ASSIGNMENT 1 FRONT SHEET

Qualification	**BTEC Level 5 HND Diploma in Computing**		
Unit number and title			
Submission date	26/08/2023	**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	
Student Name	Dinh Dinh Khoi	**Student ID**	GCC210345
Class	GCC1003	**Assessor name**	Nguyen Kim Khanh
Student declaration I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.			
		Student's signature	

Grading grid

Grade (0-10)

❒ **Summative Feedback:**		❒ **Resubmission Feedback:**
Grade:	**Assessor Signature:**	**Date:**
IV Signature:		

Assessment Brief

Student Name/ID Number	
Unit Number and Title	**Object Oriented Programming with Java**
Academic Year	2020 - 2021
Unit Tutor	
Assignment Number & Title	**Design, Implement and Test a GUI application**
Issue Date	
Submission Date	
IV Name & Date	

Submission Format
The submission is in the form of a written report. This should be written in a concise, formal business style using single spacing and font size 12. You are required to make use of headings, paragraphs and subsections as appropriate, and all work must be supported with research and referenced using the Harvard referencing system. Please also provide a bibliography using the Harvard referencing system.

Unit Learning Outcomes
LO1 Understand basic programming skills and OOP paradigm **LO2** Understand how to detect errors and handle errors **LO3** Understand how to working with files in applications **LO4** Understand how to build GUI application
Assignment Brief

You have to develop an application to solve a small business problem. The problem requires a graphical user interface with features that required reading / writing data from text file, working with a collection of data (searching for item / min / max / sum / etc.). The application must handle errors so that it will not crash at end user side. The application also need to be fully tested before the production phase.
You need to write a technical report about the development of the application. Content of the report should cover design, implementation and testing.
In the end you need to demo your application, explain your code and answer technical questions.

Learning Outcomes and Assessment Criteria
LO1 Understand basic programming skills and OOP paradigm **LO2** Understand how to detect errors and handle errors **LO3** Understand how to working with files in applications **LO4** Understand how to build GUI application
To get Pass (5 – 6.5 points) - Student can design and implement GUI for the application solve a specific problem - Student knows how to load and save data from file. - Student knows how to handle errors by using exceptions - Student knows how to write test plan, execute test cases and log results.
To get Merit (**7 – 8.5 points**) - The application is well designed, user friendly and has logical flow of actions. - Can apply MVC in the application, can apply JUnit to test automatic - Errors are well handle to avoid program crashing, the test can cover as many as possible the errors in program
To get Distinction (9 – 10 points) The application must show excellent design & implementation, runs without any errors, all inputs are validated, all errors are well handled including recover choice, rich features showing unique ideas, algorithms.

Table of Contents

I. Introduction.

KD Store, a retailer of various smartphone models, is planning to build a branch in Can Tho. They wanted us to develop an application so that administrators may manage branches more easily since they were having trouble creating a product and personnel management system. Staff can only handle products using an account that has been given to them by administrator, while administrators may manage both employee accounts and goods.

II. Requirement.

After agreeing on the requirements with the customer, I summarize the specific requirements as follows:

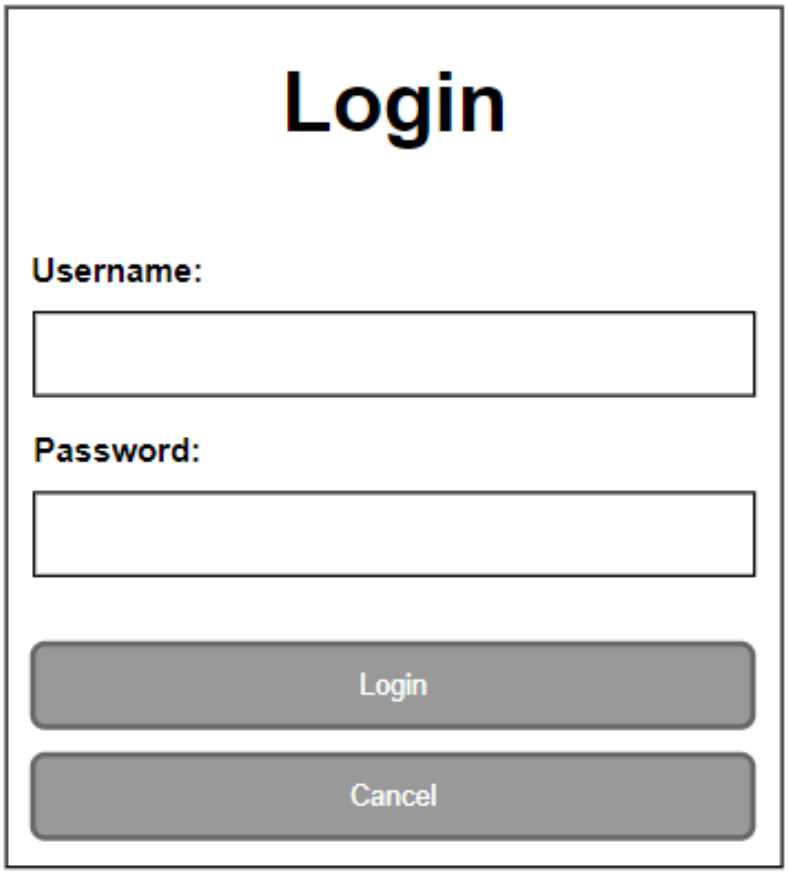
- The administrator who wants to enter the management page must enter the correct "Username" and "Password" of the administrator into the login interface.
    - Username
    - Password
- Administrators manage employee accounts through the "Staff Account" interface, where administrators can Create accounts, Update accounts, and Delete staff's accounts.
    - Username
    - Password
    - Staff name
    - Date birth
    - Gender
    - Address
    - Phone number
- Administrators and staff manage products through the “Phone Management” interface, where administrators and employees can Add new products, Update product information, and Delete products.
    - Phone ID
    - Phone name
    - Describe
    - Quantity
    - Price

- The program has the following functions:
  - Create/Add: used to create/add new employee and product accounts.
  - Update: used to update product information of staff accounts, products.
  - Delete: used to delete products and unused accounts.

III. Design.

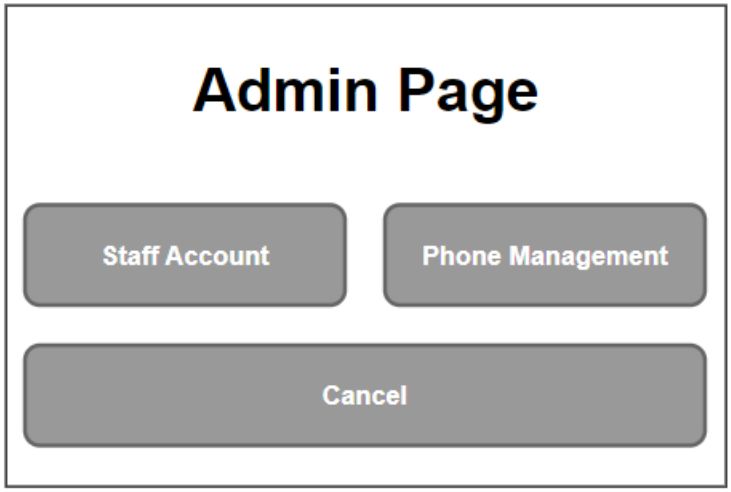
1. UI design.

