Fwd__UKS31126.docx

by

Submission date: 16-Apr-2023 10:17AM (UTC-0700)
Submission ID: 2065997446
File name: Fwd__UKS31126.docx (720.61K)
Word count: 1369
Character count: 7176

PROGRAMMING ENGINEERING

Table of Contents

Introduction

This study mainly focuses on the development of a graphical user interface (GUI) based program by utilizing Java programming. The purpose of this program is to provide a Requirement Class where users can add full time and part time staff vacancy details, such as vacancy no, qualification, and other. In addition, users can terminate employees from the Requirement Class, and set the salary for full time staff as well as set the working shifts for part time staff. This programmer is an overview of the different Java code functionalities which can be utilized to create special apps for a range of uses. Apart from the Java NetBeans platform has been considered in order to complete the task.

Class Diagram of Requirement Class

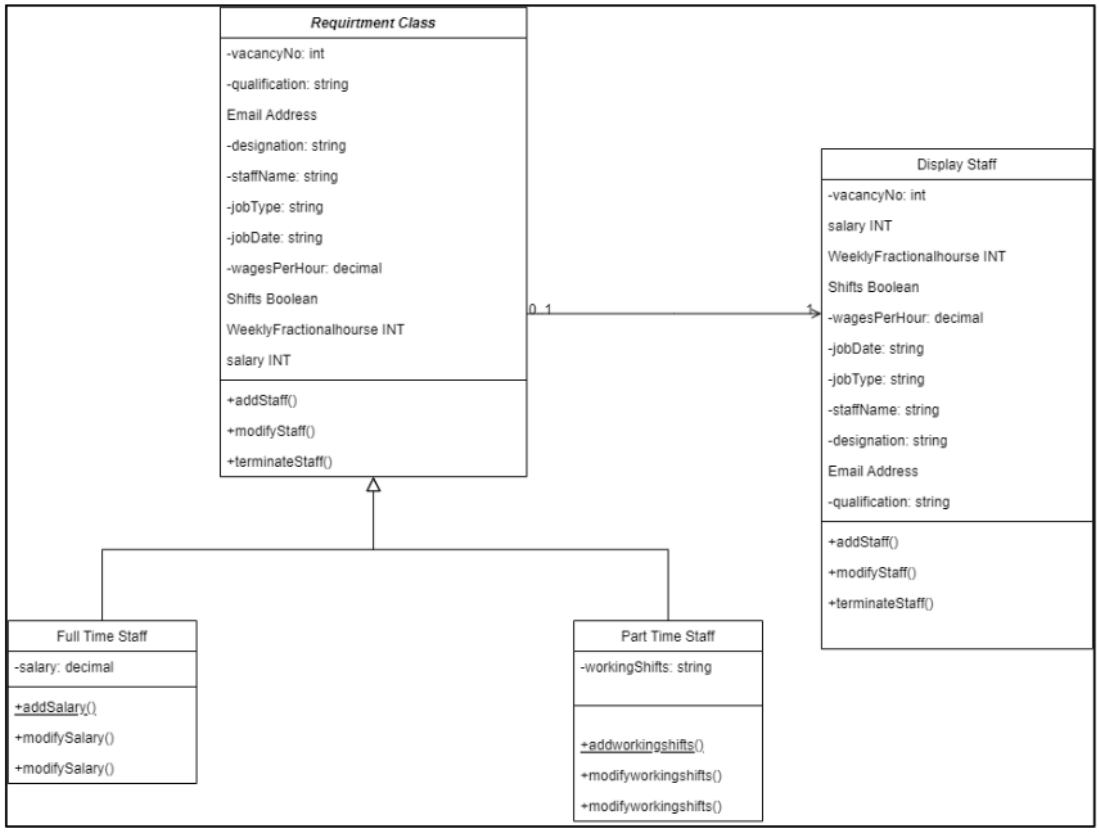


Figure 1: Class diagram of recruitment Class

(Source: Self-Created)

The Class Diagram of the Requirement Class has been represented as a system where users can manage staff vacancies, including full-time and part-time staff. Moreover, this system enables users in order to add, modify, as well as terminate staff details based on their needs. The Requirement class is the main class, which contains attributes such as vacancy number, qualification, designation, staff name, job type, job date, wages per hour, and flags for whether the staff is full-time or terminated (Díaz *et al*. 2021). These attributes are marked as private, indicating that they can only be accessed by methods within the class. This encapsulation ensures that the data stored in the class is protected from external manipulation. Apart from that, the system also contains two different subclasses, such as Fulltime Staff and Parttime Staff, which inherit from the Requirement class. These subclasses represent the two types of staff that can be hired, with Fulltime Staff having an additional attribute for salary, while Parttime Staff has an additional attribute for working shifts. This inheritance allows for code to be reutilized as well as and ensures that the subclasses have access to the attributes and methods of the parent class.

