**UNIVERSITY OF TECHNOLOGY AND EDUCATION**

**FACULTY OF HIGH QUALITY TRAINING**



**Instructor: Prof. Nguyễn Đăng Quang**

**PROJECT 1**

**PROJECT REPORT**

**PROJECT: REMOTE HUMAN RESOURSE MANAGEMENT**

**Ngũ Quốc Phong – 17110060**

**Trần Minh Quang – 17110067**

**1st SEMESTER, SCHOOL YEAR: 2019 – 2020**

**HO CHI MINH CITY – 01/12/2019**

# INDEX

Contents

[INDEX 1](#_Toc27011052)

[I. Description 2](#_Toc27011053)

[a. Project’s goal 2](#_Toc27011054)

[b. Data, input 2](#_Toc27011055)

[c. Situations 2](#_Toc27011056)

[d. Interface 3](#_Toc27011057)

[II. Task allocation 4](#_Toc27011058)

[III. Design 4](#_Toc27011059)

[1. Algorithm 4](#_Toc27011060)

[2. Class design 5](#_Toc27011061)

[3. Database design 12](#_Toc27011062)

[4. Fields 14](#_Toc27011063)

[5. Interface design 16](#_Toc27011064)

[IV. Setting and testing 18](#_Toc27011065)

[V. Conclusion 20](#_Toc27011066)

## Description

### Project’s goal

This software is designed to manage a Human Resource database written in MySQL remotely. This software is used when an employee wants to check their information, check attendance when they signed in. Moreover, it was designed for anyone with administrator rights to view, add, delete, edit any records related to the company.

The following is used in the application

* Java Programming Language.
* JDBC API.
* Java Swing UI Framework.

### Data, input

The software will receive data from the designed MySQL database and the software will receive input from user to interact with the database.

### Situations

* + 1. Admin wants to search information on the database.
    2. Admin wants to add data to the database.
    3. Admin wants to delete data from the database.
    4. Employee wants to see their information on the database.
    5. Employee wants to check attendance (With proper permission).

### Interface

## Task allocation

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Full Name | Accountability | Percent contribution |
| 17110060 | Ngu Quoc Phong | Leader | 60% |
| 17110067 | Tran Minh Quang | Assistant | 40% |

## Design

### Class design

#### Classes

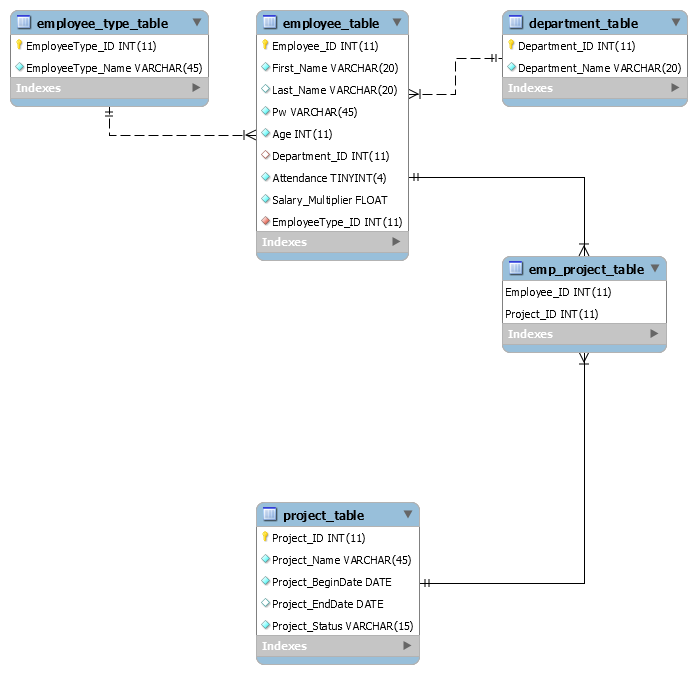
|  |  |  |
| --- | --- | --- |
|  | Class | Purpose |
| 1 | Mainframe.java | This class is the starting point of the application, it’s where all the necessary component will be created and added inside the JFrame in which this class is responsible for. |
| 2 | LoginPanel.java | This class job itself is a login form represented in JPanel form.  In this class the user can either login as Admin or as an Employee, of course with proper account permission. |
| 3 | EmployeePanel.java | This class purpose is to ensure the Employee that logged in can view all their information related to themselves and the company and of course it’s also stored in JPanel. |
| 4 | AdminPanel.java | This class is designed for the user with the Administrator permission only, in this JPanel the user can view various information of their Employee, Project, Department and can also add or delete as well. |
| 5 | DataFrame.java | This class is directly linked to the Admin Panel and it’s called to execute the task of Adding or Deleting various records, It’s in JFrame format. |
| 6 | SQLConnect.java | This class is the how the application is linked to database, it’s also where all various functions of retrieving and modifying is created. |
| 7 | Main.java | This class is to run the application. |

