National University – Manila

College of Computing and Information Technologies

Information Technology

Kween Telecom App: A Payroll System

Slaenalyn Kweentolome

John Paul Total

Joan Tompong

Patrick Allen Chiquito

Myrna Viena

Aliah Ann Bribon

Karylle Corpuz

Dr. Roben A. Juanatas

Professor

March 15, 2023

II. Project Context

Every organization's payroll department performs essential business functions. Prior to the adoption of computers, all payroll calculations were done manually. Many journals required accountants, in which they had to record every performance variation, leave, deduction, and bonus. Also, companies had to have fully separate filing rooms with records of every employee, which meant that otherwise useful space was being used to hold massive boxes of paperwork. This act of managing payroll proves to be quite strenuous, not safe, time-consuming, and not to mention obsolete. But nowadays, things are far simpler than they used to be because of the advancements in technology and innovations. A computerized one will be operating automatically to keep track of employee data, performance, and attendance. This eliminates space for error, which cannot be stated for manual payroll systems, which are prone to errors and miscalculations. These systems save a ton of time and money by automating the fundamental documentation of each employee's performance, attendance, pay, and monthly deductions. Payroll must be secured and can only be accessed by the owner or the company's administrator. And that is how we came up with the idea of creating the Kween Telecommunications Payroll System.

III. General Objective

The general objective of this project is to develop a computerized payroll system. A payroll system called Kween Telecom App was created to monitor the leaves and benefits to make sure that workers are fairly compensated for their work. The addition of a table and computation that makes it simpler for the administrator to compute and distribute staff wages is an upgrade to this system. The method makes it easier for the administrator to manage payroll by ensuring that each employee is fairly rewarded for their contributions. This system also has a database where administrators can keep track of employees’ data and performance.

IV. User Manual

This section shows the Graphical User Interface (GUI) and the functionalities of our Payroll System. This portion will also demonstrate how user-friendly it is. The proponents used Java Swing to implement the program. Java Swing is a set of APIs that provides a graphical user interface (GUI) for Java programs. Java Swing is also known as the Java GUI widget toolkit.

Figure 1. The admin log-in interface allows administrators or authorized users to access the payroll system, by entering login credentials such as username and password. This interface provides a secure way to access privileged information and perform administrative tasks such as managing employees' personal information and modifying contents of the system.

Figure 2. Once the correct credentials are entered, the user gains access to the payroll system's administrative functions and can perform tasks based on the level of permissions granted to their account as an admin. This system configured the login with the username and password as "**slaenalyn**" and "**kweentelecom**" respectively.

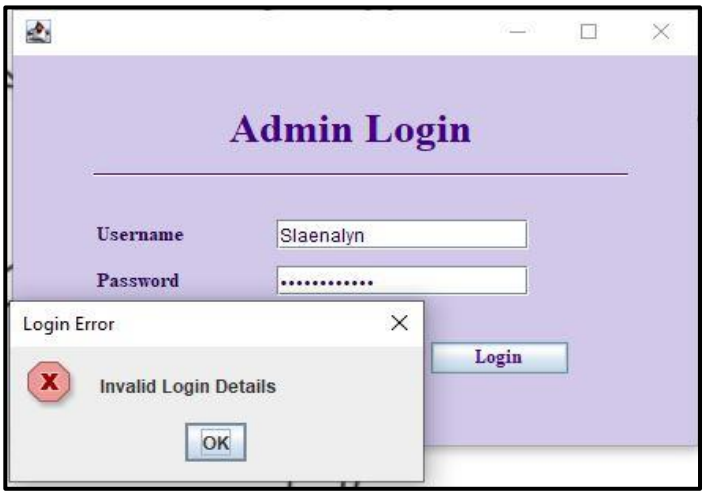

Figure 3. When you enter the wrong credentials on a login interface, you will receive an error message from a pop-up alert informing you that the username or password is incorrect. This is a security measure to prevent unauthorized access to the payroll system.

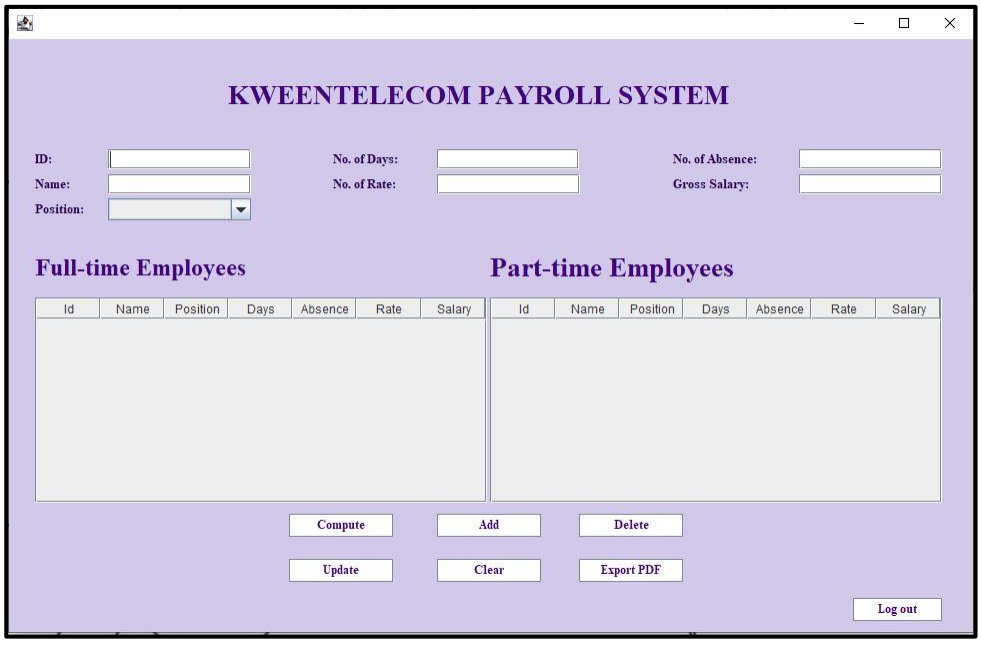

Figure 4. This figure shows the main system interface that displays toolbars and navigational elements such as different text fields and buttons that allow users to access various parts of the payroll system. It also contains two tables that provide separate information, status updates, and other important details about the employees depending on their job position, whether full-time or part-time.

