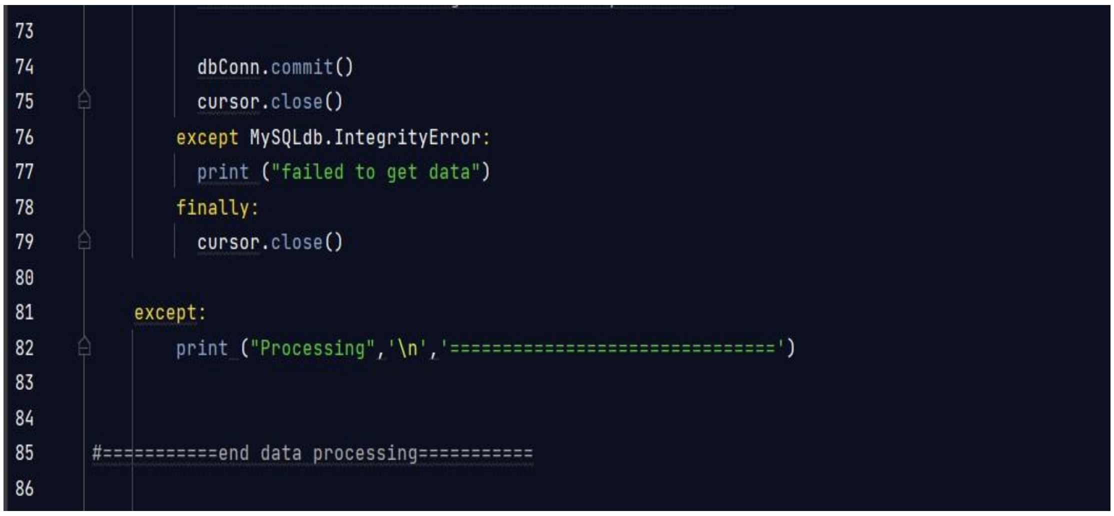
Checking card authority



```python

        #===========start checking  card-authority==========

        try:
          print('-----------------')
          print('trying to get data from db users.....')
          flag='1'
          cursor=dbConn.cursor()
          db_data=cursor.execute(f""" SELECT * FROM cards WHERE tag_id={elem}""")
          if(db_data):
            rows=cursor.fetchall()
            user_data=[]
            for row in rows:
              for i in row:
                user_data.append(i)

            if(user_data[2]=="yes"):
              arduino_connect.write(flag.encode('utf-8'))  #f"b{user_data[6]}"
              print(user_data[1],'=====>',user_data[2],'\n','==========================================')

            elif(user_data[2]=="no"):
              flag='0'
              arduino_connect.write(flag.encode('utf-8'))
              print(user_data[1],'=====>',user_data[2],'\n','==========================================')
          else:
            flag='0'
            arduino_connect.write(flag.encode('utf-8'))

          # ===========end checking card-authority==========
```

The code is executed to verify the user's permissions, as the user's card data is placed in the database, which contains the tag id, the user's permissions, and the user's status if it is active or passive. Lines 48 and 49 are opening sentences, line 50 a variable is set with a value of true until it is confirmed. On line 51, a connection to the database is opened, in line 52 a query is executed to fetch all the user data. The user data is placed in the array to verify the user's permissions and accordingly, the flag is sent but the encoding of the flag is changed to utf-8 In the event that no data is returned, the value of the flag is changed to false.

Eliminate python code



```
73
74            dbConn.commit()
75            cursor.close()
76        except MySQLdb.IntegrityError:
77            print ("failed to get data")
78        finally:
79            cursor.close()
80
81    except:
82        print ("Processing",'\n','==============================')
83
84
85 #===========end data processing==========
86
```

If the query is not executed, the flag value is changed to false, and the alarm and red light are activated

dbConn.commit() is used to perform the above query cursor. **Close ()** is used to close the connection to the database, and on line 76 to 80 an error is made, And the connection is closed.

5.4 System testing ()

#	Test Case	Expected result	Actual result
1	RFID can read the ID of the card	RFID Reader can read the ID from the card and send it to the serial correctly	success
2	The Python code listens to serial when any data is entered into it	The Python code listens to serial when entering any data to it and checks from the databases if the person has access permission sends 1 and if he does not send 0	success
3	The by serial library is the one that communicates between the serial and python	The by serial library is the one that communicates between the serial and the python, the data is transmitted from the hardware on the serial, and the python can read it, add it to the databases and then check the permissions if yes or no	Success
4	Hardware	If the user passes the card on the RFID Reader, the reader reads the data on the card and sends it to the Arduino, and the Arduino sends it to serial, and Python receives the data and checks the databases and sends 1 or 0 through the ear. Turn on the motor and open the door, and if the flag = 0, it turns on a red light and an alarm sound	Success

#	Test Case	Expected result	Actual result
1	Login with valid email and password	The admin can enter the e-mail and password and successfully log in to the main page, and when any error occurs in the email and password, a message appears showing the error	success
2	Super Admin And Admin	The super admin can control all the properties, but the admin can control all the properties except the rules	success
3	The admin can see the personal account of all users	The admin can see all the personal accounts of all users and can control all the details of the account	success
4	The admin can add a user	The admin can add a user and add a card to him	success
5	The admin can delete a user	An admin can delete a user and delete his card	success
6	The admin can edit a user	The admin can edit the user and edit his card	success
7	The admin can control where people are	The admin can control the presence of people's places and allow them to enter or not	success

5.5 Results

We conclude that the services provided by our system in detail are:

There is a login page that allows the administrator to enter the dashboard page to perform its functions

Determine the identity of people through the dashboard page, through which the administrator can add or remove people, as well as determine the number of people in the organization

The dashboard page consists of

Roles screen

It is the general screen that includes all categories (admins - users)

Administrators screen

This screen allows the main admin (view, add, edit, delete) sub-admins

Users screen

This screen allows the main administrator (view, add, edit, delete) users

Cards screen

This screen allows the main admin (view, edit, delete) cards

Attendance screen

This screen allows the main administrator (view report, add, delete) users

There is a logout function that enables the administrator to exit the dashboard page and return to the login page

5.6 Achieved goal

The main goal that we have reached and achieved in this project so far is to make a small model of the project in the form of a small box that simulates what is being applied in the project in order to illustrate the idea. We put the RFID reader, breadboard, and everything inside the box that includes the hardware. Explain the hardware part in a detailed way because it is explained enough in the hardware part. The Arduino is connected to the laptop and the sign is passed on to the reader to achieve the desired goal and reach the desired goal, which is when the card or card is swiped, the door is opened in the event that the person is allowed to enter inside the institution or the lecture hall, as we have achieved so far, thanks to God, a point that is that the student or the person who Enters the institution Registered when the card is swiped at the reader, this status also appears, this status appears on the dashboard and helps the administrator and affairs personnel determine whether the student has attended or not and from within the institution or hall. This system in general reduces the presence of the human factor and protects it from doing routine things that make security men bored, and this helps security men and everything that works within the organization and protects them from fatigue in obtaining information