#### Protocols

|  |  |  |  |
| --- | --- | --- | --- |
|  | Protocol | Purpose | File name, order, declaration line |
| 1 | **StartGUI()**  **Input:** None  **Output:** The application interface. | Initialize all the UI Components and display the UI itself | Mainframe constructor (20) and is ran in the Main.java (9) |
| 2 | **OpenPanel(int panelID)**  **Input:** Panel ID of a panel  **Output:** The Panel with the corresponding ID. | Set the value for the Boolean of the OpenSelf() function. | MainFrame.java (45)  LoginPanel.java (123, 129) |
| 3 | **openself(Boolean val)**  **Input:** Corresponding value that has been set by OpenPanel.  **Output:** Set the panel visible else invisible.  **Pseudo Code:**   * IF Val == PanelID Return True else False. * THEN JPanel.setVisible(Returned Value). | Set the JPanel Visible or Invisible. | Mainframe.java (52,53,54) |
| 4 | **getAdminPanel()**  **Input:** None.  **Output:** return AdminPanel JPanel. | Basic encapsulation of the AdminPanel JPanel. | MainFrame.java (42) |
| 5 | **getEmployeePanel()**  **Input:** None.  **Output:** return EmployeePanel JPanel. | Basic encapsulation of the EmployeePanel JPanel. | MainFrame.java (43) |
| 8 | **getLoginPanel()**  **Input:** None.  **Output:** return LoginPanel JPanel. | Basic encapsulation of the LoginPanel JPanel. | MainFrame.java (41) |
| 7 | **setComponentState(Component component, boolean enabled)**  **Input:** Component you wishes to enable or disable.  **Output:** Disable all Child components of the Input Component.  **Pseudo code:**   * SET component Enable or Disable * IF component is a container * THEN * FOR all the child components of the container. * SET Child components enabled or disabled. | To set a Component and all of it’s Child components disabled should it be a container.  SOURCE: Stackoverflow from user Tom Hawtin - tackline. | AdminPanel.java (244,249) |
| 8 | **setData()**  **Input:** None  **Output:** Data of employee that has logged in.  **Pseudo Code:**   * IF A static boolean from class **SQLConnect.java** return 'True' * THEN various JLabel receive the corresponding Data of the employee. | To get the information of the logged in employee. | LoginPanel.java (122) |
| 9 | **getConnectionLogin(JTextfield textField, JPasswordField passwordField)**  **Input:** JTextfield for the username, JPassword for the password.  **Output:** The boolean value for whether that username and password is in the database, and store the information of the employee logged in.  **Pseudo code:**   * Create temporary storage for username and password. * Create the query. * Get the connection. * Set the Query to the PrepareStatement. * Set the Parameter Index for username and password storage. * IF (The Next resultset == the input JTextfield, Jpassword) * Then set the static variable Login = true * Add the information of the Resultset into a static list. * Return the connection. | Use for a login verification of the login form, the get the data for the employee panel. | LoginPanel.java  (117) |
| 10 | **getConnectionTable (int tableNo, JTable table, DefaultTableModel model)**  **Input:** The number order of the table data you want, The JTable you want to store the information, the table model.  **Output:** The table you want to access.  **Pseudo code:**   * Get Connection * Switch (tableNo) * IF tableNo == 1 * Query * Get All Employee information from the database into the model. * IF tableNo == 2 * Query * Get All Project information from the database into the model. * IF tableNo == 3 * Query * Get All Department information from the database into the model. * IF tableNo == 4 * Query * Get All and Mixed information of the employee, department, project from the database into the model. * THEN set the table a new model. * Set the table disabled to prevent editing. | To get the desired table to be displayed on the JTable. | DataFrame.java  (48, 56, 65) |
| 11 | **getConnectionInsert(int tableNo)**  **Input:** The number order of the table data you want to insert data to.  **Output:** The Prompts for the user to input data to.  **Pseudo Code:**   * Get the connection. * Switch(tableNo) * IF tableNo == 1 * Query * Prompt the user to input wanted information to add to Employee table. * IF tableNo == 2 * Query * Prompt the user to input wanted information to add to Project table. * IF tableNo == 3 * Query * Prompt the user to input wanted infromation to add to Department table. * Return the connection. | For the user to input desired information to insert to the database respective table. | DataFrame.java  (48, 56, 65) |
| 12 | **getConnectionDelete(int tableNo)**  **Input:** The number order of the table data you want to delete data to.  **Output:** A Prompt for the user to delete data from.  **Pseudo Code:**   * Get the connection. * Switch(tableNo) * IF tableNo == 1 * Query * Prompt the user to input wanted information to delete from Employee table. * IF tableNo == 2 * Query * Prompt the user to input wanted information to delete from Project table. * IF tableNo == 3 * Query * Prompt the user to input wanted infromation to delete from Department table. * Return the connection. | For the user to input desired information to be deleted from the database respective table. | DataFrame.java  (72,79,86) |
| 13 | **searchBoxFunction(String text)**  **Input:** Text want to filter on table.  **Output:** Filtered table.  **Pseudo Code:**   * IF text trimmed length == 0 * SET rowsorter rowfilter = null * ELSE * SET rowsorter rowfilter = regular expression filter case-insensitive + text. | Act as a search function for the search box.  SOURCE: Stackoverflow  user: Paul Samsotha | AdminPanel.java  (159, 175) |

