Criterion-C

There are several modules I have made in a creative way to meet clients needs with accordance to my limited knowledge. Here is a list those modules:

1. Word divider
2. Room availability
3. Security check to change credentials
4. Verification of data
5. Room booking
6. Creating Employee Accounts
7. Updating all the database tables

1. Word Divider
   Code:

```
  // get room type cost
private void get_RT_cost(String RT)
{
    try
    {
        String sql ="SELECT * FROM `rooms`WHERE `Room_type` = '"+RT+"'";
        PreparedStatement ps = cn.prepareCall(sql);
        ResultSet rs = ps.executeQuery();
        while(rs.next())
        {
            cost_each_room = rs.getInt("cost_per_night");
            array_of_bookedrooms( room_list,rs.getInt("cost_per_night"),RT);
        }
    }
    catch(SQLException e )
    {
        JOptionPane.showMessageDialog(null, e);
    }
}
```

```
//get all the booked rooms and puts them in an array
private void array_of_bookedrooms(String RM,int RT_cost ,String RT)
{

    int num_words = 0;
    // array to store each room
            String t2 ="";
            if(RM.equals("")!= true)
            {
            int y = RM.length();
            // to get the tot number of rooms in booking
            for(int z=0; z<y ; z++)
            {
                String t =RM.substring(z,z+1);
                if(t.equals(","))
                {
                    num_words++;
                }
            }
            String[] list = new String[num_words];
            // array to store each room
            int ct = 0;
            //breaks all the roooms and puts them into the array
            for(int z=0; z<y ; z++)
            {
                String t = RM.substring(z,z+1);

                if(t.equals(","))
                {
                    list[ct] = t2;
                    ct++;
                    t2 ="";
                }
                else
                {
                    t2 = t2+t ;
                }
            }
            set_table(RT, RT_cost, list, num_words);
            }
            else
            {
                String[] list =new String[1];
                list[0]="";
                set_table(RT, RT_cost, list, num_words);
            }
}
```

During the room choosing process where the user needs to choose the rooms for booking. The program needs to know which rooms are booked already in the given span of the current booking that's being registered. So, it can hide them from the user while making the current selection. To do so we need an array of already booked rooms and then compare it to the list with all the rooms and show only the ones that are not in the already booked rooms.

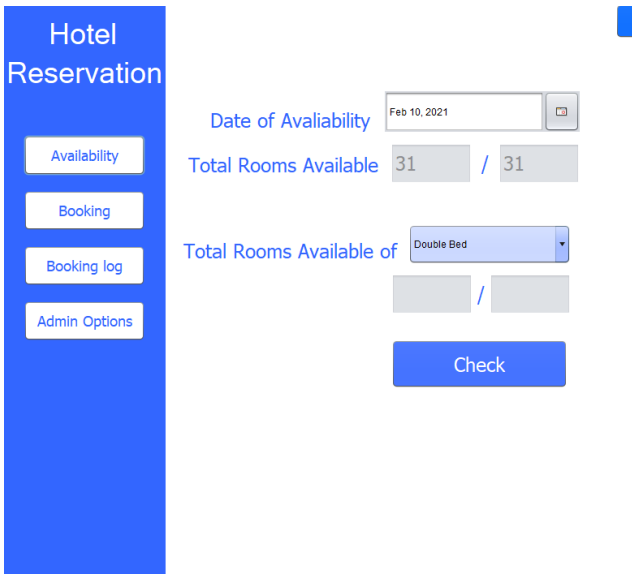
Therefore, to make an array with already booked room first I make a sting that contains all the booked rooms in the current given date span. The rooms are already in string form in the database with a comma after every room number. This will function as an indicator to the program to separate the room numbers individually. The code is iterated twice with a little difference. First it is iterated to get the number of rooms, to initialize the array. Then it is iterated again to separate the words at the comma and insert into the array. This then gives an array with the utility to exclude the rooms in this array from the options of the user. So, no one room can be booked by two different booking instances.