- ***This is the Login interface.***



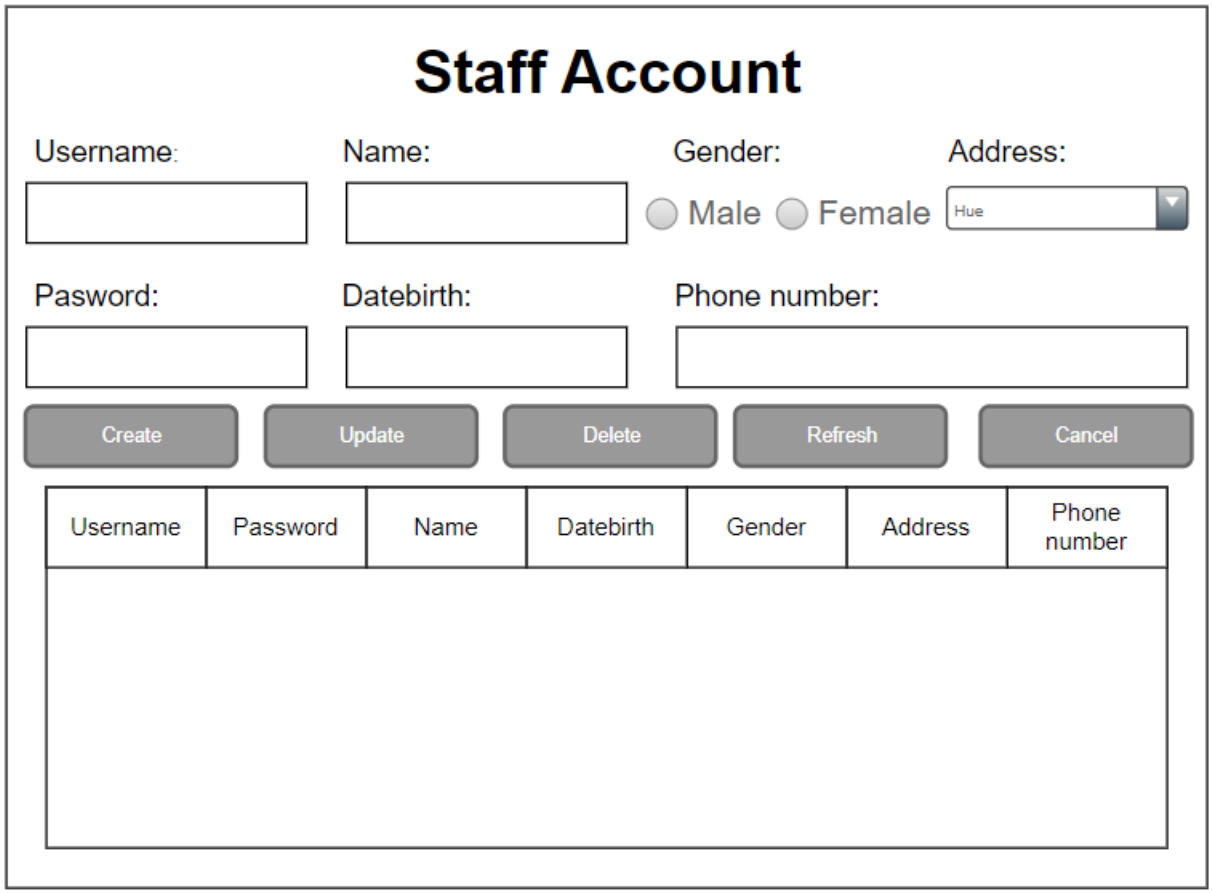
Login interface.

- ***This is Admin interface.***



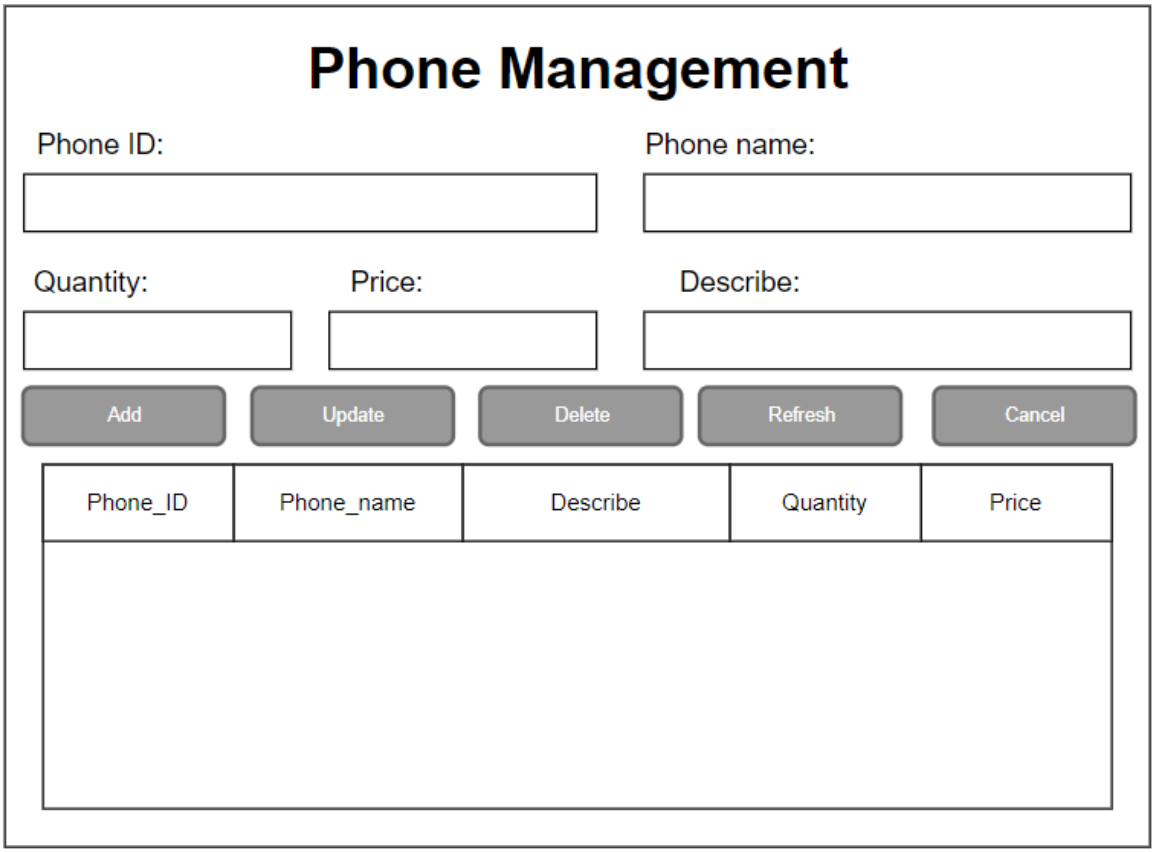
Admin page (only for admin).