### Database design

#### Entity – Relationship Diagram



#### Table description

|  |  |  |
| --- | --- | --- |
|  | Table name | Explaination |
| 1 | Employee\_table | Store information about employees, contain personal data and login info |
| 2 | Employee\_type\_table | List of employee type can be set for each employee, all new insert to employee\_table is set default to low-priority. |
| 3 | Project\_table | Store projects in the company from past to presents, also display status of each project (finished, on-going, suspended) |
| 4 | Employee\_project\_table | Display which employee is currently working on, or have worked in which project, one project can be anticipated by multiple employees. |
| 5 | Department\_table | Every employee belongs to one of the departments listed in this table. |

### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| Table | Field name | Data type | Purpose |
| Employee | Employee\_ID | INT(11) | Unique ID of each employee |
| First\_name | VARCHAR(20) | First name of each employee, also |
| Last\_name | VARCHAR(20) | Last name of each employee |
| Pw | VARCHAR(45) | Password to login |
| Age | INT(11) | Employee’s age |
| Department\_ID | INT(11) | FK, show which department an employee belongs to |
| Attendance | TINYINT(4) | Check employee’s attendance daily, default to ‘0’  \*TINYINT to replace BIT data type\* |
| Salary\_multiplier | FLOAT | Each employee has different salary rate, default to 1.0 |
| Employee\_type\_ID | INT(11) | Set employee’s priority level |
| Employee\_type | Employee\_type\_ID | INT(11) | Unique ID of each employee type |
| EmployeeType\_Name | VARCHAR(45) | Employee type name |
| Project | Project\_ID | INT(11) | Unique ID of each project |
| Project\_Name | VARCHAR(45) | Project name |
| Project | Project\_BeginDate | DATE | Begin date of a project |
| Project\_EndDate | DATE | End date of a project, can be null if project is still in progress/suspended |
| Project\_Status | VARCHAR(15) | Current status of a project |
| Emp\_project | Employee\_ID | INT(11) | ID of employee |
| Project\_ID | INT(11) | ID of project |
| Department | Department \_ID | INT(11) | Unique ID of a department |
| Department\_Name | VARCHAR(45) | Department name |