Figure 5. This figure shows that when a user attempts to submit a form or input data into a field that has not been filled out, a warning message will appear. This warning message serves as a reminder to the user that the field must be completed before the form can be submitted or the data can be saved because incomplete or inaccurate data can cause errors, delays, or other issues. Aside from that, warning messages ensure that the data entered into the system is complete and accurate, which is critical for the proper functioning of the payroll system.

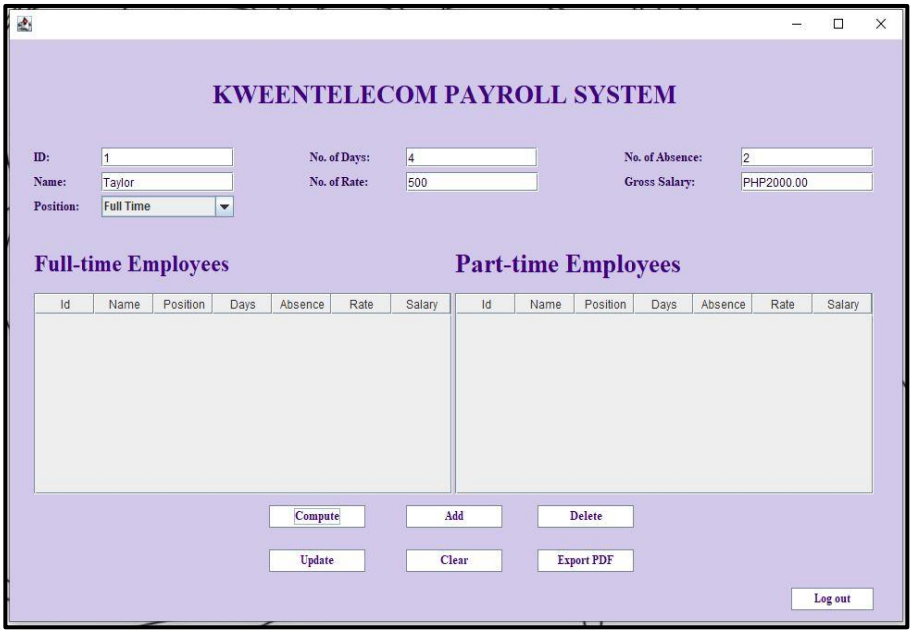

Figure 6. This figure shows the automated number of absences and the exact amount of salary for a full-time employee by clicking the compute button as shown in figure 6. Remember that rate and salary vary depending on the position and number of days an employee worked.

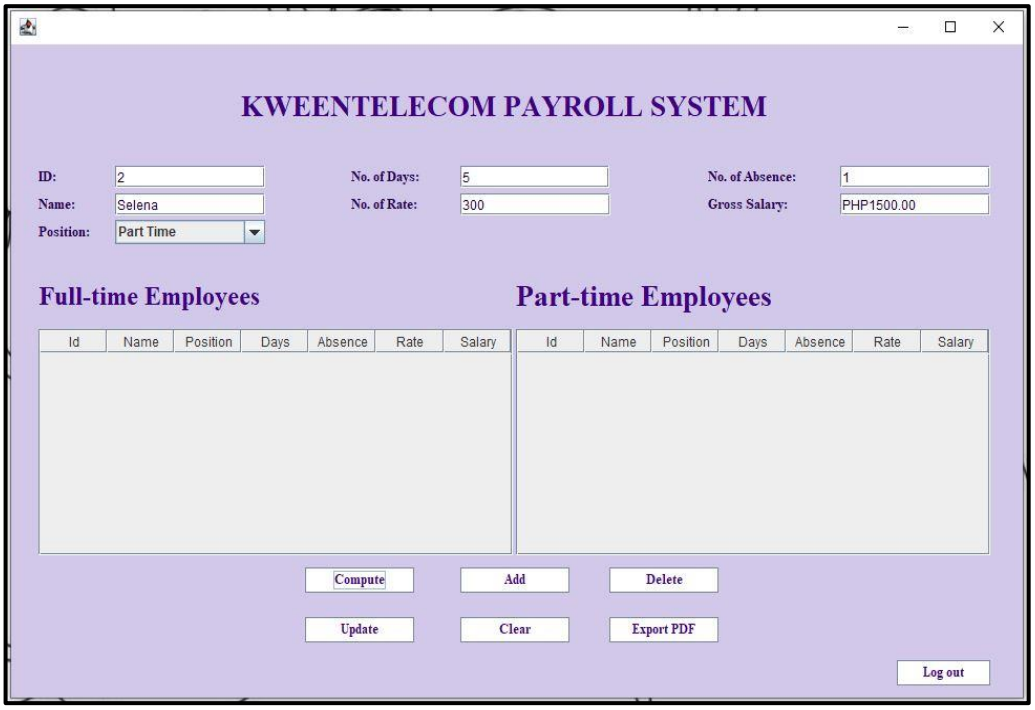

Figure 7. This figure shows the automated number of absences and the exact amount of salary for a part-time employee by clicking the compute button as shown in figure 6. Remember that rate and salary vary depending on the position and number of days an employee worked.