2. Room Availability
   Code:

```
// get total & available rooms and display them
public void get_totRooms()
{
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
    String date  = sdf.format(date_chooser_ava.getDate());
    long millis=System.currentTimeMillis();
    java.sql.Date today =new java.sql.Date(millis);
    String Today = sdf.format(today);
    int booked_rooms=0;
    int tot_rom=0;
    try{
    String sql = "SELECT Count(`room_no`) FROM `each_room`";
        PreparedStatement ps = cn.prepareStatement(sql);
        ResultSet rs = ps.executeQuery();
        while(rs.next())
        {
            tot_rom =Integer.parseInt(rs.getString("Count(`room_no`)"));
            totalRooms_textbox.setText(String.valueOf(tot_rom));
        }
    }
    catch(SQLException e )
    {
        JOptionPane.showMessageDialog(null,"not connected " );
    }
    int OS_rooms =0;
    try{
    String sql = "SELECT Count(`room_no`) FROM `each_room` Where`room_in_out_service` = 'Out of Service' ";
        PreparedStatement ps = cn.prepareStatement(sql);
        ResultSet rs = ps.executeQuery();
        while(rs.next())
        {
            OS_rooms = Integer.parseInt(rs.getString("Count(`room_no`)"));

        }
    }
    catch(SQLException e )
    {
        JOptionPane.showMessageDialog(null,"not connected " );
    }
    try
    {
        String sql = "SELECT * FROM `bookings` ";
        PreparedStatement ps = cn.prepareStatement(sql);
        ResultSet rs = ps.executeQuery();
        while(rs.next())
        {
            String datef = rs.getString("Booked_from");
            String datet = rs.getString("Booked_to");
            boolean ch1 = datef.compareTo(date) <= 0;
            boolean ch2 = datet.compareTo(date) >= 0;
            if(ch1 && ch2){booked_rooms++;}

        }
        int ava= 0 ;
        ava = tot_rom-(booked_rooms+OS_rooms);


        String aval = String.valueOf(ava);

        totalRoomsAvailable_textbox.setText(aval);

    }
     catch(SQLException e )
    {
        JOptionPane.showMessageDialog(null,"not connected " );
    }
}
```

First a date is selected and then it is compared to all the bookings. Then if the selected date is within the span of any bookings then the rooms of the bookings are called. For total number of rooms, the total number of rooms in the data base is set in the app. Then, the rooms available can be found by subtracting the booked rooms and out of service rooms from the total number of rooms. This gives us the in service and not booked rooms.

```
//gets available rooms of each type
private void eachTRoomTypeAva2(String rt,int tot_rom)
{


     SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
    String date  = sdf.format(date_chooser_ava.getDate());
    long millis=System.currentTimeMillis();
    java.sql.Date today =new java.sql.Date(millis);
    String Today = sdf.format(today);
    int booked_rooms=0;
    int OS_rooms= 0;

    // get all out of service rooms
     try{
        String sql = "SELECT Count(`room_no`) FROM `each_room` Where`room_in_out_service` = 'Out of Service' AND`room_type` ='"+rt+"'";
        PreparedStatement ps = cn.prepareStatement(sql);
        ResultSet rs = ps.executeQuery();
        while(rs.next())
        {
            OS_rooms = Integer.parseInt(rs.getString("Count(`room_no`)"));


        }

    }
    catch(SQLException e){ System.out.println("3");}
     // all the booking and send to find which one of them has a room booked of the roomtype we are right now searching of
      try
    {
        String sql = "SELECT * FROM `bookings` ";
        PreparedStatement ps = cn.prepareStatement(sql);
        ResultSet rs = ps.executeQuery();
        String allRooms= "";
        while(rs.next())
        {
            String datef = rs.getString("Booked_from");
            String datet = rs.getString("Booked_to");
            boolean ch1 = datef.compareTo(date) <= 0;
            boolean ch2 = datet.compareTo(date) >= 0;
            System.out.println(ch1+" "+ch2);
            if(ch1 && ch2)
            {
                allRooms = allRooms +rs.getString("Booked_rooms");
            }

        }
        find_room(tot_rom,OS_rooms,allRooms ,rt );


    }
     catch(SQLException e )
    {
        JOptionPane.showMessageDialog(null,"3 " );
    }

}
```