In addition to that, the Class Diagram of the Requirement Class provides a clear and concise representation of a system that allows users to manage staff vacancies. The class diagram shows how the system can handle full-time and part-time staff, and how users can add, modify, and terminate staff details. With the addition of methods to set salaries and working shifts, this system provides a comprehensive solution for managing staff vacancies.

A short description of each method in the Recruitment System class

The Recruitment System class provides methods for efficiently managing the recruitment and onboarding of staff members. The Add a full-time staff member method allows for the hiring of a new full-time employee. This method helps to enter the personal details like name, vacancy details and others of the new employee. It also allows for the selection of job categories and the setting of a salary. The Add a part-time staff member method allows for the hiring of a new part-time employee. This method helps to enter the personal details like name, vacancy details and others of the new employee (Bashir *et al*. 2021). Moreover, it also allows for the selection of job categories and the setting of shifts. Apart from that, The Set Salary - full-time staff member method allows the user to adjust the salary of a full-time employee. This method requires the entry of the employee's name and the new salary amount. The Set Shifts - part-time staff member method allows the user to adjust the shifts of a part-time employee. This method requires the entry of the

employee's name and the new shift details. Besides, the Terminate a part-time staff member method allows for the termination of a part-time employee. This method requires the entry of the employee's name and the effective date of termination. Furthermore, the Recruitment System class provides a comprehensive set of methods for managing the recruitment and onboarding process of staff members. These types of methods enabled for more efficient as well as accurate recruitment or onboarding of both full-time / part-time employees.

Pseudocode

- ***Add a vacancy for a full-time staff member***

```
WHEN "Add a vacancy for a full-time staff member" button is clicked
    CREATE a new full-time staff member
    ADD the new full-time staff member to the arraylist
END
```

Figure 2: Pseudocode

(Source: Self-Created)

- ***Add a vacancy for a part-time staff member***

```
WHEN "Add a vacancy for a part-time staff member" button is clicked
    CREATE a new part-time staff member
    ADD the new part-time staff member to the arraylist
END
```

Figure 3: Pseudocode

(Source: Self-Created)

- ***Set Salary full-time staff member***

```
WHEN "Set Salary full-time staff member" button is clicked
    PROMPT user for the salary for the full-time staff member
    SET the salary for the full-time staff member to the value entered
END
```

Figure 4: Pseudocode

(Source: Self-Created)

- ***Set Working shifts part-time staff member***

```
WHEN "Set Working shifts part-time staff member" button is clicked
    PROMPT user for the working shifts for the part-time staff member
    SET the working shifts for the part-time staff member to the value entered
END
```

Figure 5: Pseudocode

(Source: Self-Created)

- ***Terminate a part-time staff member***

```
WHEN "Terminate a part-time staff member" button is clicked
    PROMPT user for the part-time staff member to terminate
    REMOVE the part-time staff member from the arraylist
END
```

Figure 6: Pseudocode

(Source: Self-Created)

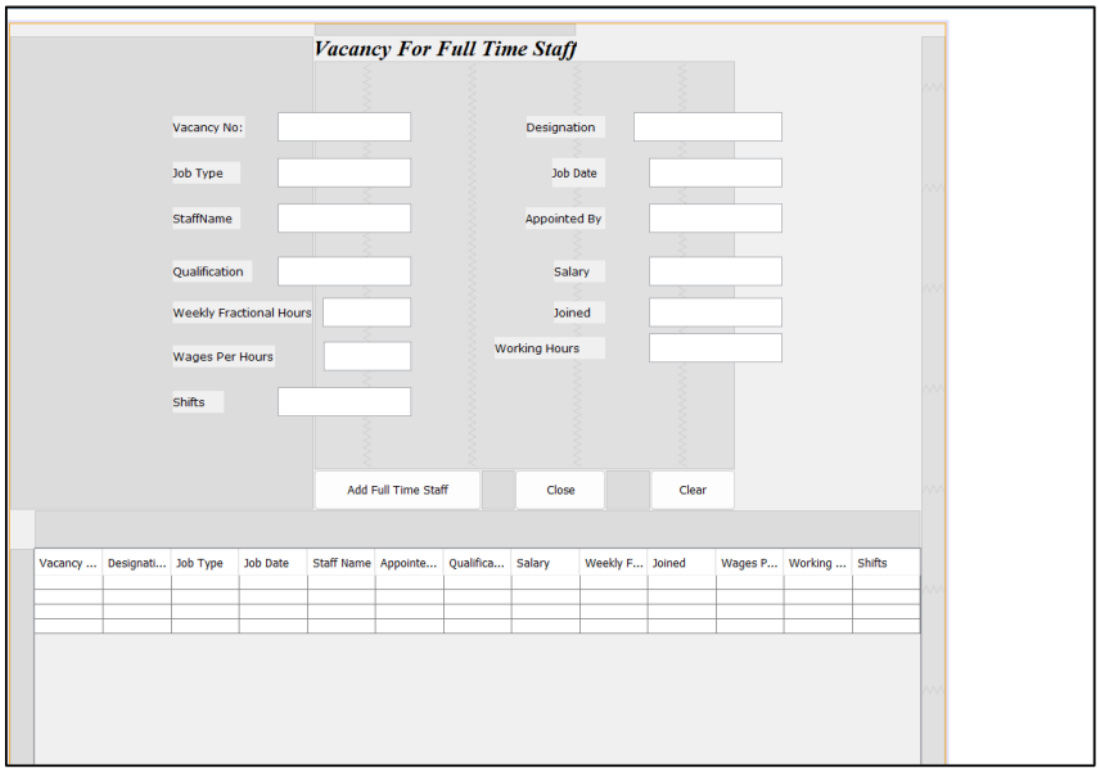
- ***Displaying a staff member in the array list***

```
WHEN "Displaying a staff members in the arraylist" button is clicked
    LOOP through the arraylist
        DISPLAY the details of each staff member in the arraylist
    END LOOP
END
```