Figure 8. The user can remove every input of the user by using the CLEAR button.

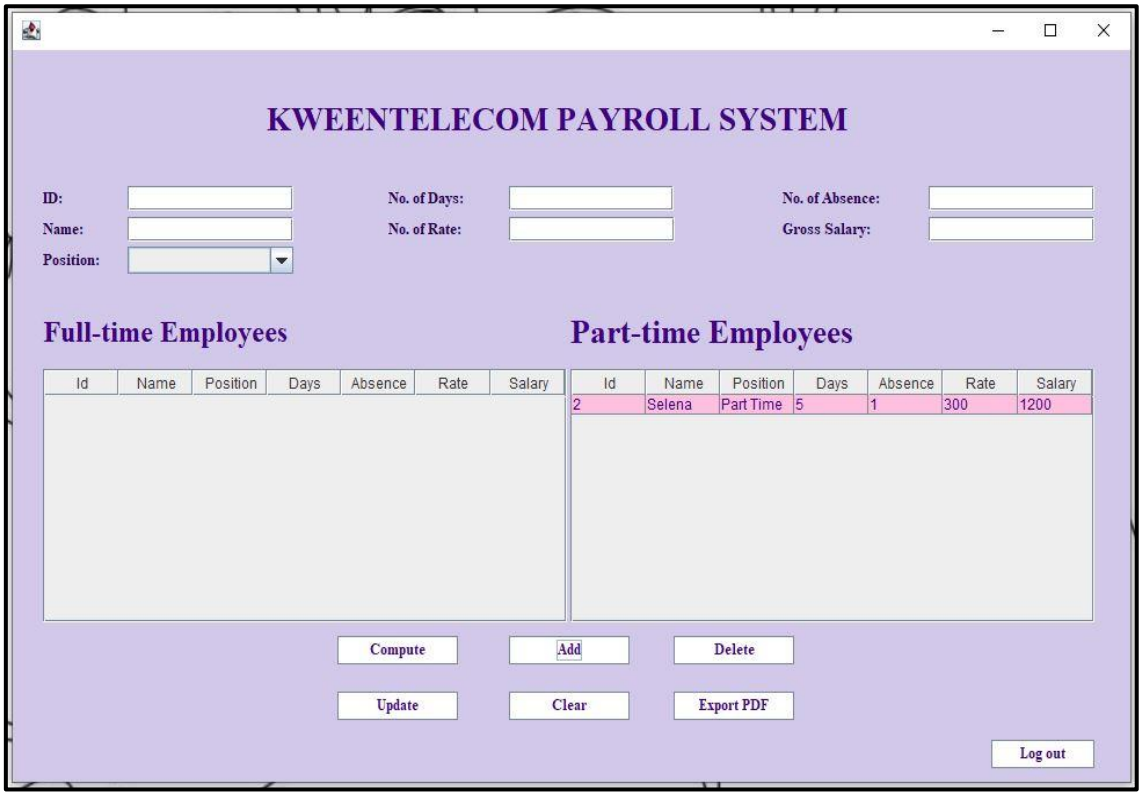

Figure 9 & 10. The figures show how the add button works. User must input the necessary employee information such as Id, Name, Position, Number of Days, and Rate then compute for absences and salary through compute button and finally, click the add button to display entered employee info from what table they belong.