For each room it is a little bit different. Due to the extra constrains that the program needs to check if the room in the booking is from the selected room type. So, the breaking the room code is used here and the array is used to compare and check if the room is from the room type. This is then stored and used in further calculations. The similar concepts like the total rooms are used here also.

```
private void find_room(int tot_room, int out_of_service , String RM, String RT)
{
   int num_words = 0;
    // array to store each room

            String t2 ="";
            System.out.println(RM);
            if(RM.equals("")!= true)
            {
            int y = RM.length();
            // to get the tot number of rooms in booking
            for(int z=0; z<y ; z++)
            {
                String t =RM.substring(z,z+1);
                if(t.equals(","))
                {
                    num_words++;
                }
            }
            String[] list = new String[num_words];
            // array to store each room
            int ct = 0;
            //breaks all the roooms and puts them into the array
            for(int z=0; z<y ; z++)
            {
                String t = RM.substring(z,z+1);

                if(t.equals(","))
                {
                    list[ct] = t2;
                    ct++;
                    t2 ="";
                }
                else
                {
                    t2 = t2+t ;
                }
            }
            int bookedR= 0;

            for(int coc=0; coc<num_words; coc++ ){
                System.out.println(list[coc]);
                try
                {
                   String sql = "SELECT * FROM `each_room` Where `room_no` = '"+list[coc]+"'";
                   PreparedStatement ps = cn.prepareStatement(sql);
                   ResultSet rs = ps.executeQuery();
                   while(rs.next())
                   {
                       String room_t = rs.getString("room_type");
                       if(room_t.equals(RT)){bookedR++;}
                   }

                }
                catch(SQLException e){ System.out.println("3");}
            }
            int ava = tot_room - (out_of_service+bookedR );
            String aval = String.valueOf(ava);
            totalRoomsAvailableOf_textbox.setText(aval);



            }
            else
            {
                totalRoomsAvailableOf_textbox.setText(String.valueOf(tot_room));
            }

}
```

3. Security check to change credentials
   Code:

```
public class permission {

    private boolean permission ;
    public void Set_per(boolean per)
    {
        permission = per;
    }
    public boolean Return()
    {
        return permission ;
    }
}
```

The permission class.

```
    permission pl = new permission();
    public username_forgot() {
        initComponents();
        this.setLocationRelativeTo(null);
        conection();
        empno_get_question.setVisible(true);
        question_ans.setVisible(false);
        Set_Password.setVisible(false);
    }
```

The initialization of the class.

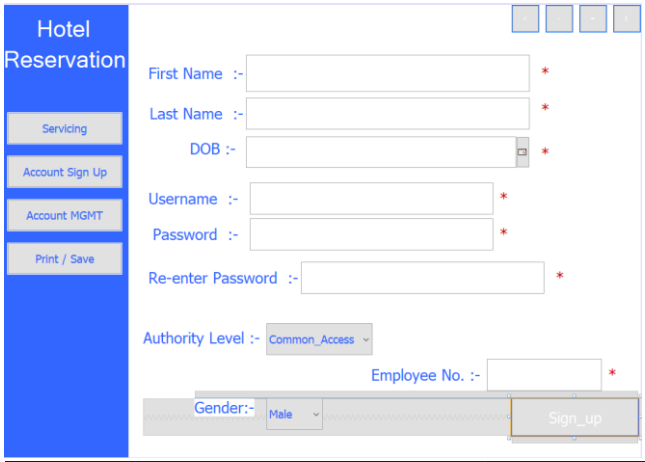
```
// add image of identification that has been scanned
try{
    String sql = "SELECT * FROM `login_employee` WHERE `Emp.no` = '"+empno+"'";
    PreparedStatement ps = cn.prepareStatement(sql);
    ResultSet rs = ps.executeQuery();
    while(rs.next()){
    String ans = rs.getString("Security _Answer");
    if(ans.equals(answer_textbox2.getText()))
    {
        pl.Set_per(true);
        empno_get_question.setVisible(false);
        question_ans.setVisible(false);
        Set_Password.setVisible(true);}
    else{Answer_lable2.setForeground(Color.red);}}
}catch(SQLException e){ System.out.println(e);}
```

Setting the private variable in the class as true.

```
String username = username_textbox.getText();
String reenter = reenter_textbox.getText();
boolean C1 = username.equals("");
boolean C2 = reenter.equals("");
boolean C3 = ...Return();
if(C1){username_lable.setForeground(Color.red);}
if(C2){passre_lable.setForeground(Color.red);}

if(C1== true && C2== true && C3==true)
{

    try{
        String sql = "UPDATE `login_employee` SET `username`='"+username+"'WHERE `Emp.no` ='"+empno+"'";
        PreparedStatement ps = cn.prepareStatement(sql);
        int rs = ps.executeUpdate();
        if(rs >= 0)
        {
            JOptionPane.showMessageDialog(null,"Submited" );
            this.dispose();
        }

    }
    catch(SQLException e) {System.out.println(e);}

}
```

Then using the class's set variable as a verification as a precaution to check if the previous step is done. Thus, it serves as a security check.