❖ ***This is Staff Account interface.***



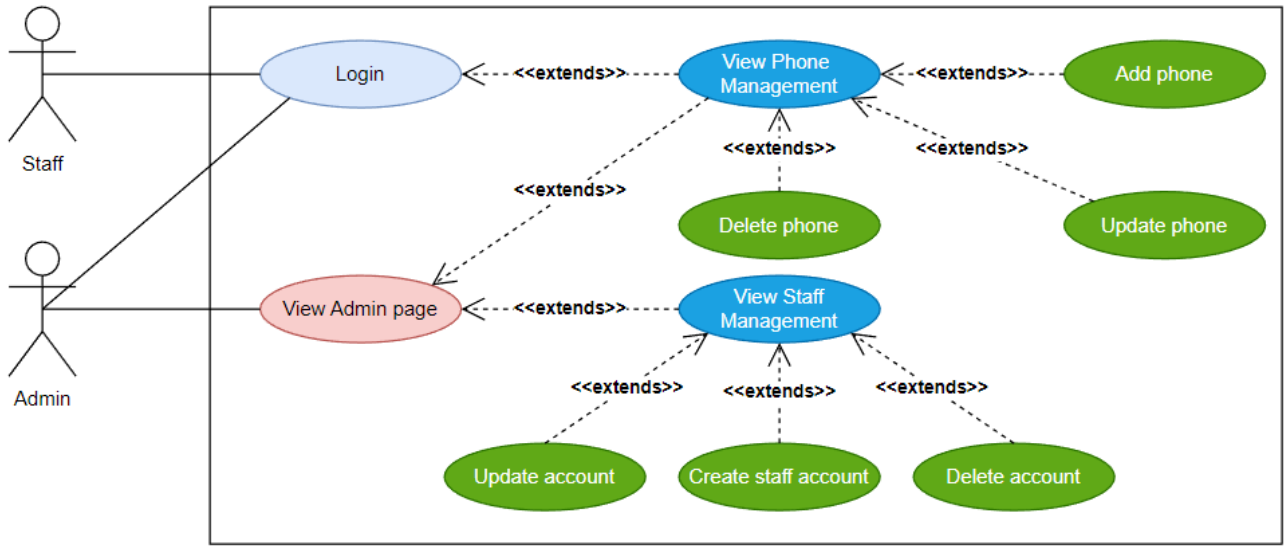
Staff Account page (Admin is used to manage employee accounts).

❖ ***This is Phone Management interface.***



Phone Management page used for product management.

2. Use case.

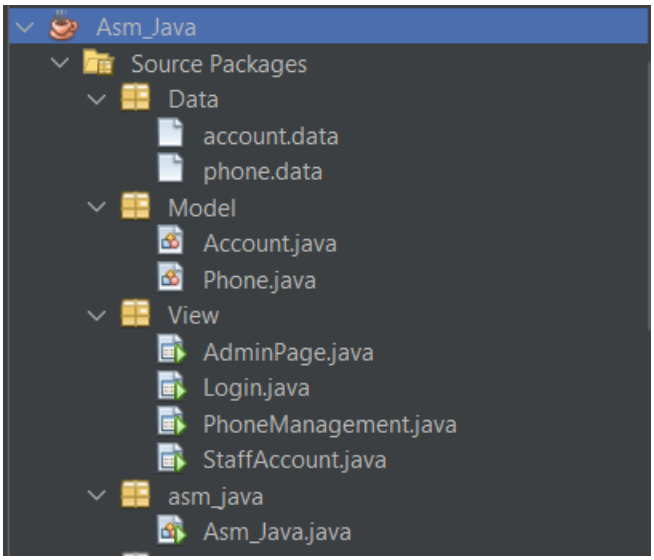


Use case diagram of project.

IV. Implementation.

1. Project Structure.

My project consists of 4 main Source Packages: Data, Model, View and asm_java. Data is a package used to store account information and product information. Model is used to hold the Phone and Account objects. View contains the Login interface, Admin interface, Phone Management interface and Staff Account interface. Finally, asm_java contains the main functionality of the project.



2. Explain class.

a. Phone class.

Phone class are used to get product information. In this class, I import java.io.Serializable so that the class can receive and run the program's interface. The implement keyword is an important keyword for the interface to be able to receive objects. Next, create the property variables for the class.

```
import java.io.Serializable;
```

```
public class Phone implements Serializable{
    private String phoneID, phoneName, describe;
    private String quantity;
    private String priceString;
```

The constructor creates a data object on the property variables declared above to receive data about the product's information.

```
public Phone(String phoneID, String phoneName, String describe, String quantity, String priceString) {
    this.phoneID = phoneID;
    this.phoneName = phoneName;
    this.describe = describe;
    this.quantity = quantity;
    this.priceString = priceString;
}
```

The set functions are responsible for taking product information entered by the user and assigning them to the properties of the object. At the same time, the get function is used to get data from the object when the user needs it.

```
public void setPhoneID(String phoneID) {
    this.phoneID = phoneID;
}

public void setPhoneName(String phoneName) {
    this.phoneName = phoneName;
}

public void setDescribe(String describe) {
    this.describe = describe;
}

public void setQuantity(String quantity) {
    this.quantity = quantity;
}

public void setPriceString(String priceString) {
    this.priceString = priceString;
}

public String getPhoneID() {
    return phoneID;
}

public String getPhoneName() {
    return phoneName;
}

public String getDescribe() {
    return describe;
}

public String getQuantity() {
    return quantity;
}

public String getPriceString() {
    return priceString;
}
```

b. Account class.

Account class are used to get staff account information. I import java.io.Serializable so that the class can receive and run the program's interface. The implement keyword is an important keyword for the interface to be able to receive objects. Next, create the property variables for the class.

```
import java.io.Serializable;
```

```
public class Account implements Serializable{
    private String userName, passWord, staffName, dateBirth;
    private boolean gender;
    public String address, phoneNumber;
```

The constructor creates a data object on the property variables declared above to receive data about the staff account's information.

```
public Account(String userName, String passWord, String staffName,
        String dateBirth, boolean gender, String address,
        String phoneNumber) {
    this.userName = userName;
    this.passWord = passWord;
    this.staffName = staffName;
    this.dateBirth = dateBirth;
    this.gender = gender;
    this.address = address;
    this.phoneNumber = phoneNumber;
}
```

The set functions are responsible for taking staff account information entered by the user and assigning them to the properties of the object. At the same time, the get function is used to get data from the object when the user needs it.

```java
    public void setUserName(String userName) {
        this.userName = userName;
    }

    public void setPassWord(String passWord) {
        this.passWord = passWord;
    }

    public void setStaffName(String staffName) {
        this.staffName = staffName;
    }

    public void setDateBirth(String dateBirth) {
        this.dateBirth = dateBirth;
    }

    public void setGender(boolean gender) {
        this.gender = gender;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }

    public String getUserName() {
        return userName;
    }

    public String getPassWord() {
        return passWord;
    }
```

```java
    public String getStaffName() {
        return staffName;
    }

    public String getDateBirth() {
        return dateBirth;
    }

    public boolean isGender() {
        return gender;
    }

    public String getAddress() {
        return address;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }
```

3. Explain code:

- **Login button:**

Firstly, declare two variables user and pass to receive 2 values username and password.

```
private void btn_LoginActionPerformed(java.awt.event.ActionEvent evt) {
    String user = txt_User.getText();
    String pass = new String(value:txt_Pass.getPassword());
```

The "for-each" statement has a function to create two ArrayList containing the staff's username and password:

```
for(Account a : accList){
    userList.add(e: a.getUserName());
    passList.add(e: a.getPassWord());
}
```

StringBuffer and "if" statements are used to check if username and password are empty or not:

```
//Check Null
StringBuffer sb = new StringBuffer();
if (user.equals(anObject: "")) {
    sb.append(str: "\nUser name is empty!");
}
if (pass.equals(anObject: "")) {
    sb.append(str: "\nPassword is empty!");
}
if (sb.length() > 0) {
    JOptionPane.showMessageDialog(parentComponent: this, message: sb.toString(), title: "Invalidation!",
            messageType: JOptionPane.ERROR_MESSAGE);
    return; //end of this action
}
```