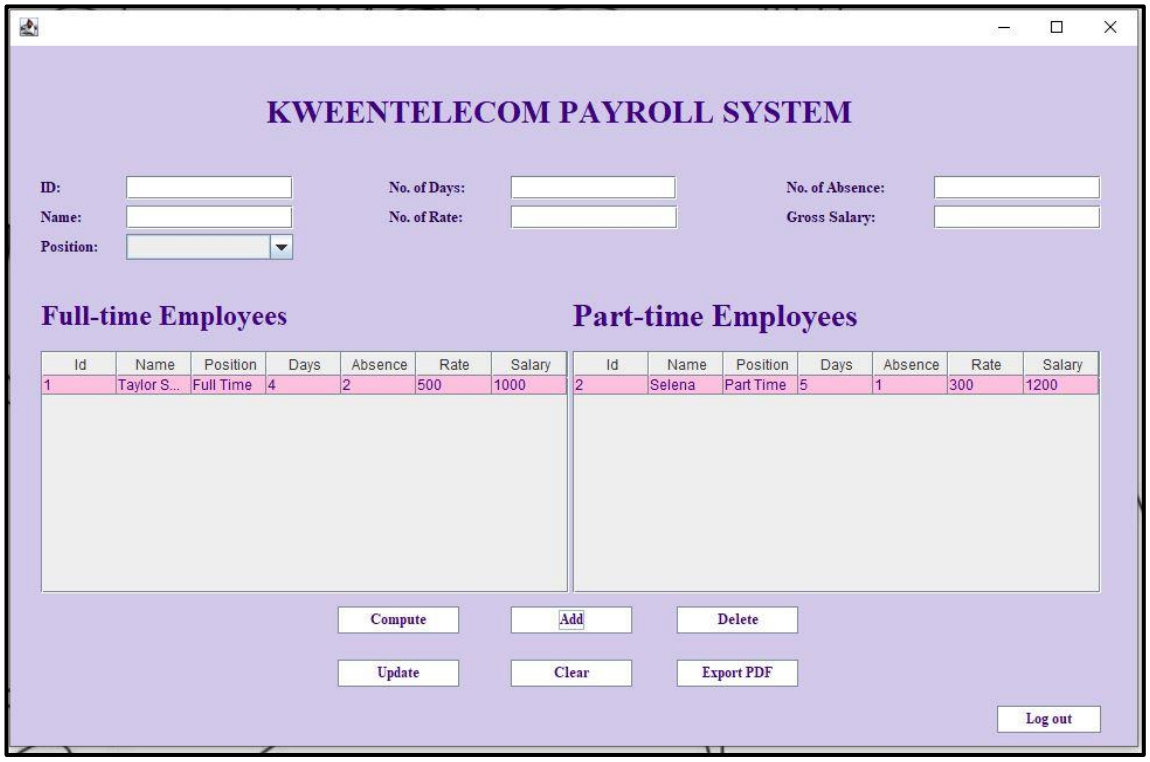

Figure 11. This figure shows how the system displays separate tables of part-timers and full-timers. Remember to indicate different employee IDs to prevent system errors.

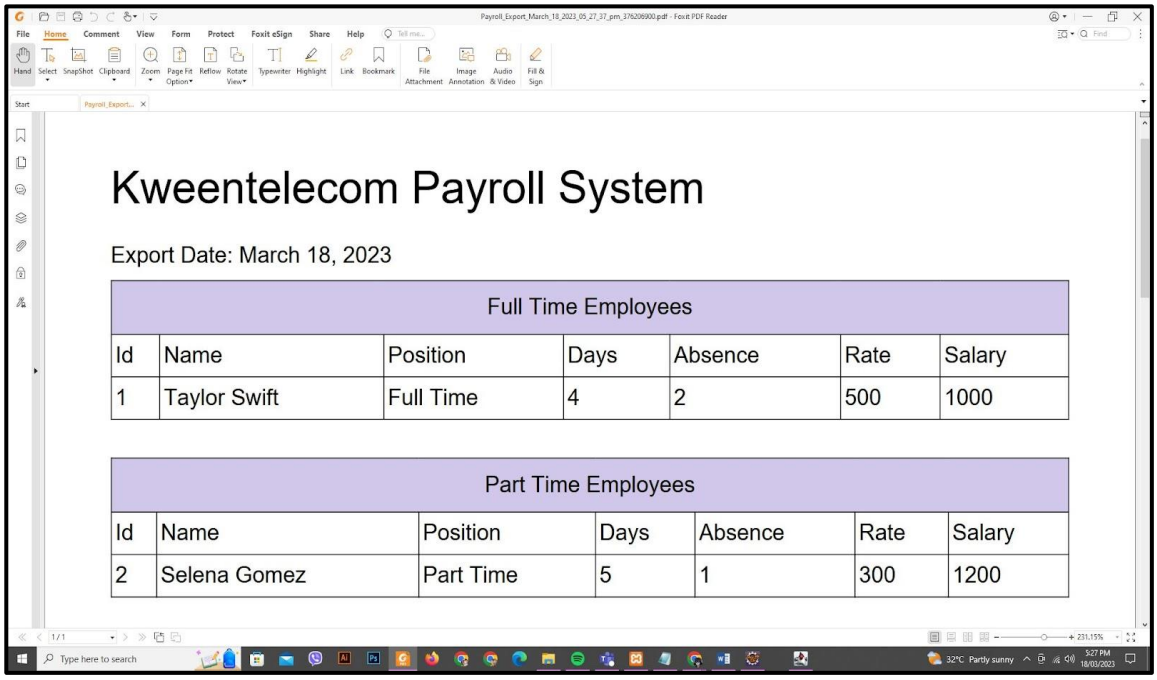

Figure 12. The user can export the list of part-time and full-time employee data in PDF format by selecting the ExportPDF button.

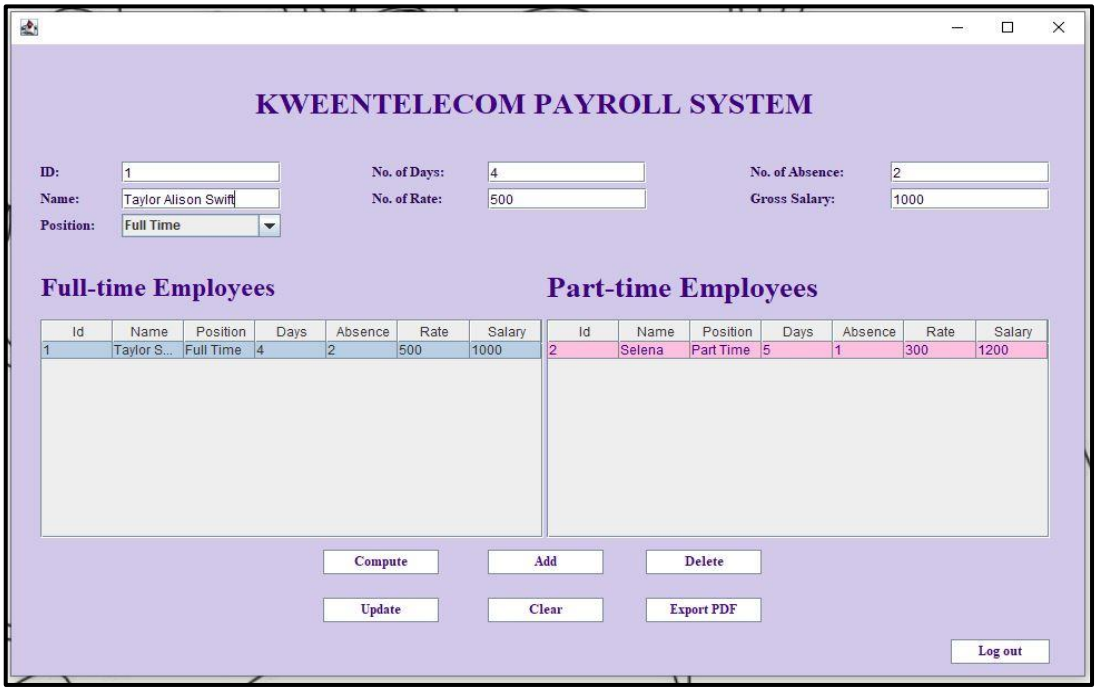

Figure 13. This figure shows how the system allows the user to change employee information using the button UPDATE. The previous information must reflect on their specific field first before re-entering the new information. Lastly, the changes reflect on the list when the update button is clicked.