4. Verification of Data:



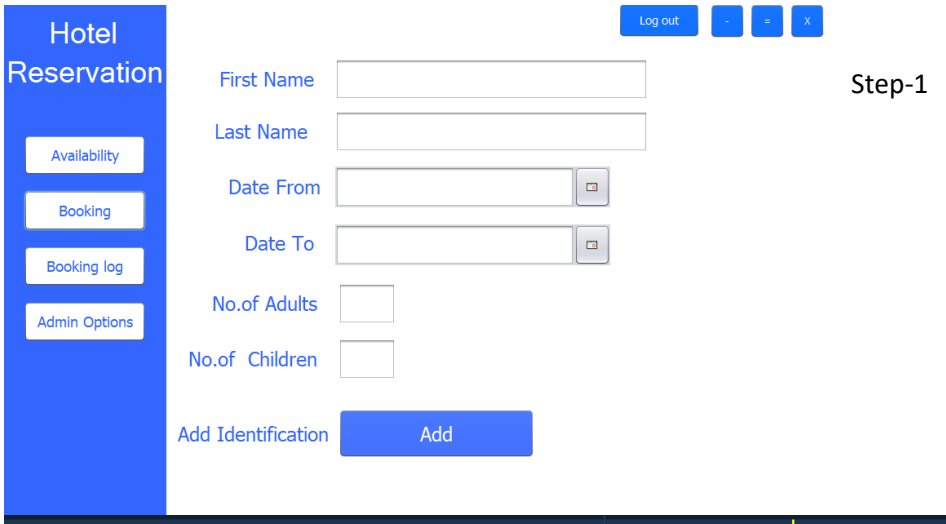
The password is entered once in the password field and then again in the "re-enter password" in order for the user to be sure of the entered password.

```
private void accountMake_buttonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String first_name = "";
    String last_name = "";
    String username = "";
    String password = "";
    String authority_level = "";
    String gender = "";
    String empNo = "";
    String dob = "";
    String passwordRe = "";
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
    first_name = firstName_textbox.getText();
    last_name = lastName_textbox.getText();
    username = username_textbox.getText();
    password = password_textbox.getText();
    authority_level = String.valueOf(authorityLevel_combobox.getSelectedItem());
    gender = String.valueOf(gender_combobox.getSelectedItem());
    empNo = employeeNo_textbox.getText();
    try{
    dob = sdf.format(dob_datechooser.getDate());}
    catch(NullPointerException e){System.out.println(e);}
    passwordRe = passwordRe_textbox.getText();
    long millis=System.currentTimeMillis();
    java.sql.Date today =new java.sql.Date(millis);
    String Today = sdf.format(today);
    String check = "";
    boolean c0 = (dob.compareTo(Today)>0);
    boolean c01 = dob.equals("");
    boolean c1 = first_name.equals(check);
    boolean c2 = last_name.equals(check);
    boolean c3 = username.equals(check);
    boolean c4 = password.equals(check);
    boolean c5 = empNo.equals(check);
    boolean c6 = password.equals(passwordRe);
    if (first_name.equals(check)){error1_lable.setVisible(true);}
    if (last_name.equals(check)){error2_lable.setVisible(true);}
    if (username.equals(check)){error4_lable.setVisible(true);}
    if (password.equals(check)){error5_lable.setVisible(true);}
    if (empNo.equals(check)){error7_lable.setVisible(true);}
    if (c0){error3_lable.setVisible(true);}
    if (c01){error3_lable.setVisible(true);}
    if (c6 == false){error6_lable.setVisible(true);}
    if (c1==false && c2==false && c3==false && c4==false && c5==false && c0 == false && c6 == true )
    {error_disable(); dob_datechooser.setDate(null);
    new Security_questions(first_name, last_name, username, password, authority_level,gender,empNo ,dob,passwordRe, cn).setVisible(true);
    }
}
```

Here is the code for the verification. There is check "C6" which checks the verification of the password.

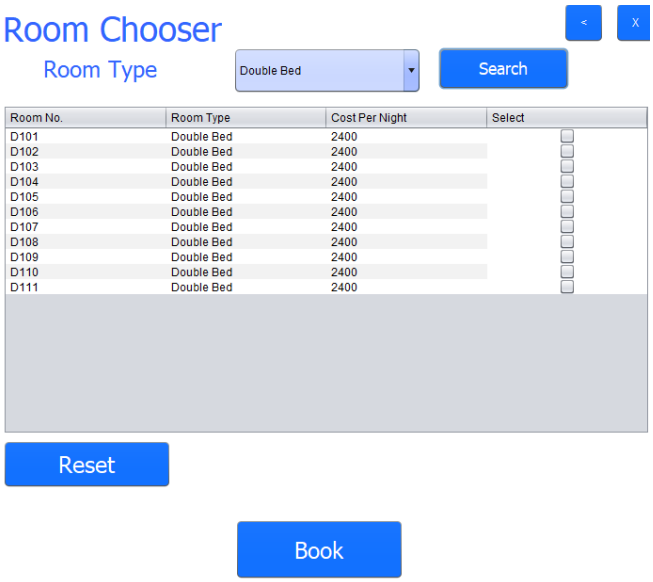
5. Room Booking:

Booking a room happens in three steps. First is to take in customer credentials and ID photos which is then stored in the data base. Second is the selection of the room, the customer is showed the available rooms of the desired room type and they can select from it. Third the payment is made physically and confirmation pop is displayed. (note: the current version of the application does not contain a E-payment method. This is an improvement criterion for later version



Step-1

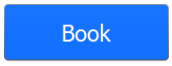
```
private void bookConfirm_buttonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
    //get all the information
    String firstname = firstName_textbox.getText();
    String lastname = lastName_textbox.getText();
    String datefrom = "";
    String dateto = "";
    try{
        datefrom = sdf.format(dateFrom_datechooser.getDate());
        dateto = sdf.format(dateTo_datechooser.getDate());}
    catch(NullPointerException e){System.out.println(e);}
    String num_adult = noOfAdults_textbox.getText();
    String num_children = noFoChildren_textbox.getText();

    // check for missing information
    String c = "";
    long millis=System.currentTimeMillis();
    java.sql.Date today =new java.sql.Date(millis);
    String Today = sdf.format(today);

    boolean c1 = firstname.equals(c);
    boolean c2 = lastname.equals(c);
    boolean c3 = Today.compareTo(datefrom) > 0;
    boolean c4 = Today.compareTo(dateto) > 0;
    boolean c34 = datefrom.compareTo(dateto)>0;
    boolean c33 = dateto.equals("");
    boolean c44 = datefrom.equals("");
    boolean c5 = num_adult.equals(c);
    boolean c6 = num_children.equals(c);
    boolean c7 = image_indentification ==null;
    boolean ct = false;
    if (c1==false && c2==false && c3==false && c4==false && c34==false && c5==false && c6==false && c7 == false ){ct = true;}
    // show according to the checks
    if (c1){error1_lable_book.setVisible(true);}
    if (c2){error2_lable_book.setVisible(true);}
    if (c3){error5_lable_book.setVisible(true);}
    if (c33){error5_lable_book.setVisible(true);}
    if (c4){error5_lable_book.setVisible(true);}
    if (c44){error5_lable_book.setVisible(true);}
    if (c34){error5_lable_book.setVisible(true);}
    if (c5){error3_lable_book.setVisible(true);}
    if (c6){error4_lable_book.setVisible(true);}
    if(c7){error565_lable_book.setVisible(true);}
    System.out.println(ct);
    System.out.println(c1);
    System.out.println(c2);
    System.out.println(c3);
    System.out.println(c4);
    System.out.println(c5);
    System.out.println(c6);
    System.out.println(c34);
    //send info to booking confirmation window
    if (ct)
    {new Booking_choose_room(firstname,lastname,num_children,num_adult,datefrom,dateto,_username, image_indentification, cn ).setVisible(true);
        this.dispose();}
```



Step-2

```
//Get the table model and reffer it to the table in the booking_choose_room pane
DefaultTableModel model = (DefaultTableModel)roomChooser_table.getModel();
//Array to hold the booked rooms
String[] RoomNos = new String[model.getRowCount()]  ;
 int count=0;
 //Cycle through the displayed table and check whihc rooms are chosen
while (model.getRowCount() > 0)
{
    if (model.getValueAt(0, 3).toString().equals("true")){count++; RoomNos[count]= model.getValueAt(0, 0).toString(); }
    model.removeRow(0);
}
String t = String.valueOf(cost_each_room);
int costEachRoom = Integer.parseInt(t) ;
int Total_cost = costEachRoom*count;
// check if the count > 0, as that will signify that there are rooms chosen to book if not then a error is displayed
if(count > 0){
new booking_confirm(_Booked_by_firstname, _Booked_by_lastname, _childerNO, _adultNo, _Booked_from, _Booked_to,RoomNos,Total_cost,count,image_indentification, cn).setVisible(true);
}
else
{
    Error_roomNoSelect.setVisible(true);
}
```