The login button implements a function to check if the login account is indeed a valid account by using an “if-else if” statement to check the conditions.

```
if (user.equals(anObject: "Admin") && pass.equals(anObject: "Admin123@")) {
    JOptionPane.showMessageDialog(parentComponent: this, message: "Login successfully! \n Welcome Admin!");

    this.setVisible(b: false);
    AdminPage admin = new AdminPage();
    admin.setVisible(b: true);
}else if(searchUser(user: txt_User.getText()) < 0){
    JOptionPane.showMessageDialog(parentComponent: this, message: "Username or Password incorrect!");
}else{
    int p = searchUser((txt_User.getText()));
    int i = txt_Pass.getPassword().toString().compareTo(anotherString: passList.get(index: p));
    if(i != 0){
        JOptionPane.showMessageDialog(parentComponent: this, message: "Login success");
        this.setVisible(b: false);
        PhoneManagement phone = new PhoneManagement();
        phone.setVisible(b: true);
    }else{
        JOptionPane.showMessageDialog(parentComponent: this, message: "pass not correct");
    }
}
}
```

- ***Cancel button:*** The function of the Cancel button is to close the program when the user wants to exit the application.

```
private void btn_CancelActionPerformed(java.awt.event.ActionEvent evt) {
    int choice = JOptionPane.showConfirmDialog(parentComponent: this, message: "Do you want to exit?",
            title: "Confirmation", optionType: JOptionPane.YES_NO_OPTION, messageType: JOptionPane.QUESTION_MESSAGE);
    if(choice == JOptionPane.YES_OPTION){
        System.exit(status: 0);
    }
}
```

First, declare a variable choice to receive the application exit confirmation message. If choice = YES_OPTION it will close the application immediately.

- showDetail: perform the function of displaying the data of the selected object in the table.

```
//Show Detail
private void showDetail(int r){
    String phoneID = (String) tb_Phone.getValueAt(row: r, column: 0);
    txt_PhoneID.setText(t: phoneID);
    String phoneName = (String) tb_Phone.getValueAt(row: r, column: 1);
    txt_PhoneName.setText(t: phoneName);
    String describe = (String) tb_Phone.getValueAt(row: r, column: 2);
    txt_Describe.setText(t: describe);
    String quantity = (String) tb_Phone.getValueAt(row: r, column: 3);
    txt_Quantity.setText(t: quantity);
    String price = (String) tb_Phone.getValueAt(row: r, column: 4);
    txt_Price.setText(t: price);
}
```

Show detail function of Phone Management interface.

Declare the variables that receive data values in the table in turn. (Show detail of Staff Account interface is the same too).

```
//Show Detail
private void showDetail(int r){
    String user = (String) tb_Account.getValueAt(row: r, column: 0);
    txt_User.setText(t: user);
    String pass = (String) tb_Account.getValueAt(row: r, column: 1);
    txt_Pass.setText(t: pass);
    String name = (String) tb_Account.getValueAt(row: r, column: 2);
    txt_Name.setText(t: name);
    String date = (String) tb_Account.getValueAt(row: r, column: 3);
    txt_Date.setText(t: date);
    Boolean gender = (Boolean) tb_Account.getValueAt(row: r, column: 4);
    rd_Male.setSelected(b: gender);
    rd_Female.setSelected(!gender);
    String address = (String) tb_Account.getValueAt(row: r, column: 5);
    cb_Address.setSelectedItem(anObject: address);
    String phone = (String) tb_Account.getValueAt(row: r, column: 6);
    txt_Phone.setText(t: phone);
}
```

Show detail function of Staff Account interface.

- clickDetail: has the function of displaying selected line information on the screen.

```
//Click Detail
private void clickDetail(){
    chosenRow= tb_Phone.getSelectedRow();
    showDetail(r: chosenRow);
}
```

Click detail of Phone Management.

First assign the data of the selected row to the chosenRow variable. Then call the showDetail function to display the data of the selected chosenRow on the screen. (Click detail of Staff Account interface is the same too).

```
//Click Detail
private void clickDetail(){
    chosenRow= tb_Account.getSelectedRow();
    showDetail(r: chosenRow);
}
```

Click detail of Staff Account interface.

- initTable: create table. Initialize the String[] array to get the data fields of the object. Next, initialize the object table and then assign the data fields to the table.

```
//Create Table
public void initTable(){
    String[] collumnName = {"Phone_ID", "Phone_Name", "Describe",
        "Quantity", "Price"};
    tbModel = new DefaultTableModel(columnNames: collumnName, rowCount: 0);
    tb_Phone.setModel(dataModel: tbModel);
}
```

initTable of Phone Management.

```
//Create Table
public void initTable(){
    String[] collumnName = {"Username", "Password", "Name", "Datebirth",
        "Gender(Male)", "Address", "Phone_Number"};
    tbModel = new DefaultTableModel(columnNames: collumnName, rowCount: 0);
    tb_Account.setModel(dataModel: tbModel);
}
```

initTable of Staff Account.

- initAddress: create combo box. Initialize the combo box object, then create a String[] array to get the staff address. Next, use a for-each loop to assign the values of the String[] array to the combo box.

```
//Create Combobox Address
public void initAddress(){
    cbModel = new DefaultComboBoxModel();
    String [] address = {"Huế", "Đà Nẵng", "Quy Nhơn",
    "Nha Trang", "Cam Ranh", "Đà Lạt",
    "Vũng Tàu", "Mỹ Tho", "Cần Tho"};
    for(String t : address){
        cbModel.addElement(anObject: t);
    }
    cb_Address.setModel(aModel: cbModel);
}
```

- fillToTable: add Phone/Staff account data to the table using for-each loop. The for-each loop is used to assign product/employee information to an object. Then add the object's data to the table.

```
//Fill To Table
private void fillToTable(){
    tbModel.setRowCount(rowCount: 0);
    for(Phone p : phoneList){
        Object[] row = new Object[]{p.getPhoneID(), p.getPhoneName(), p.getDescribe(), p.getQuantity(), p.getPriceString()};
        tbModel.addRow(rowData: row);
    }
}
```

Fill to table of Phone Management.

```
//Fill To Table
private void fillToTable(){
    tbModel.setRowCount(rowCount: 0);
    for(Account a : accountList){
        Object[] row = new Object[]{
            a.getUserName(),a.getPassWord(), a.getStaffName(), a.getDateBirth(), a.isGender(),
            a.getAddress(), a.getPhoneNumber()};
        tbModel.addRow(rowData: row);
    }
}
```

Fill to table of Staff Account.

- addPhone/ addAccount: There is a function to add new/ create products/ staff accounts. Use if-else if statement to check input conditions, format, etc. If none of the conditions are satisfied, create a new object to save the data to the file and display it in the table.

```java
//Add Phone
private void addPhone(){
    if(txt_PhoneID.getText().equals(anObject: "")){
        JOptionPane.showMessageDialog(parentComponent: this, message: "Phone ID cannot null!");
    }else if(isMixedID() == false){
        JOptionPane.showMessageDialog(parentComponent: this, message: "Phone ID must be written as GCxx ( x is a number)!");
    }else if(existID(str: txt_PhoneID.getText()) == true){
        JOptionPane.showMessageDialog(parentComponent: this, message: "Phone ID already exist!");
    }else if(txt_PhoneName.getText().equals(anObject: "")){
        JOptionPane.showMessageDialog(parentComponent: this, message: "Phone name cannot null!");
    }else if(existName(str: txt_PhoneName.getText()) == true){
        JOptionPane.showMessageDialog(parentComponent: this, message: "The phone name already exist!");
    }else if(txt_Describe.getText().equals(anObject: "")){
        JOptionPane.showMessageDialog(parentComponent: this, message: "Describe cannot null!");
    }else if(txt_Quantity.getText().equals(anObject: "")){
        JOptionPane.showMessageDialog(parentComponent: this, message: "Quantity cannot null!");
    }else if(isNumber() == false){
        JOptionPane.showMessageDialog(parentComponent: this, message: "Quantity must be a number!");
    }else if(txt_Price.getText().equals(anObject: "")){
        JOptionPane.showMessageDialog(parentComponent: this, message: "Price cannot null!");
    }else if(isPrice() == false){
        JOptionPane.showMessageDialog(parentComponent: this, message: "Price must be written as xx,xxx,xxx!");
    }else{
        Phone p = new Phone(
            phoneID: txt_PhoneID.getText(), phoneName: txt_PhoneName.getText(), describe: txt_Describe.getText(),
            quantity: txt_Quantity.getText(), priceString: txt_Price.getText());
        phoneList.add(e: p);
    }
}
```