Figure 14. Lastly, when log-out button is clicked, the system returns to the first interface of the payroll system, the admin log-in.

IV. Program Code

This part shows the program code for the connection of the system into the database. The SQL part of "MySQL" stands for "Structured Query Language". SQL is the most common standardized language used to access databases.

ADD BUTTON

```
                //ADD BUTTON
                JButton btnAdd = new JButton("Add");
                btnAdd.addActionListener(new ActionListener() {
                    public void actionPerformed(ActionEvent arg0) {
                        if (
                                    id.getText().equals("") ||
                            name.getText().equals("") ||
                            days.getText().equals("") ||
                            absence.getText().equals("") ||
                            rate.getText().equals("") ||
                            position.getSelectedItem().toString().equals("") ||
                            salary.getText().equals(""))
                        {
                            JOptionPane.showMessageDialog(null, "Please Fill Complete
Information!");
                        } else {
                            // Make position field value reusable.
                            String positionValue = position.getSelectedItem().toString();
                            // Calculate salary
                            int salaryValue = 0;
                            if (positionValue.equals("Part Time")) {
                                int daysValue = Integer.parseInt(days.getText());
                                int rateValue = Integer.parseInt(rate.getText());
                                int absenceValue = Integer.parseInt(absence.getText());
                                salaryValue = (daysValue * rateValue) - (absenceValue *
rateValue);
                            } else if (positionValue.equals("Full Time")) {
                                int daysValue = Integer.parseInt(days.getText());
                                int rateValue = Integer.parseInt(rate.getText());
                                int absenceValue = Integer.parseInt(absence.getText());
                                salaryValue = (daysValue * rateValue) - (absenceValue *
rateValue);
                            }
                            salary.setText("" + String.valueOf(salaryValue));

                            Object tableRow[] = {id.getText(), name.getText(), positionValue,
days.getText(),
                                    absence.getText(), rate.getText(), salary.getText()};

                            DefaultTableModel currentTableModel = model;

                            if (positionValue.equals("Part Time")) {
                                //SQL query for part-time employees
                                String sql = "INSERT INTO `employeetable` " +
                                "(`employee_id`, `employee_name`, `employee_position`,
`employee_days`, `employee_rate`, `employee_absences`, `employee_salary`) " +
                                        "VALUES "+"('"+id.getText()+"'," + // AUTOINCREMENT ID
                                        "'" + name.getText() + // NAME
                                        "','Part Time','" + // POSITION
                                        Integer.parseInt(days.getText()) + // DAYS
                                        "','" + Integer.parseInt(rate.getText()) + //RATE
                                        "','" + Integer.parseInt(absence.getText()) + //ABSENT
                                        "','" + salaryValue + "');"; //SALARY

                                // Database connection and insertion for part-time employees
                                String url = "jdbc:mysql://localhost/";
                                String dbName = "payrollsystemdb";
                                String driver = "com.mysql.jdbc.Driver";
                                String userName = "root";
                                String password = "";

                                try (Connection conn = DriverManager.getConnection(url+dbName,
userName, password);
                                    Statement state = conn.createStatement();
                                    ) {

                                    int rowsAffected = state.executeUpdate(sql);
                                    System.out.println(rowsAffected + " row(s) affected");
```

```
userName, password);
                                    Statement state = conn.createStatement();
                                    ) {

                                    int rowsAffected = state.executeUpdate(sql);
                                    System.out.println(rowsAffected + " row(s) affected");

                                    conn.close();

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

                                currentTableModel = partTimeEmployeesTableModel;

                            } else if (positionValue.equals("Full Time")) {
                                // SQL query for full-time employees
                                String sql = "INSERT INTO `employeetable` " +
                                "(`employee_id`, `employee_name`, `employee_position`,
`employee_days`, `employee_rate`,`employee_absences`,`employee_salary`) " +
                                        "VALUES "+("'"+id.getText()+"'," + // AUTOINCREMENT ID
                                        "'" + name.getText() + // NAME
                                        "','Full Time','" + // POSITION
                                        Integer.parseInt(days.getText()) + // DAYS
                                        "','" + Integer.parseInt(rate.getText()) + //RATE
                                        "','" + Integer.parseInt(absence.getText()) + //ABSENT
                                        "','" + salaryValue + "');"; //SALARY

                                // Database connection and insertion for full-time employees
                                  String url = "jdbc:mysql://localhost/";
                                      String dbName = "payrollsystemdb";
                                      String driver = "com.mysql.jdbc.Driver";
                                      String userName = "root";
                                      String password = "";

                                        try (Connection conn =
DriverManager.getConnection(url+dbName, userName, password);
                                            Statement state = conn.createStatement();
                                            ) {

                                            int rowsAffected = state.executeUpdate(sql);
                                            System.out.println(rowsAffected + " row(s)
affected");

                                            conn.close();

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

                                        currentTableModel = fullTimeEmployeesTableModel;
                                }

                            currentTableModel.addRow(tableRow);

                            parentObject.clearFormFields();

                            Employee el = new Employee(tableRow);

                            }
                        }
                    });
```

This code is for the "Add" button in a payroll system. When the button is clicked, it checks whether all the required fields (id, name, days, absence, rate, position, salary) have been filled. If they have not been filled, a message box is displayed asking the user to fill in all the required fields. If all fields have been filled, the code calculates the salary based on the employee's position (part-time or full-time), updates the salary field, and then inserts the employee information into a database table using SQL queries.