Figure 7: Pseudocode

(Source: Self-Created)

Technical Analysis

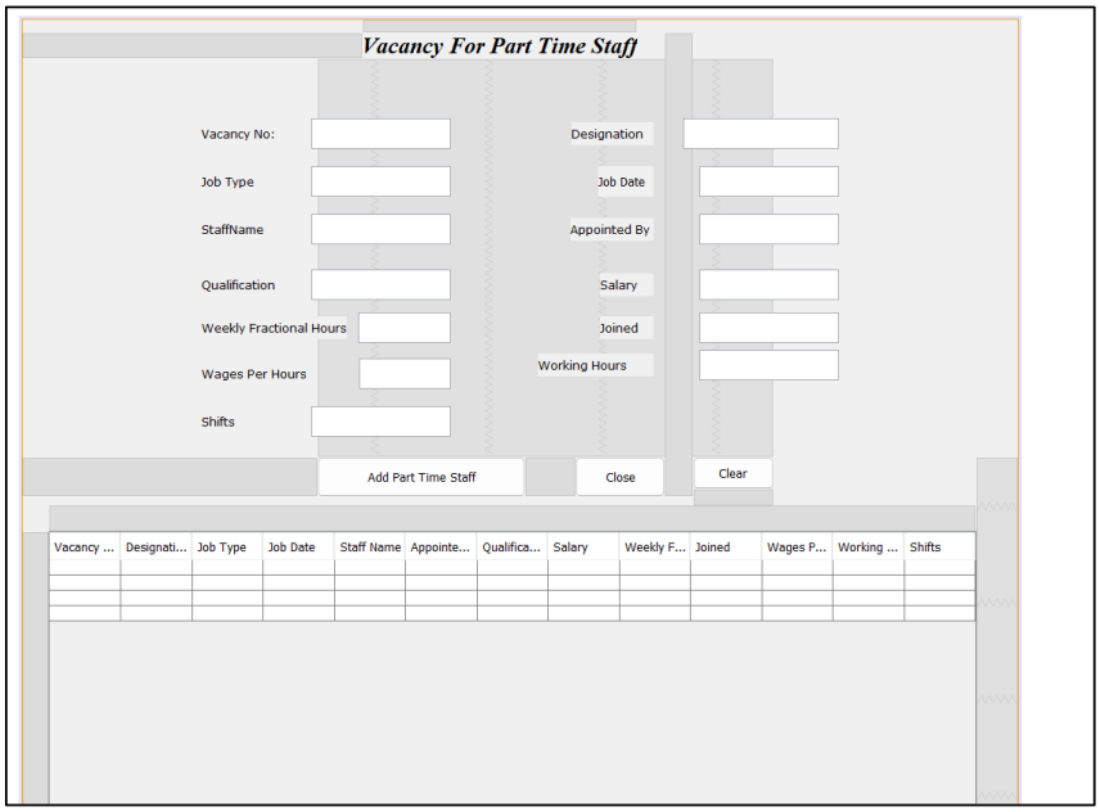


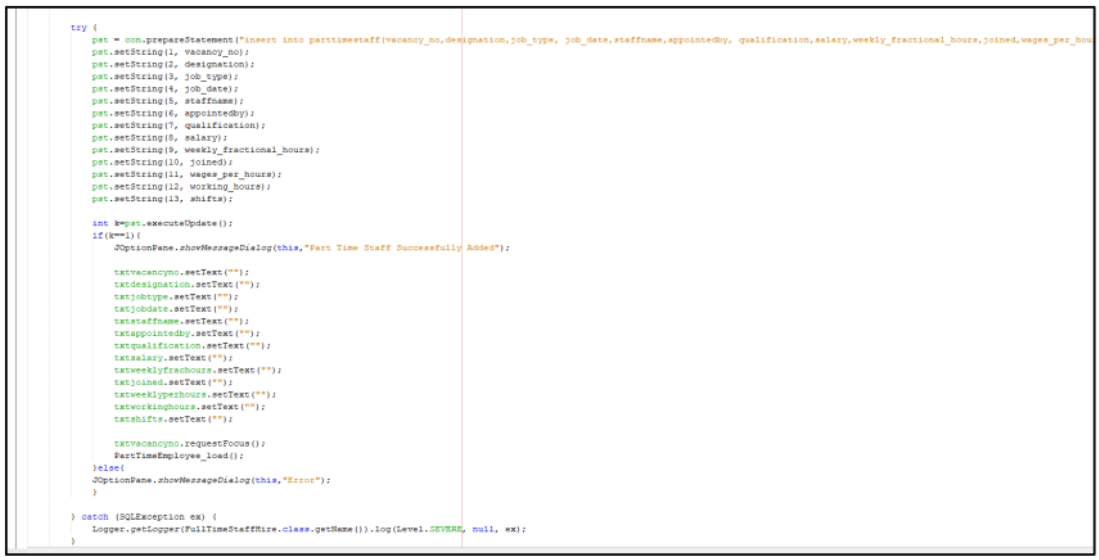
```
String designation=txtdesignation.getText();
String job_type=txtjobtype.getText();
String job_date=txtjobdate.getText();
String staffname=txtstaffname.getText();
String appointedby=txtappointedby.getText();
String qualification=txtqualification.getText();
String salary=txtsalary.getText();
String weekly_fractional_hours=txtweeklyfrachours.getText();
String joined=txtjoined.getText();
String wages_per_hours=txtweeklyperhours.getText();
String working_hours=txtworkinghours.getText();
String shifts=txtshifts.getText();

try {
    pst = con.prepareStatement("insert into fulltimestaff(vacancy_no,designation,job_type, job_date,staffname,appointedby, qualification,salary,weekly_fractional_hours,joined,wages_per_ho
    pst.setString(1, vacancy_no);
    pst.setString(2, designation);
    pst.setString(3, job_type);
    pst.setString(4, job_date);
    pst.setString(5, staffname);
    pst.setString(6, appointedby);
    pst.setString(7, qualification);
    pst.setString(8, salary);
    pst.setString(9, weekly_fractional_hours);
    pst.setString(10, joined);
    pst.setString(11, wages_per_hours);
    pst.setString(12, working_hours);
    pst.setString(13, shifts);

    int k=pst.executeUpdate();
    if(k==1){
        JOptionPane.showMessageDialog(this,"Full Time Staff Successfully Added");
        txtvacancyno.setText("");
        txtdesignation.setText("");
        txtjobtype.setText("");
        txtjobdate.setText("");
        txtstaffname.setText("");
        txtappointedby.setText("");
        txtqualification.setText("");
        txtsalary.setText("");
        txtweeklyfrachours.setText("");
        txtjoined.setText("");
        txtweeklyperhours.setText("");
        txtworkinghours.setText("");
```

Figure 8: Code to implement the Add Full time staff and its results

(Source: Self-Created)

In this case, the GUI base program has been designed based on the java programming and also the try-catch function enables the system to store the full-time staff vacancy details into this system. Besides, the "*getText* ()" function allows you to take the inputs from the text boxes and save them to the database.