Add Phone.

```java
//Add Account
private void addAccount(){
    if(txt_User.getText().equals(anObject: "")){
        JOptionPane.showMessageDialog(parentComponent: this, message: "Username cannot null!");
    }else if(existUser(str: txt_User.getText()) == true){
        JOptionPane.showMessageDialog(parentComponent: this, message: "Username already exist!");
    }else if(txt_Pass.getText().equals(anObject: "")){
        JOptionPane.showMessageDialog(parentComponent: this, message: "Password cannot null!");
    }else if(txt_Name.getText().equals(anObject: "")){
        JOptionPane.showMessageDialog(parentComponent: this, message: "Staff name cannot null!");
    }else if(isString() == false){
        JOptionPane.showMessageDialog(parentComponent: this, message: "The name must be String type!");
    }else if(txt_Phone.getText().equals(anObject: "")){
        JOptionPane.showMessageDialog(parentComponent: this, message: "Phone numer cannot null!");
    }else if(existPhone(str: txt_Phone.getText()) == true){
        JOptionPane.showMessageDialog(parentComponent: this, message: "Phone number already exist!");
    }else if(IsPhoneNumber() == false){
        JOptionPane.showMessageDialog(parentComponent: this, message: "The format phone must be 10 number start at 0!");
    }else if(isDay() == true){
        Account a = new Account(userName: txt_User.getText(), passWord: txt_Pass.getText(), staffName: txt_Name.getText(),
            dateBirth: txt_Date.getText(), gender: rd_Male.isSelected(), address: cb_Address.getSelectedItem().toString(),
                phoneNumber: txt_Phone.getText());
        accountList.add(e: a);
    }else{
        JOptionPane.showMessageDialog(parentComponent: this, message: "Day must be have format dd-mm-yyyy");
    }
}
```

Add Account.

- updatePhone/ updateAcc: Update information. First create a phoneList that receives data from the selected row in the table. Then use StringBuffer to check for errors when updating information. If there are no errors, then the data will be updated to the table.

```java
//UPdate Phone
private void updatePhone(){
    Phone p = phoneList.get(index: chosenRow);

    StringBuffer sb = new StringBuffer();
    if(txt_PhoneName.getText().equals(anObject: "")){
        sb.append(str: "\nPhone name cannot null!");
    }
    if(txt_Describe.getText().equals(anObject: "")){
        sb.append(str: "\nDescribe cannot null!");
    }
    if(txt_Quantity.getText().equals(anObject: "")){
        sb.append(str: "\nQuantity cannot null!");
    }
    if(isNumber() == false){
        sb.append(str: "\nQuantity must be a number!");
    }
    if(txt_Price.getText().equals(anObject: "")){
        sb.append(str: "\nPrice cannot null");
    }
    if(isPrice() == false){
        sb.append(str: "\nPrice must be written as xx,xxx,xxx!");
    }

    if(sb.length() > 0){
        JOptionPane.showMessageDialog(parentComponent: this, message: sb.toString());
    }else{
        p.setPhoneName(phoneName: txt_PhoneName.getText());
        p.setDescribe(describe: txt_Describe.getText());
        p.setQuantity(quantity: txt_Quantity.getText());
        p.setPriceString(priceString: txt_Price.getText());
    }
}
```

Update Phone.

```java
//UPdate Account
private void updateAcc(){
    Account a = accountList.get(index: chosenRow);

    StringBuffer sb = new StringBuffer();
    if(txt_User.getText().equals(anObject: "")){
        sb.append(str: "Username cannot null!");
    }
    if(txt_Pass.getText().equals(anObject: "")){
        sb.append(str: "\nPassword cannot null!");
    }
    if(txt_Name.getText().equals(anObject: "")){
        sb.append(str: "\nStaff name cannot null!");
    }
    if(isString() == false){
        sb.append(str: "\nThe name must be String type!");
    }
    if(txt_Phone.getText().equals(anObject: "")){
        sb.append(str: "\nPhone numer cannot null!");
    }
    if(IsPhoneNumber() == false){
        sb.append(str: "\nThe format phone must be 10 number start at 0!");
    }
    if(isDay() == false){
        sb.append(str: "\nDay must be have format dd-mm-yyyy");
    }

    if(sb.length() > 0){
        JOptionPane.showMessageDialog(parentComponent: this, message: sb.toString());
    }else{
        a.setUserName(userName: txt_User.getText());
        a.setPassWord(passWord: txt_Pass.getText());
        a.setStaffName(staffName: txt_Name.getText());
        a.setDateBirth(dateBirth: txt_Date.getText());
        a.setGender(gender: rd_Male.isSelected());
        a.setAddress(address: cb_Address.getSelectedItem().toString());
        a.setPhoneNumber(phoneNumber: txt_Phone.getText());
    }
}
```

Update Account.

- delete: Delete data in the table. If choosenRow > -1, declare the variable re to receive a message as YES_NO_OPTION asking if the user wants to remove it. If re = YES_OPTION then delete and save the file. In case the chosenRow does not meet the condition, the message "Please select someone!" is displayed.

```
//Delete Account
public void delete(){
    if(chosenRow > -1){
        int re = JOptionPane.showConfirmDialog(parentComponent: this, message: "Delete?", title: "",
                optionType:JOptionPane.YES_NO_OPTION, messageType: JOptionPane.WARNING_MESSAGE);
        if(re == JOptionPane.YES_OPTION){
            accountList.remove(index:chosenRow);
            clearForm();
            fillToTable();
            saveFile();
        }
    }else{
        JOptionPane.showMessageDialog(parentComponent: this, message: "Please choose someone!");
    }
}
```

Delete phone/ account.

4. Explain how to handle errors

- existID/ existName/ existUsername/ existPhone: Check if phone id, phone name, username, phone number are already exist using regular expression. Declare an ArrayList of object ids. Then use for-each to assign an id to the newly created list. Next, create a variable resultString with a boolean data type that takes a false. Use a for loop to check the ids in the list, if the id entered is the same as the existing id, return true and close the program. Otherwise, return false and return resultString. The remaining cases are similar.

```java
//Check ID
private boolean existID(String str){
    ArrayList<String> idList = new ArrayList<>();

    for(Phone p : phoneList){
        idList.add(e: p.getPhoneID());
    }
    boolean resultString = false;

    for(int i = 0; i < idList.size(); i++){
        if(str.equals(anObject: idList.get(index: i))){
            resultString = true;
            break;
        }else{
            resultString = false;
        }
    }
    return resultString;
}
```

existID.

```java
//Exist name
private boolean existName(String str){
    ArrayList<String> nameList = new ArrayList<>();

    for(Phone p : phoneList){
        nameList.add(e: p.getPhoneName());
    }
    boolean resultString = false;

    for(int i = 0; i < nameList.size(); i++){
        if(str.equals(anObject: nameList.get(index: i))){
            resultString = true;
            break;
        }else{
            resultString = false;
        }
    }
    return resultString;
}
```

existName.

```java
//Check user
private boolean existUser(String str){
    ArrayList<String> userList = new ArrayList<>();

    for(Account a : accountList){
        userList.add(e: a.getUserName());
    }
    boolean resultString = false;

    for(int i = 0; i < userList.size(); i++){
        if(str.equals(anObject: userList.get(index: i))){
            resultString = true;
            break;
        }else{
            resultString = false;
        }
    }
    return resultString;
}
```

existUser.

```java
//Check Phone
private boolean existPhone(String str){
    ArrayList<String> phoneList = new ArrayList<>();
    for(Account a : accountList){
        phoneList.add(e: a.getPhoneNumber());
    }
    boolean resultString = false;

    for(int i = 0; i < phoneList.size(); i++){
        if(str.equals(anObject: phoneList.get(index: i))){
            resultString = true;
            break;
        }else{
            resultString = false;
        }
    }
    return resultString;
}
```

existPhoneNumber.