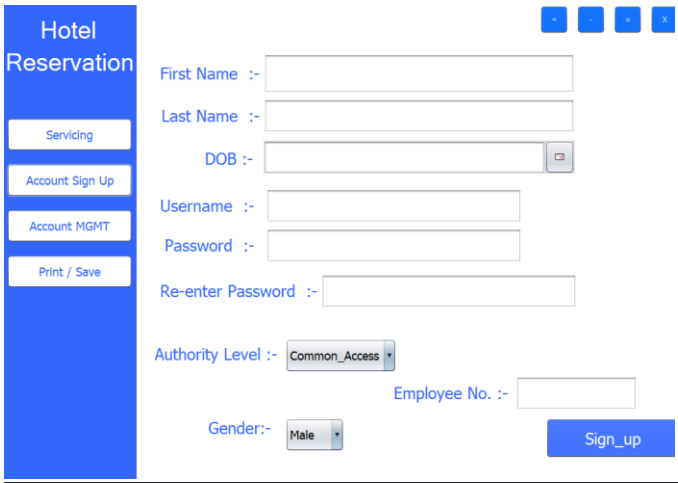
Step-3



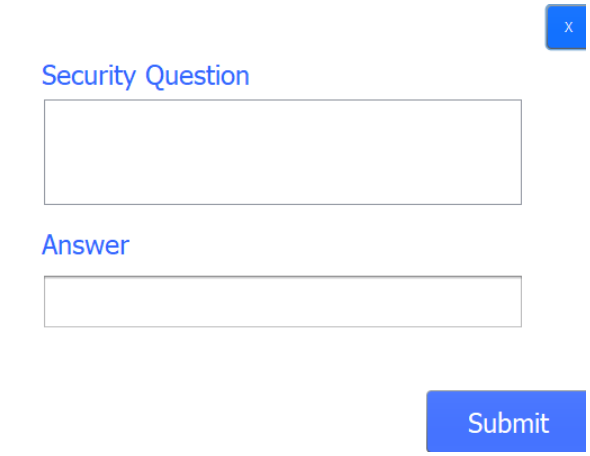

```
public void Set_Booking(String Booked_by_firstname,String Booked_by_lastname,String childerNO,String adultNo,String Booked_from,
        String Booked_to, String AllRoomNos,int total_cost) throws SQLException
{
    int child_no = Integer.parseInt(childerNO);
    int adult_no = Integer.parseInt(adultNo);
    //set new index number for the given booking
    getset_index();

    //prepare statement
    String sql =("INSERT INTO `bookings`(`Booked_by_firstname`, `Booked_by_lastname`, `Booked_from`, `Booked_to`, `Booked_rooms`, "
            + "`Total_cost`, `No. of Children`, `No. of Adults`,`Ind_book`,`identification`,`stat`) VALUES (?,?,?,?,?,?,?,?,?,?)") ;
    PreparedStatement ps = cn.prepareStatement(sql);

   try {
   //set values
    ps.setString(1,Booked_by_firstname );
    ps.setString(2,Booked_by_lastname );
    ps.setString(3,Booked_from );
    ps.setString(4,Booked_to );
    ps.setString(5,AllRoomNos );
    ps.setInt(6,total_cost );
    ps.setInt(7,child_no );
    ps.setInt(8,adult_no );
    ps.setInt(9,Set_index );
    ps.setBytes(10, image_indentification);
    ps.setString(11,"yet");
    // execute query
    ps.execute();

    JOptionPane.showMessageDialog(null, "submitted");
   }
   catch (HeadlessException | SQLException e){System.out.println(e);JOptionPane.showMessageDialog(null, "here" +e);}
}
```

6. Creating an employee account:

Creating an employee account requires employee's details, then a security question setup is needed. Lastly a pop-up display if the data was submitted or not.