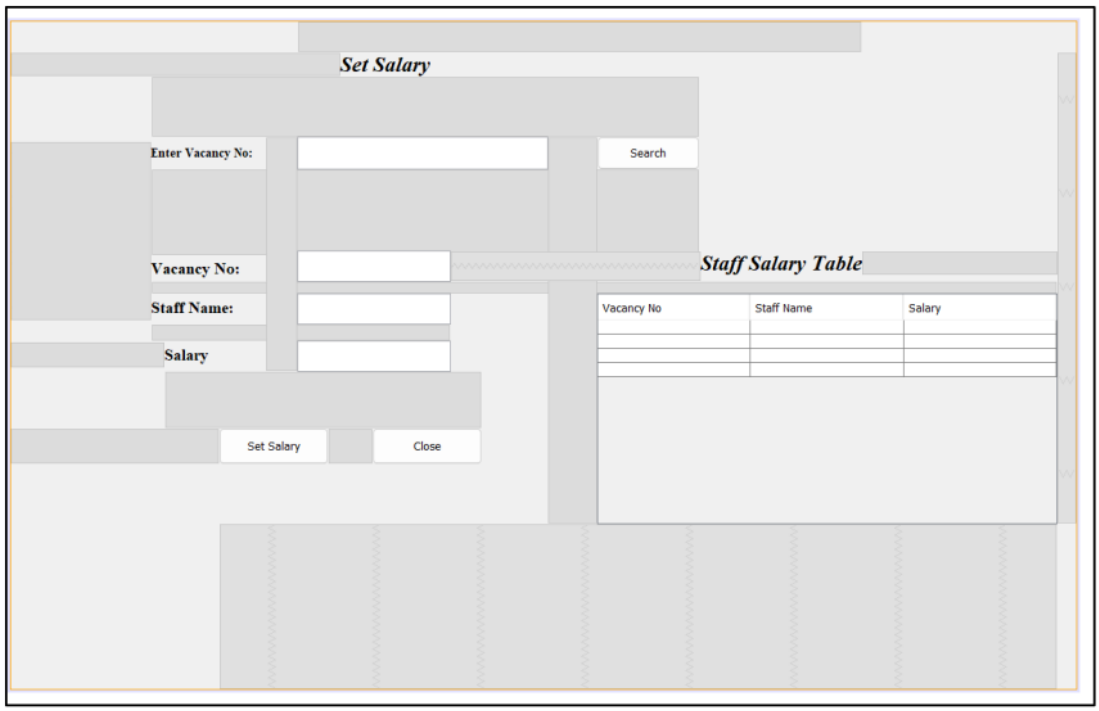


```
try {
    pst = con.prepareStatement("insert into parttimestaff(vacancy_no,designation,job_type, job_date,staffname,appointedby, qualification,salary,weekly_fractional_hours,joined,wages_per_hou
    pst.setString(1, vacancy_no);
    pst.setString(2, designation);
    pst.setString(3, job_type);
    pst.setString(4, job_date);
    pst.setString(5, staffname);
    pst.setString(6, appointedby);
    pst.setString(7, qualification);
    pst.setString(8, salary);
    pst.setString(9, weekly_fractional_hours);
    pst.setString(10, joined);
    pst.setString(11, wages_per_hours);
    pst.setString(12, working_hours);
    pst.setString(13, shifts);

    int k=pst.executeUpdate();
    if(k==1){
        JOptionPane.showMessageDialog(this,"Part Time Staff Successfully Added");

        txtvacancyno.setText("");
        txtdesignation.setText("");
        txtjobtype.setText("");
        txtjobdate.setText("");
        txtstaffname.setText("");
        txtappointedby.setText("");
        txtqualification.setText("");
        txtsalary.setText("");
        txtweeklyfrachours.setText("");
        txtjoined.setText("");
        txtweeklyperhours.setText("");
        txtworkinghours.setText("");
        txtshifts.setText("");

        txtvacancyno.requestFocus();
        PartTimeEmployee_load();
    }else{
    JOptionPane.showMessageDialog(this,"Error");
    }

} catch (SQLException ex) {
    Logger.getLogger(FullTimeStaffHire.class.getName()).log(Level.SEVERE, null, ex);
}
```

Figure 9: Code to implement the Add part time staff and its results

(Source: Self-Created)

The following image shows the java codes as well as outputs of the part time staff GUI page corresponding to the task. In this case, a part time staff function has been created to store part time vacancy details. The try-catch function is also applied in which SQL statements as well as java execution codes are written (Qiu, 2021).