- isMixedID: Initialize the id variable with the format id "^[GC]{2}\\d{2}$", making the variable re of the boolean data type. Create a variable str that takes input from the keyboard. Then match str with id and return the value re.

```
//Fomat ID
public boolean isMixedID() {
    String id = "^[GC]{2}\\d{2}$";
    boolean re;
    String str = txt_PhoneID.getText();
    re = str.matches(regex:id);
    return re;
}
```

- isPrice: Initialize the price variable with the format price "^[0-9]{2},[0-9]{3},[0-9]{3}$", making the variable re of the boolean data type. Create a variable str that takes input from the keyboard. Then match str with price and return the value re.

```
//Check price type
public boolean isPrice() {
    String price = "^[0-9]{2},[0-9]{3},[0-9]{3}$";
    boolean re;
    String str = txt_Price.getText();
    re = str.matches(regex:price);
    return re;
}
```

- isNumber: Initialize the number variable with the format number "[0-9]{2}$", making the variable re of the boolean data type. Create a variable str that takes input from the keyboard. Then match str with number and return the value re.

```
//Check Quantity
private boolean isNumber(){
    String number = "[0-9]{2}$";
    boolean re;
    String str = txt_Quantity.getText();
    re = str.matches(regex:number);
    return re;
}
```

- isPhoneNumber: Initialize the phone variable with the format phone "^[0][0-9]{2}[0-9]{3}[0-9]{4}$", making the variable re of the boolean data type. Create a variable P that takes input from the keyboard. Then match P with phone and return the value re.

```
//Fomat phone
private boolean IsPhoneNumber(){
    String phone ="^[0][0-9]{2}[0-9]{3}[0-9]{4}$";
    boolean re;
    String P = txt_Phone.getText();
    re= P.matches(regex:phone);
    return re;
}
```

- isDay: Initialize the dayPattern variable enter the format day "\\d{1,2}-\\d{1,2}-\\d{4}", create the isDay variable to receive the value false. Create a variable str to receive data from the keyboard, then match str with dayPattern and return isDay.

```
//Check Day
private boolean isDay(){
    String datePattern = "\\d{1,2}-\\d{1,2}-\\d{4}";
    boolean isDay = false;
    String str = txt_Date.getText();
    isDay = str.matches(regex:datePattern);
    return isDay;
}
```

- isString: Initialize the name variable with the format name "[a-zA-Z]+", making the variable re of the boolean data type. Create a variable str that takes input from the keyboard. Then match str with name and return the value re.

```
//Check name type
public boolean isString() {
    String name = "[a-zA-Z ]+";
    boolean re;
    String str = txt_Name.getText();
    re = str.matches(regex:name);
    return re;
}
```

- loadFile: There is a function to read input data streams from filePath, check for exceptions that occur during system running.
    - FileNotFoundException will be thrown by FileInputStream constructor when file with the specified pathname does not exist.
    - IOException occurs when an input or output operation fails.
    - ClassNotFoundException occurs when the Java Virtual Machine (JVM) cannot find the class specified in the code.

```
//Load File
public boolean loadFile(){
    try{
        FileInputStream fis = new FileInputStream(name: filePath);
        ObjectInputStream ois = new ObjectInputStream(in:fis);
        accList = (ArrayList<Account>) ois.readObject();
        ois.close();
        fis.close();
        return true;
    }catch(FileNotFoundException ex){
        System.err.println(x: "FIle not found!");
    }catch(IOException ex){
        System.err.println(x: "Fail!");
    }catch(ClassNotFoundException ex){
        System.err.println(x: "Class not found!");
    }
    return false;
}
```

- ClassNotFoundException occurs when the Java Virtual Machine (JVM) cannot find the class specified in the code.

- saveFile: This function performs 3 functions saving data to the specified file path, displaying data in a table and catching exceptions.

```
//Save File
public void saveFile(){
    try{
        FileOutputStream fos = new FileOutputStream(name: filePath);
        ObjectOutputStream oos = new ObjectOutputStream(out: fos);
        oos.writeObject(obj: phoneList);
        oos.flush();
        oos.close();
    }catch(FileNotFoundException e){
        System.err.println(x: "File not found!");
    }catch(IOException e){
        System.err.println(x: "Fail!");
    }
}
```

- First, in the try statement initialize a FileOutputStream (fos) object that receives data from the "filePath" path and an ObjectOutputStream (oos) object that receives a fos object. Then write the data to the ArrayList and display the data to the table. Finally, close the oos object. If the process occurs one of the exceptions, the catch statement will be executed.

V. Test.

- Test plan:
  - Creator: Dinh Dinh Khoi.
  - Test date: 21/08/2023.

No.	*Test case*	*Function*	*Test data*	*Expected output*	*Actual output*	*Result*
1	Verify that the screen shows the message "Login successful!" and "Welcome Administrator!" when admin enter the correct Username and Password of the account for admin.	Login	-Username: Admin -Password: Admin123@	When the login is successful, the message "Login successful!" and "Welcome Admin!", then will switch to the "Admin page" interface.	When the login is successful, the message "Login successful!" and "Welcome Admin!", then will switch to the "Admin page" interface.	Pass

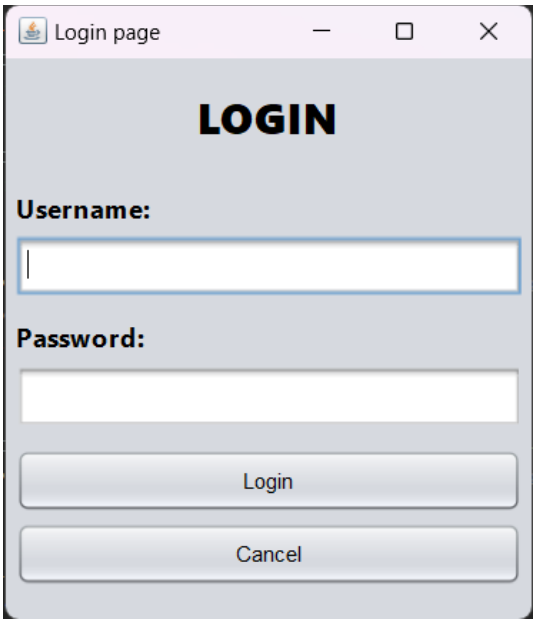
2	Verify that the screen shows the message " Login success" when staff enter the correct Username and Password of the account for staff.	Login	-Username: dinhkhoi15 -Password: khoidinh123	When the login is successful, the message "Login successful!" and switch to the "Phone Management" interface.	When the login is successful, the message "Login successful!" and switch to the "Phone Management" interface.	Pass
3	Verify that, phone information will be saved to the table if admin or staff enter all correct information.	Add	-Phone ID: GC01 -Phone name: Samsung s23 -Describe: 8gb RAM -Quantity: 50 -Price: 20,000,000	The phone information will be saved to the table.	The phone information will be saved to the table.	Pass
4	Verify that the screen shows the error "Phone ID must be written as GCxx (x is a number)!" if the administrator or employee enters the wrong phone ID format.	Add	-Phone ID: 001 -Phone name: Iphone 14 -Describe: 128gb -Quantity: 50 -Price: 20,000,000	The screen will display the error "Phone ID must be written as GCxx (x is a number)!".	The screen will display the error "Phone ID must be written as GCxx (x is a number)!".	Pass
5	Verify that the screen shows the error "Phone name already exists!" if the administrator or employee enters	Add	-Phone ID: GC04 -Phone name: Samsung s23 -Describe: 64gb -Quantity: 50 -Price: 15,000,000	The screen will display the error "The phone name already exist!".	The screen will display the error "The phone name already exist!".	Pass