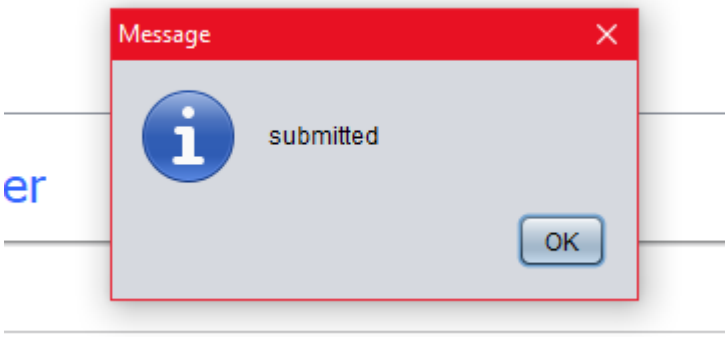



```
private void accountMake_buttonActionPerformed(java.awt.event.ActionEvent evt) {
    //initialization of variables
    String first_name = "";
    String last_name = "";
    String username = "";
    String password = "";
    String authority_level = "";
    String gender = "";
    String empNo = "";
    String dob = "";
    String passwordRe = "";
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
    //entering the data into the variables
    first_name = firstName_textbox.getText();
    last_name = lastName_textbox.getText();
    username = username_textbox.getText();
    password = password_textbox.getText();
    authority_level = String.valueOf(authorityLevel_combobox.getSelectedItem());
    gender = String.valueOf(gender_combobox.getSelectedItem());
    empNo = employeeNo_textbox.getText();
    //get the date from the pane
    try{
    dob = sdf.format(dob_datechooser.getDate());}
    catch(NullPointerException e){System.out.println(e);}
    passwordRe = passwordRe_textbox.getText();
    long millis=System.currentTimeMillis();
    java.sql.Date today =new java.sql.Date(millis);
    String Today = sdf.format(today);
    String check = "";
    //checks
    boolean c0 = (dob.compareTo(Today)>0);
    boolean c01 = dob.equals("");
    boolean c1 = first_name.equals(check);
    boolean c2 = last_name.equals(check);
    boolean c3 = username.equals(check);
    boolean c4 = password.equals(check);
    boolean c5 = empNo.equals(check);
    boolean c6 = password.equals(passwordRe);
    //checking the checks created earlier
    if (first_name.equals(check)){error1_lable.setVisible(true);}
    if (last_name.equals(check)){error2_lable.setVisible(true);}
    if (username.equals(check)){error4_lable.setVisible(true);}
    if (password.equals(check)){error5_lable.setVisible(true);}
    if (empNo.equals(check)){error7_lable.setVisible(true);}
    if (c0){error3_lable.setVisible(true);}
    if (c01){error3_lable.setVisible(true);}
    if (c6 == false){error6_lable.setVisible(true);}
    //execution of the process
    if (c1==false && c2==false && c3==false && c4==false && c5==false && c0 == false && c6 == true )
    {error_disable(); dob_datechooser.setDate(null);
    new Security_questions(first_name, last_name, username, password, authority_level,gender,empNo ,dob,passwordRe, cn).setVisible(true);
    }
}
```




```java
// get data from textboxes
String question = Question.getText();
String Ans = Answer.getText();
// check data
boolean c1  = question.equals("");
boolean c2  = Ans.equals("");
// Display error
if(c1){Error1.setVisible(true);}
if(c2){Error2.setVisible(true);}
//
if (c1 == false && c2 == false )
{

    dataInput(First_name ,Last_name ,Username,Password ,Authority_level ,Gender ,  EmpNo,  Dob ,PasswordRe ,question,Ans);
    System.out.println("3");
}
```

```java
// input into the database
public void dataInput(String first_name, String last_name, String username, String password, String authority_level,
       String gender, String empNo, String dob,String passwordRe,String question,String Ans)
{
    try {  //preparing the statement
           String sql = "INSERT INTO `login_employee`(`Emp.no`, `username`, `password`, `authority`, `First_name`, `Last_name`, `Gender`, `Date_of_birth`, `Security_Question` `
                   + ",`Security _Answer`) VALUES ('"+empNo+"','"+username+"','"+password+"','"+authority_level+"','"+first_name+"','"+last_name+"','"
                   +gender+"','"+dob+"','"+question+"','"+Ans+"')";
           PreparedStatement ps = cn.prepareStatement(sql);
           //execution of the update
           int rs = ps.executeUpdate();

           System.out.println("2");

           if(rs > 0  ){JOptionPane.showMessageDialog(null, "submitted");}
           this.dispose();
       } catch (SQLException e) {
           JOptionPane.showMessageDialog(null, "There was an error. Please try again.");
       }
}
```



7. Updating all the database tables:

```
package loop;
import java.sql.*;
import java.text.SimpleDateFormat;


public class get_set_data {
public  Connection cn =null;
    public  Statement st = null;
    public  void main(Connection con) {
        //conection();
        cn = con;
        checkBooking();

    }
```