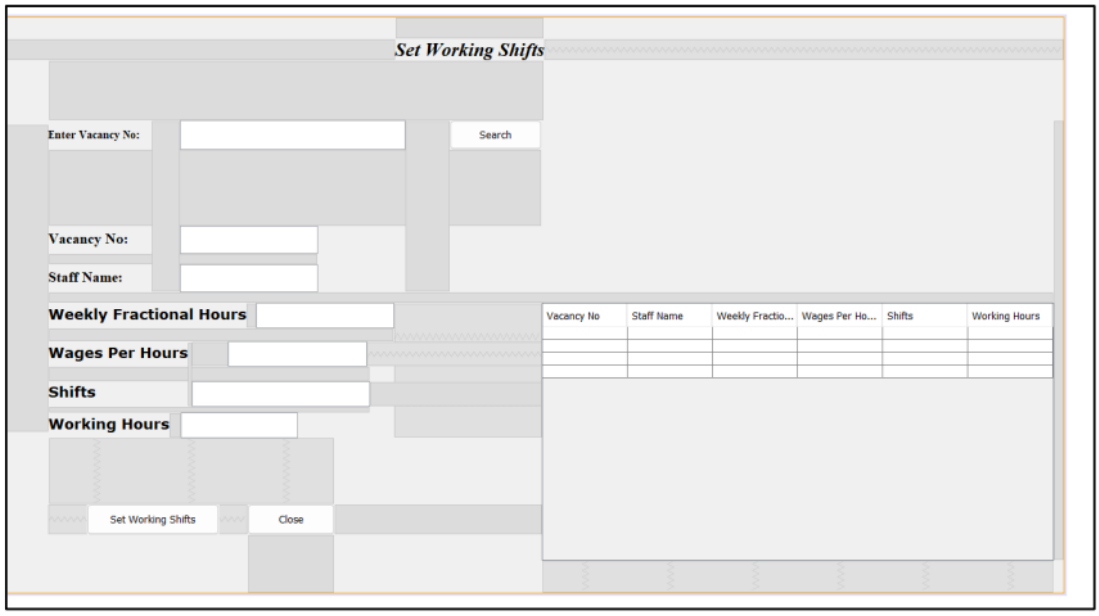

```
private void btnaddsalaryActionPerformed(java.awt.event.ActionEvent evt) {
    String vacancy_no=txtvacancyno.getText();

    String staffname=txtstaffname.getText();

    String salary=txtsalary.getText();
    try {
        pst = con.prepareStatement("insert into salarytable(vacancy_no,staffname,salary)values(?,?,?)");
        pst.setString(1, vacancy_no);
        pst.setString(2, staffname);
        pst.setString(3, salary);

        int k=pst.executeUpdate();
        if(k==1){
            JOptionPane.showMessageDialog(this,"Salary Successfully Added");

            txtvacancyno.setText("");

            txtstaffname.setText("");

            txtsalary.setText("");

            txtvacancyno.requestFocus();
            salary_table_load();
        }else{
        JOptionPane.showMessageDialog(this,"Error");
        }

    } catch (SQLException ex) {
        Logger.getLogger(SetSalary.class.getName()).log(Level.SEVERE, null, ex);
    }

}
```

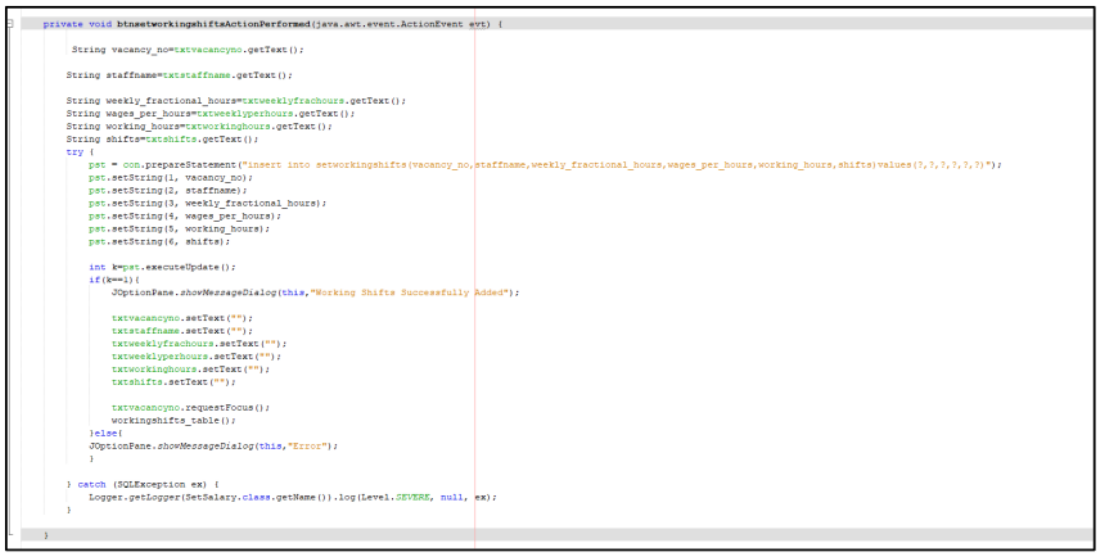
```
private void btnsearchActionPerformed(java.awt.event.ActionEvent evt) {

    try {
         pst= con.prepareStatement("select vacancy_no,staffname from fulltimestaff where vacancy_no=? ");
         int vacancy_no=Integer.parseInt(txtsearch.getText());
         pst.setInt(1, vacancy_no);
         ResultSet rs1= pst.executeQuery();
         if(rs1.next()==false){
         JOptionPane.showMessageDialog(this,"No Record Found");
         txtvacancyno.setText("");
         txtstaffname.setText("");
         txtvacancyno.requestFocus();
         }else{
         txtvacancyno.setText(rs1.getString("vacancy_no"));
         txtstaffname.setText(rs1.getString("staffname"));
         txtvacancyno.requestFocus();
         }
    } catch (SQLException ex) {
         Logger.getLogger(SetSalary.class.getName()).log(Level.SEVERE, null, ex);
     }
```

Figure 10: Code to implement the set salary and its results

(Source: Self-Created)

The above codes are helpful in order to design the page for set salary in which users can find the staff details by entering the vacancy no as well as system administrator can easily set up their salary details.