- If the employee is part-time, the SQL query inserts the employee's information into a table called "employeetable" with the values for the corresponding columns (employee_id, employee_name, employee_position, employee_days, employee_rate, employee_absences, employee_salary). If the employee is full-time, the SQL query is similar but with "Full Time" instead of "Part Time" in the "employee_position" column.
- After the employee information has been added to the database, the code updates the current table model (either for part-time or full-time employees), adds the employee information to the current table model, clears the form fields, and creates a new Employee object with the added employee's information.
- This is where this code connects to a MySQL database using JDBC (Java Database Connectivity) API. The connection is established in the following lines of code.

```
                                String url = "jdbc:mysql://localhost/";
                                String dbName = "payrollsystemdb";
                                String driver = "com.mysql.jdbc.Driver";
                                String userName = "root";
                                String password = "";

                                try (Connection conn =
DriverManager.getConnection(url+dbName, userName, password);
                                    Statement state = conn.createStatement();
                                    ) {

                                    int rowsAffected = state.executeUpdate(sql);
                                    System.out.println(rowsAffected + " row(s)
affected");

                                    conn.close();

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

- The url string contains the URL of the MySQL server, including the protocol, hostname, and port number. The dbName string contains the name of the database to connect to. The userName and password strings contain the username and password to authenticate with the database server.
- The DriverManager.getConnection method is used to establish a connection to the database. Once the connection is established, a Statement object is created to execute SQL queries or updates.
- After the database operation is completed, the conn.close() method is called to close the connection to the database. Any exceptions thrown during the connection process or database operation are caught and printed to the console.

Delete Button

```
                //DELETE ROW
                JButton btnDelete = new JButton("Delete");
                btnDelete.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {

                                JTable focusedTable = parentObject.focusedTable;
                                int rowIndex = focusedTable.getSelectedRow();

                                System.out.println("Focused Table Name: " +
parentObject.focusedTableName);
                                System.out.println("rowIndex: " + rowIndex);

                                if(rowIndex >= 0)
                                {
                                        String positionValue =
focusedTable.getModel().getValueAt(rowIndex, 2).toString();

                                        System.out.println(positionValue);

                                        DefaultTableModel currentTableModel = model;
                                        if (positionValue.equals("Part Time")) {

//
        ***************************************************
                                                String sql = "DELETE FROM `employeetable` WHERE
`employeetable`.`employee_id` = ?;";
                                                 // Database connection and insertion for part-time
employees
                                String url = "jdbc:mysql://localhost/";
                                String dbName = "payrollsystemdb";
                                String driver = "com.mysql.jdbc.Driver";
                                String userName = "root";
                                String password = "";

                                try (Connection conn = DriverManager.getConnection(url+dbName,
userName, password);
                                        Statement state = conn.createStatement();
                                        ) {

                                        PreparedStatement stmt = conn.prepareStatement(sql);
                                        stmt.setInt(1,
Integer.parseInt(id.getText().toString()));
                                        stmt.executeUpdate();

//                                      int rowsAffected = ;
                                        System.out.println(state.executeUpdate(sql));

                                        conn.close();

                                        } catch (SQLException el) {
                                            System.out.println(el.getMessage());
                                        }

//                                      ***************************************************

                                                currentTableModel =
parentObject.partTimeEmployeesTableModel;
                                        currentTableModel.removeRow(rowIndex);
                                        parentObject.clearFormFields();

                                                JOptionPane.showMessageDialog(null, "Deleted
Successfully");

                                        } else if (positionValue.equals("Full Time")); {

//
        ***************************************************
                                                String sql = "DELETE FROM `employeetable` WHERE
`employeetable`.`employee_id` = ?;";
```

```
                                        // Database connection and insertion for part-time
employees
                                String url = "jdbc:mysql://localhost/";
                                String dbName = "payrollsystemdb";
                                String driver = "com.mysql.jdbc.Driver";
                                String userName = "root";
                                String password = "";

                                try (Connection conn = DriverManager.getConnection(url+dbName,
userName, password);
                                        Statement state = conn.createStatement();
                                        ) {

                                    PreparedStatement stmt = conn.prepareStatement(sql);
                                    stmt.setInt(1,
Integer.parseInt(id.getText().toString()));
                                    stmt.executeUpdate();

//                                      int rowsAffected = ;
                                        System.out.println(state.executeUpdate(sql));

                                        conn.close();

                                    } catch (SQLException e1) {
                                        System.out.println(e1.getMessage());
                                    }

//                              ****************************************************

                                            currentTableModel =
parentObject.fullTimeEmployeesTableModel;
                                    currentTableModel.removeRow(rowIndex);
                                    parentObject.clearFormFields();