### Interface design

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Screen/Window/Dialog | Purpose | Description |
| 1 | Login Window | For user to login. | Allow user to login, Administrator login as administrator, employee login as employee. |
| 2 | Employee Window | For user to check their information and check attendance. | A brief description of the employee who logged in, their image, and check attendance. |
| 3 |  | For user to view, search, and access to the edit frame. | An overall view of the database of the server  allow user to search, access to the edit frame. |
| 4 | Edit frame | Allow user to add or delete. | This frame designed not only for user to view the chosen table but it's also for interacting with the table itself. |
| 5 | Add dialog | Accept user input to add the information to the database. | Once clicked on the add button these similar prompts will popup and accept user input. |
| 6 | Delete dialog | Accept user input to delete the information from the database. | Once clicked on the delete button these similar prompts will popup and accept user input. |

## Setting and testing

Situations:

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Situation. | Purpose | Why choose that situation. |
| 1 | Login Form  Input  username:  jon/Jon/JON  Password:  jon/JON/jon | To see if the application recognize case sensitive input or not. | This situation was to test whether user can login with their password be not case sensitive, said if the password was jon if the user inputted JON then the application would still accept that and proceed to login.  (Test successful, application does allow user to login, was fixed using Query) |
| 2 | Admin View form  Input to search: !@#!@$@#%@#111. | To see if the application breaks on special and integer input. | This situation was to test whether the application breaks on the input of integer or special. (It did not break because this is nothing but a string and the code running for this was nothing but a regular expression filter). |
| 3 | Edit Frame  Input to Add dialog  Employee ID "asdasd#@". | To see if the application breaks on a string input on an integer data type. | This situation was to test whether the application breaks on the input of string or special. (It did not break but produces an error in the console) to fix simply implement a pattern check for any input that is not a number (Is not implemented). |
| 4 | Edit Frame  Input to Delete dialog  Employee ID "@!#@!#asdasd". | To see if the application breaks on a string input on an integer data type. | This situation was to test whether the application breaks on the input of string or special. (It did not break but produces an error in the console) to fix simply implement a pattern check for any input that is not a number (Is not implemented). |
| 5 | Edit Frame  Cancel Add/Delete dialogs | To see if the application breaks on a null input. | This situation was to test whether the application breaks on the null input. (It did not break but produces an error in the console) to fix simply implement a pattern check for any input that is not null |

## Conclusion

Overall, we would say this project is about 70% completed with a few missing features:

* 1. Employee cannot check attendance because it is not implemented.
  2. The ability to edit the tables is not implemented.
  3. The ability to assign Employees to Departments, Projects is not implemented.
  4. Missing various check for input.
  5. Missing a model class for various implementations.

Difficulty during development:

1. Overall UI design, the UI design went through a lot of redesign due to the limitations of Swing but ultimate, through a lot of researching, came out pretty decent.
2. Studying how to UI work was pretty difficult because layout, however after understanding and using various set of layouts we were able to work with it.
3. One of the main problem was how to prevent the application constantly creating new object when opening a JPanel, to workaround that we were forced to find a way to initialize all the objects (JPanels) and have functions to set visible.
4. Querying was mostly easy except when working with the mixed table due to the amount of table of merging and also to prevent duplicate.

References:

* javaguides.net
* [j](https://www.javatpoint.com/java-swing)avatpoint.com/java-swing
* stackoverflow.com