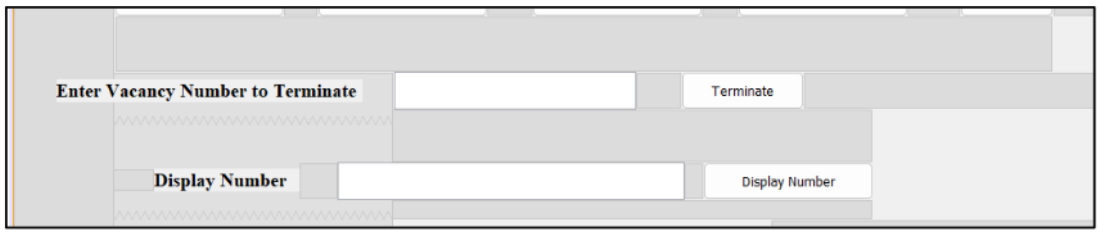


```
private void btnsetworkingshiftsActionPerformed(java.awt.event.ActionEvent evt) {

    String vacancy_no=txtvacancyno.getText();

    String staffname=txtstaffname.getText();

    String weekly_fractional_hours=txtweeklyfrachours.getText();
    String wages_per_hours=txtweeklyperhours.getText();
    String working_hours=txtworkinghours.getText();
    String shifts=txtshifts.getText();
    try {
        pst = con.prepareStatement("insert into setworkingshifts(vacancy_no,staffname,weekly_fractional_hours,wages_per_hours,working_hours,shifts)values(?,?,?,?,?,?)");
        pst.setString(1, vacancy_no);
        pst.setString(2, staffname);
        pst.setString(3, weekly_fractional_hours);
        pst.setString(4, wages_per_hours);
        pst.setString(5, working_hours);
        pst.setString(6, shifts);

        int k=pst.executeUpdate();
        if(k==1){
            JOptionPane.showMessageDialog(this,"Working Shifts Successfully Added");

            txtvacancyno.setText("");
            txtstaffname.setText("");
            txtweeklyfrachours.setText("");
            txtweeklyperhours.setText("");
            txtworkinghours.setText("");
            txtshifts.setText("");

            txtvacancyno.requestFocus();
            workingshifts_table();
        }else{
        JOptionPane.showMessageDialog(this,"Error");
        }

    } catch (SQLException ex) {
        Logger.getLogger(SetSalary.class.getName()).log(Level.SEVERE, null, ex);
    }

}
```

Figure 11: Code to implement the set shifts and its results

(Source: Self-Created)

Here, system users can find the staff vacancy details by entering the vacancy no and it shows the staff name. Therefore, users can easily set shifts as well as working hours in this system. The above codes are utilized to design and execute the backend part of the set shifts GUI page (Torres and Joanna, 2021).




```
private void btnterminatestaffActionPerformed(java.awt.event.ActionEvent evt) {
    try{
        Class.forName("com.mysql.jdbc.Driver");
        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/employee", "root", "");
        String sql="delete from fulltimestaff where vacancy_no=?";
        PreparedStatement pst=con.prepareStatement(sql);
        pst.setInt(1, Integer.parseInt(txtterminatevacancyno.getText()));
        pst.executeUpdate();
        JOptionPane.showMessageDialog(null,"Terminated Successfully");
        con.close();

    } catch (Exception e) {

    }
}
```

Figure 12: Code to implement the terminate employees and its results

(Source: Self-Created)

The following codes are applied for which users can easily terminate their staff by entering the vacancy no on this GUI based system.