                                            JOptionPane.showMessageDialog(null, "Deleted
Successfully");
                                }
                                }       else {
                                    JOptionPane.showMessageDialog(null, "Please Select a
Row");
                                }
                            }
                        });
```

This code defines a button named "Delete" that will perform an action when clicked. The actionPerformed method is defined within an anonymous ActionListener class, which is added as a listener to the button. When the button is clicked, the actionPerformed method is called.

- Within the actionPerformed method, the code first retrieves the selected row index of a JTable object named focusedTable from a parent object. It then checks if the selected row index is greater than or equal to 0. If so, it retrieves the value at column 2 of the selected row and checks if it is equal to "Part Time" or "Full Time". Depending on the value, the code either sets the currentTableModel to the partTimeEmployeesTableModel or the fullTimeEmployeesTableModel from the parent object, respectively.
- After setting the currentTableModel, the code then executes a SQL DELETE statement to remove the selected row's data from a MySQL database named payrollsystemdb. The code creates a connection to the database, prepares and executes a SQL statement to delete the selected employee data using a PreparedStatement object, and then closes the connection.
- If the deletion is successful, the code then removes the selected row from the currentTableModel and clears the form fields in the parent object. Finally, the code displays a message dialog indicating that the deletion was successful. If the selected row index is less than 0, the code displays a message dialog asking the user to select a row.

DATABASE EXPLANATION

- This is where this code connects to a MySQL database using JDBC (Java Database Connectivity) API. The connection is established in the following lines of code.

```
                                        String sql = "DELETE FROM `employeetable`
WHERE `employeetable`.`employee_id` = ?;";
                                    // Database connection and insertion for
part-time employees
                                String url = "jdbc:mysql://localhost/";
                                String dbName = "payrollsystemdb";
                                String driver = "com.mysql.jdbc.Driver";
                                String userName = "root";
                                String password = "";

                                try (Connection conn =
DriverManager.getConnection(url+dbName, userName, password);
                                        Statement state = conn.createStatement();
                                        ) {

                                    PreparedStatement stmt =
conn.prepareStatement(sql);
                                    stmt.setInt(1,
Integer.parseInt(id.getText().toString()));
                                    stmt.executeUpdate();

//                                      int rowsAffected = ;

System.out.println(state.executeUpdate(sql));

                                        conn.close();

                                    } catch (SQLException el) {
                                        System.out.println(el.getMessage());
                                    }

//                              ******************************************************

                                        currentTableModel =
parentObject.fullTimeEmployeesTableModel;
                                    currentTableModel.removeRow(rowIndex);
                                    parentObject.clearFormFields();

                                        JOptionPane.showMessageDialog(null,
"Deleted Successfully");
                                }
                                }       else {
                                        JOptionPane.showMessageDialog(null, "Please
Select a Row");
                                }
                            }
                        });
```

- First, the necessary JDBC driver is loaded using the com.mysql.jdbc.Driver class. This is done using the following line of code:
  String driver = "com.mysql.jdbc.Driver";
  Class.forName(driver);
- After loading the driver, a connection to the MySQL database is established using the DriverManager.getConnection() method. The method takes three parameters: the database URL, the username, and the password. In this code, the following line of code is used to connect to the database:
  String url = "jdbc:mysql://localhost/";
  String dbName = "payrollsystemdb";
  String userName = "root";
  String password = "";
  Connection conn = DriverManager.getConnection(url + dbName, userName, password);
- After obtaining a connection, a PreparedStatement object is created using the SQL query DELETE FROM employeetable WHERE employee_id = ?;. This is done using the following lines of code:
  String sql = "DELETE FROM employeetable WHERE employee_id = ?;";
  PreparedStatement stmt = conn.prepareStatement(sql);
  stmt.setInt(1, Integer.parseInt(id.getText().toString()));
  stmt.executeUpdate();
- The PreparedStatement object is used to set the value of the parameter employee_id using the setInt() method. The value is retrieved from a GUI component with the id field.
- Finally, the stmt.executeUpdate() method is called to execute the SQL statement and delete the row from the database.
- The connection is then closed using the conn.close() method to free up resources and avoid memory leaks.
- Overall, the code uses JDBC to connect to a MySQL database, execute a DELETE query on a specific row, and close the connection once the operation is complete.

Update Button

```
//UPDATE BUTTON
JButton btnUpdate = new JButton("Update");
btnUpdate.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {

                if (
                        id.getText().equals("") ||
                        name.getText().equals("") ||
                        days.getText().equals("") ||
                        absence.getText().equals("") ||
                        rate.getText().equals("") ||
                        position.getSelectedItem().toString().equals("")
                ) {
                        JOptionPane.showMessageDialog(null, "Please Fill
Complete Information!");
                        return;
                }


                JTable focusedTable                             =
parentObject.focusedTable;
                int rowIndex                                                    =
focusedTable.getSelectedRow();
                String currentPositionValue             =
focusedTable.getValueAt(rowIndex, 2).toString(); // Value from selected row
                String newPositionValue                         =
position.getSelectedItem().toString(); // Value from combo box
                DefaultTableModel currentTableModel = model;
                String oldID = id.getText().toString();


                // If user decided to change the Employee Type value
                // from what is current selected.
                if (!currentPositionValue.equals(newPositionValue)) {

                        if (currentPositionValue.equals("Part Time")) {
                                currentTableModel = partTimeEmployeesTableModel;
                        } else {
                                currentTableModel = model;
                        }
                        // Since position value from combo box has changed
                        // remove from the current focused table and we will
                        // move to the other table.
                        currentTableModel.removeRow(rowIndex);

                        if (currentPositionValue.equals("Full Time")) {
                                currentTableModel = fullTimeEmployeesTableModel;
                        } else {
                                currentTableModel = model;
                        }
                        currentTableModel.removeRow(rowIndex);

                        // Transfer to another table
                        Object tableRow[] = {
                                id.getText(),
                                name.getText(),
                                newPositionValue,
                                days.getText(),
                                absence.getText(),
                                rate.getText(),
                                salary.getText()
                        };