Daily updates need to be made to the application as it is opened. This necessary as the auto unbooking system is in place to free up rooms for later re-booking the room under new customers. For this the update needs to happen on a frequency of daily bases.

```
// checks booking for updating the occupancy of the rooms and other attributes
private void checkBooking()
{
    try
    {
        String sql = "SELECT * FROM `bookings`";
        PreparedStatement ps = cn.prepareStatement(sql);
        ResultSet rs = ps.executeQuery();

        while(rs.next())
        {
            String booking_stat = rs.getString("stat");
            if(booking_stat.equals("yet"))
            {
                // get the booking dates
                String booked_date_from = rs.getDate("Booked_from").toString();
                String booked_date_to = rs.getDate("Booked_to").toString();

                // get todays date
                SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
                long millis=System.currentTimeMillis();
                java.sql.Date today =new java.sql.Date(millis);
                String Today = sdf.format(today);

                //get the booking info
                String fn = rs.getString("Booked_by_firstname");
                String ln = rs.getString("Booked_by_lastname");


                //comparison of dates
                boolean c1 = booked_date_from.compareTo(Today) <= 0;
                boolean c2 = booked_date_to.compareTo(Today) >= 0;

                System.out.println(c1+"  "+c2);

                //today is within the booked span, so set the room occupied
                if(c1 == true && c2 == true)
                {
                    break_rooms(rs.getString("Booked_rooms"),true, rs.getString("Ind_book"),fn,ln,booked_date_from,booked_date_to);
                }
                //today is after the total booked span, so set the booking to been and rooms to vacant
                if(c1 == true && c2 == false)
                {
                    break_rooms(rs.getString("Booked_rooms"),false, rs.getString("Ind_book"),fn,ln,booked_date_from,booked_date_to);
                }

            }
        }
    }
    catch(SQLException e )
    {
        System.out.print("1");
    }
}
```

This method takes in all the booked rooms of the booking and break it down in order to perform separate updates on each room.

```
private void break_rooms(String rooms,boolean check, String booking_ind, String fn, String ln,String bf,String bt)
{
    String rm = rooms;
            int num_words = 0;
            int y = rm.length();
            // to get the tot number of rooms in booking
            for(int z=0; z<y ; z++)
            {
                String t =rm.substring(z,z+1);
                if(t.equals(","))
                {
                    num_words++;
                }
            }


            // array to store each room
            String[] list = new String[num_words];
            String t2 ="";
            int ct = 0;
            //breaks all the roooms and puts them into the array
            for(int z=0; z<y ; z++)
            {
                String t = rm.substring(z,z+1);

                if(t.equals(","))
                {
                    list[ct] = t2;
                    ct++;
                    t2 ="";
                }
                else
                {
                    t2 = t2+t ;
                }

            }

            //set occupy or vacant depending on the date comparision in previous method
                if(check)
                {
                    set_occupy(list,num_words,booking_ind,fn,ln,bf,bt);
                }
                if(check == false)
                {
                    System.out.println("here");
                    set_vacant(list,num_words);
                }

            //set stat of a booking been cause today has passed the span of booked dates
            if(check == false)
            {
                set_booking_been(booking_ind);
            }
}
```

These are the separate updates that will be performed on each according to the output of the last method.

```
    private void set_occupy(String[] room,int num, String booking_ind, String first_name, String last_name, String bookfrom, String bookto)
    {
        System.out.println(room[0]);
        if(num>0){
        for(int s=0 ; s<num; s++){
        try
        {
            String sql ="UPDATE `each_room` SET `room_occupancy`='Occupied',`booked_from`= '"+bookfrom+"',`booked_to`='"+bookto+"',"
                    + "`Booked_by_firstname`='"+first_name
                    +"',`Booked_by_lastname`='"+last_name+"',`Related_bookingIndex`='"+booking_ind+"' WHERE `room_no` ='"+room[s]+"'";
            PreparedStatement ps = cn.prepareStatement(sql);
            int rs = ps.executeUpdate();
        }
        catch(SQLException e)
        {
           System.out.print("2" );
        }}}
    }
    private void set_vacant(String []room, int num)
    {
        System.out.println(num);
        if(num>0){
        for(int s=0 ; s<num; s++){
        try
        {
            String sql ="UPDATE `each_room` SET `room_occupancy`='Vacant',`booked_from`= null,`booked_to`=null,`Booked_by_firstname`='',"
                    + "`Booked_by_lastname`='',`Related_bookingIndex`='' WHERE `room_no` = '"
                    +room[s]+"'";
            PreparedStatement ps = cn.prepareStatement(sql);
            int rs = ps.executeUpdate();

        }
        catch(SQLException e )
        {
           System.out.print("3");
        }}}
    }
    private void set_booking_been(String booking_ind)
    {
         try
        {
            String sql ="UPDATE `bookings` SET `stat`='been' WHERE `Ind_book` = '"+booking_ind+"'";
            PreparedStatement ps = cn.prepareStatement(sql);
            int rs = ps.executeUpdate();
        }
        catch(SQLException e )
        {
           System.out.print("4");
        }
    }
}
```

I have also made use of additional libraries. Here is a list of the libraries I used:

- Jdatepicker- 1.3.4
- Sqlite-jdbc 3.7.2