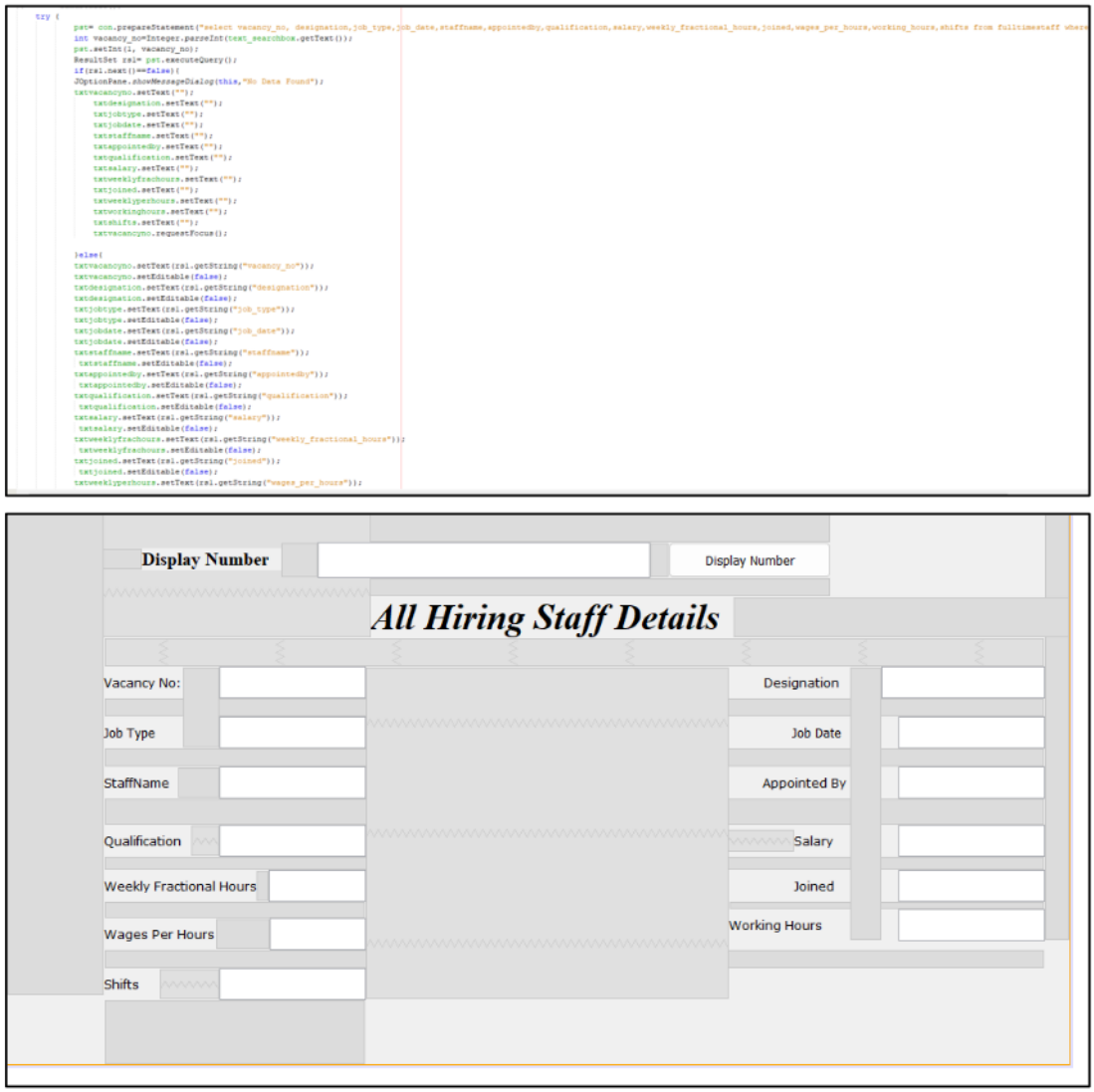


Figure 13: Code to implement the Display vacancy numbers and its results

(Source: Self-Created)

The above image shows the codes and outputs of the GUI page in which users can view the full time or part time vacancy details by entering the vacancy no.

Conclusion

It can be concluded that the implementation of the Requirement Class in Java has been successfully performed corresponding to the task. It has been designed to allow users to add full-time and part-time staff vacancy details, as well as terminate employees and set salary and working shifts for part-time staff. In addition to that, more effective, this technique may be applied for speeding up the hiring process of an organization or others. The Requirement Class's functionality may be expanded in the future by the addition of new features. For instance, to help customers manage their personnel more effectively, the capability to save and analyze data from the Requirement Class may be implemented.

Reference Lists

Bashir, N., Bilal, M., Liaqat, M., Marjani, M., Malik, N. and Ali, M., 2021, March. Modeling class diagram using nlp in object-oriented designing. In *2021 National Computing Colleges Conference (NCCC)* (pp. 1-6). IEEE.

Díaz, E., Panach, J.I., Rueda, S. and Vanderdonckt, J., 2021. An empirical study of rules for mapping BPMN models to graphical user interfaces. *Multimedia Tools and Applications*, *80*, pp.9813-9848.

Qiu, C., 2021, May. Design and Implementation of Personnel Recruitment System in Higher Vocational Colleges Based on SSH2. In *2021 6th International Conference on Smart Grid and Electrical Automation (ICSGEA)* (pp. 284-287). IEEE.

Torres, E.D. and Joanna, A., 2021. Implementation of Decision Support Personnel Recruitment System for Laguna State Polytechnic University-San Pablo City Campus. *International Journal of Managing Information Technology (IJMIT) Vol*, *13*.

Fwd__UKS31126.docx

ORIGINALITY REPORT

0%	0%	0%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On		