                        if (newPositionValue.equals("Part Time")) {
                                currentTableModel = partTimeEmployeesTableModel;
                        } else {
                                currentTableModel = model;
                        }
```

```
                                        currentTableModel.addRow(tableRow);

                                        if (newPositionValue.equals("Full Time")) {
                                                currentTableModel = fullTimeEmployeesTableModel;
                                        } else {
                                                currentTableModel = model;
                                        }

                                        currentTableModel.addRow(tableRow);

                                } else {
                                        if (currentPositionValue.equals("Part Time")) {
                                                currentTableModel =
parentObject.partTimeEmployeesTableModel;
                                        } else if (currentPositionValue.equals("Full Time")) {
                                                currentTableModel =
parentObject.fullTimeEmployeesTableModel;
                                        }

                                        currentTableModel.setValueAt(id.getText(), rowIndex, 0);
                                        currentTableModel.setValueAt(name.getText(), rowIndex,
1);
                                        currentTableModel.setValueAt(currentPositionValue,
rowIndex, 2);
                                        currentTableModel.setValueAt(days.getText(), rowIndex,
3);
                                        currentTableModel.setValueAt(absence.getText(),
rowIndex, 4);
                                        currentTableModel.setValueAt(rate.getText(), rowIndex,
5);
                                        currentTableModel.setValueAt(salary.getText(), rowIndex,
6);
                                }

//                              SQL QUERY ************************************
                                String sql = "UPDATE "+
                                                "`employeetable` SET "+
                                                "`employee_name` = ?, `employee_position` = ?,
`employee_days` = ?, `employee_rate` = ?, `employee_absences` = ?, `employee_salary` = ? "+
                                                "WHERE `employeetable`.`employee_id`
= ?;";


                                // Database connection and insertion for part-time employees
                String url = "jdbc:mysql://localhost/";
                String dbName = "payrollsystemdb";
                String driver = "com.mysql.jdbc.Driver";
                String userName = "root";
                String password = "";

                try (Connection conn = DriverManager.getConnection(url+dbName, userName,
password);
                                Statement state = conn.createStatement();
                                ) {

                        PreparedStatement stmt = conn.prepareStatement(sql);
                        stmt.setString(1, name.getText().toString()); //employee_name
                        stmt.setString(2, newPositionValue); //employee_position
                        stmt.setInt(3, Integer.parseInt(days.getText().toString()));
//employee_days
                        stmt.setInt(4, Integer.parseInt(rate.getText().toString()));
//employee_rate
                        stmt.setInt(5, Integer.parseInt(absence.getText().toString()));
//employee_absences
                        stmt.setInt(6, Integer.parseInt(salary.getText().toString()));
//employee_salary

                        stmt.setInt(7, Integer.parseInt(oldID)); //employee_salary
                        stmt.executeUpdate();
```

```
                } catch (SQLException e1) {
                    System.out.println(e1.getMessage());
                }

//              ****************************************************

                    // Clear form fields
                    parentObject.clearFormFields();

                    //
                    parentObject.focusedTable.clearSelection();
                }
            });
```

This is a Java code block that contains an event listener for a JButton called "btnUpdate". When the button is clicked, the actionPerformed method is called, which performs several actions:

- Checks if all required fields are filled out. If not, it displays a message and returns from the method.
- Gets the currently selected row in a JTable component and saves the current position value and the new position value.
- If the position value has changed, it removes the current row from the current table model and adds a new row to the appropriate table model based on the new position value.
- If the position value has not changed, it updates the values in the current row of the appropriate table model.
- Executes an SQL query to update the employee record in the database with the new values.
- Clears the form fields and deselects the row in the JTable component.

The code is designed to work with a payroll system that manages employee records in a MySQL database. It updates the employee record in the database based on the information entered in the form fields.

DATABASE EXPLANATION:

This code block contains database connection and data manipulation code using JDBC (Java Database Connectivity) API for MySQL database.

The following lines establish the database connection:

// Database connection and insertion for part-time employees

String url = "jdbc:mysql://localhost/";

String dbName = "payrollsystemdb";

String driver = "com.mysql.jdbc.Driver";

String userName = "root";

String password = "";

try (Connection conn = DriverManager.getConnection(url+dbName, userName, password);

Statement state = conn.createStatement();

) {

- Here, url specifies the database URL, dbName specifies the database name, userName and password are the credentials to access the database.
- The DriverManager class is used to get the connection with the database, and getConnection method is used to establish a connection with the database.
- After establishing a connection, a prepared statement is created using the SQL query for updating the database table:

PreparedStatement stmt = conn.prepareStatement(sql);

stmt.setString(1, name.getText().toString()); //employee_name

stmt.setString(2, newPositionValue); //employee_position

stmt.setInt(3, Integer.parseInt(days.getText().toString())); //employee_days

stmt.setInt(4, Integer.parseInt(rate.getText().toString())); //employee_rate

stmt.setInt(5, Integer.parseInt(absence.getText().toString())); //employee_absences

stmt.setInt(6, Integer.parseInt(salary.getText().toString())); //employee_salary

stmt.setInt(7, Integer.parseInt(oldID)); //employee_salary

stmt.executeUpdate();

- Here, sql contains the SQL query to be executed. Then, the values to be updated are set using the setString and setInt methods of the PreparedStatement class. Finally, the executeUpdate method is called to execute the SQL query.
- The try-catch block is used to handle exceptions that may occur during the database connection and query execution process.
- Overall, this code is updating the database table with the values entered in the GUI form.