	the phone name that appeared before.					
6	Verify that the screen shows the error "Price must be write as xx,xxx,xxx!" if admin or staff enter wrong price format.	Add	-Phone ID: GC05 -Phone name: Samsung X -Describe: 32gb -Price: 2000sdad	The screen will display the error "Price must be write as xx,xxx,xxx!".	The screen will display the error "Price must be write as xx,xxx,xxx!".	Pass
7	Verify that the phone's information is deleted from the board when an administrator or employee presses the "Delete" button.	Delete	Click on the phone information line containing the ID GC03	The data of the phone ID GC03 will be deleted from the table.	The data of the phone ID GC03 will be deleted from the table.	Pass
8	Verify that the data of the selected line will be updated after the administrator or employee presses the "Update" button.	Update	-Phone ID: GC02 -Phone name: Samsung SX -Describe: 8gb RAM -Quantity: 50 -Price: 20,000,000	The phone name of ID GC02 will change to “Samsung SX”.	The phone name of ID GC02 will change to “Samsung SX”.	Pass
9	Verify that the screen shows the	Update	-Phone ID: GC02	The screen will display the error	The screen will display the error	Pass

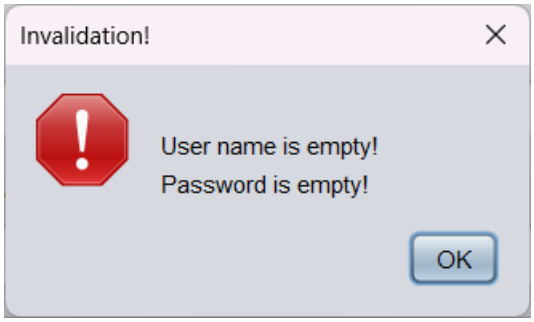
	error "Quantity must be a number!" if admin or staff enter wrong quantity type.		-Phone name: Samsung SX -Describe: 8gb RAM -Quantity: asd -Price: 20,000,000	" Quantity must be a number!".	" Quantity must be a number!".	
10	Verify that the screen shows the error " Price must be written as xx,xxx,xxx!" if admin or staff enter wrong price format.	Update	-Phone name: Samsung SX -Describe: 8gb RAM -Quantity: asd -Price: asdada	The screen will display the error " Price must be written as xx,xxx,xxx!".	The screen will display the error " Price must be written as xx,xxx,xxx!".	Pass

VI. Result.

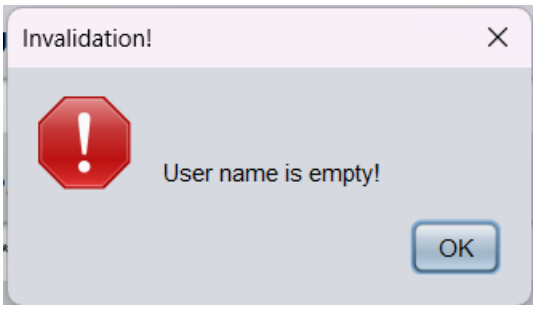
- Login interface:
  - When the Administrator or Staff runs the program, the first interface that appears is the login interface that requires them to enter "Username" and "Password" to log in to the application.



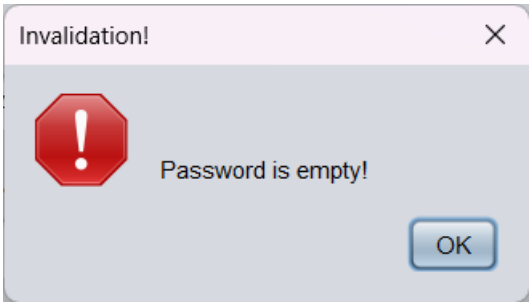
- If the Administrator or Staff do not enter “Username” and “Password” but presses the Login button, the program will show error message:



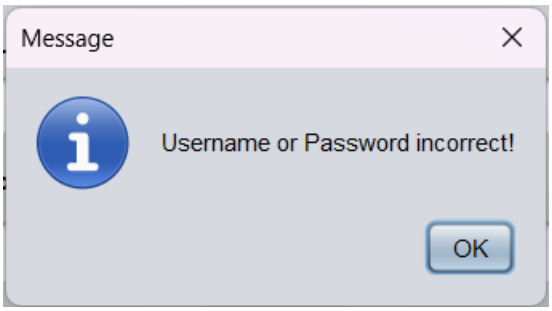
- If the Administrator or Staff do not enter “Username”, the program will show error message:



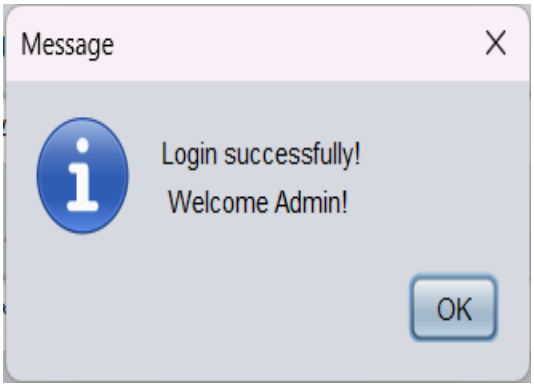
- If the Administrator or Staff do not enter “Password”, the program will show error message:

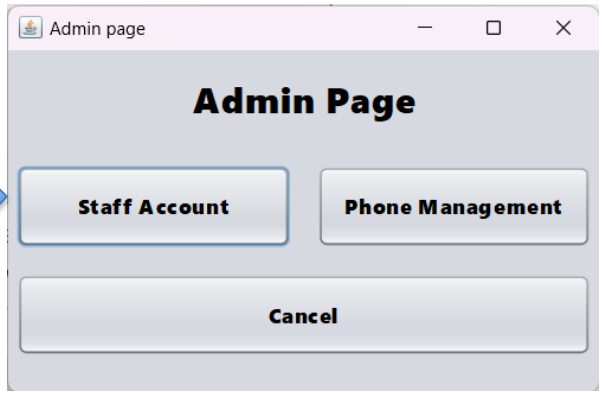


- If the Administrator enter wrong “Username” and “Password” or both, the program will show error message:

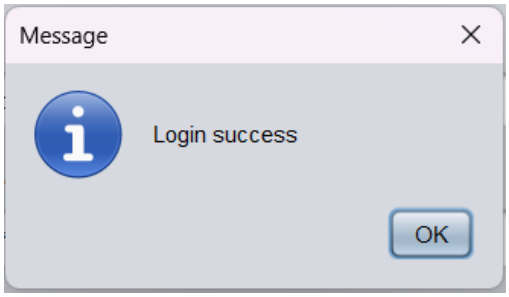


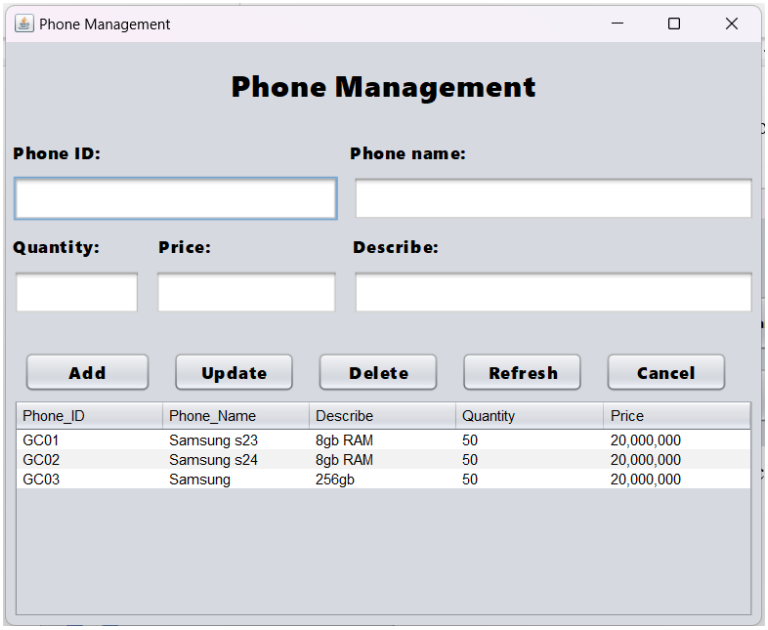
- In case the administrator enters the correct administrator account, the program will notify "Login successful!" and “Welcome Admin!”, then switch to the Admin page interface:



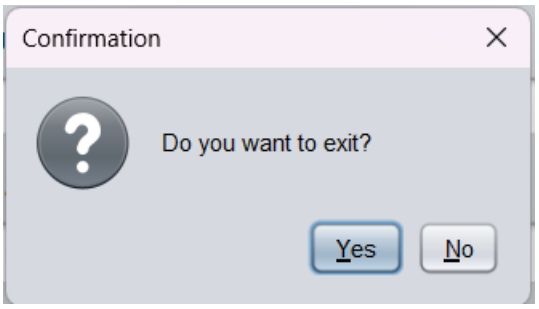


- In case the staff enters the correct staff account, the program will notify "Login success", then switch to the Phone Management interface:

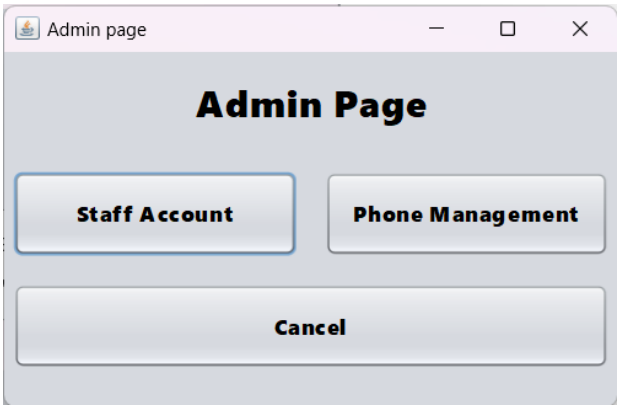




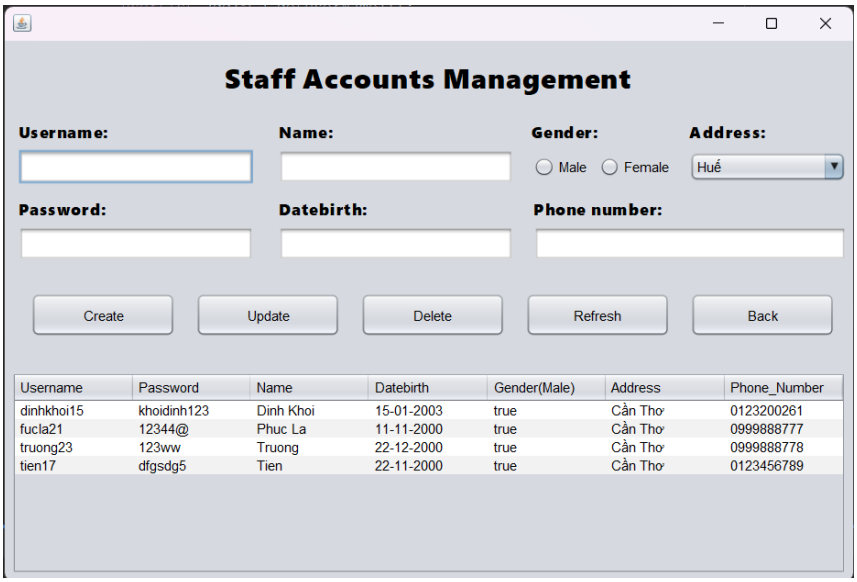
- If the Administrator or Staff want to exit the program, press the “Cancel” button:



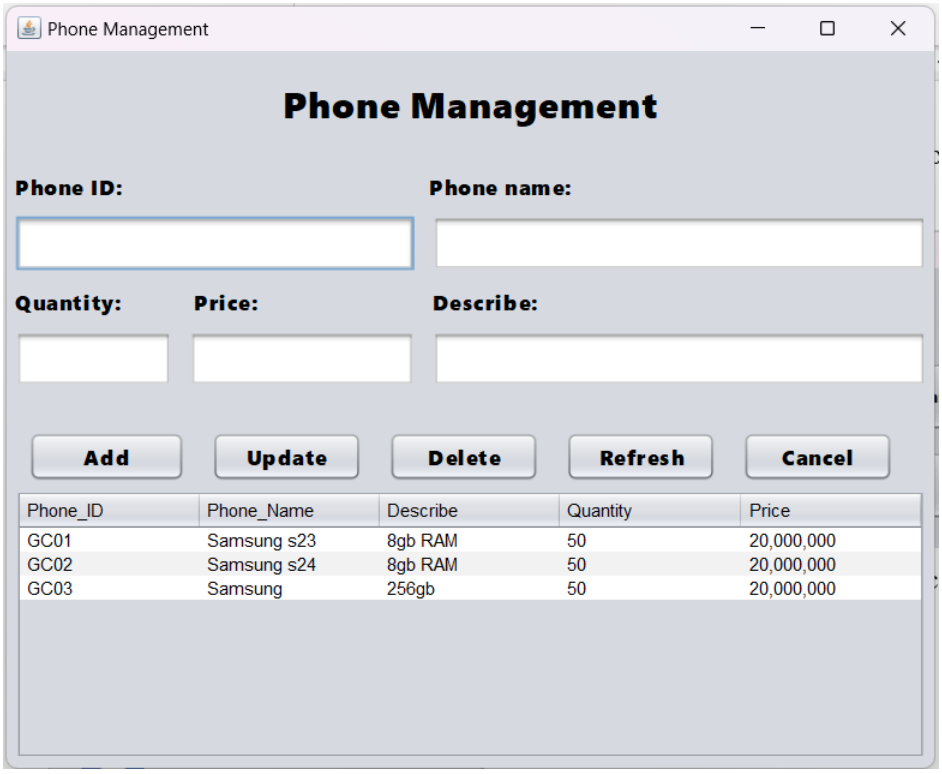
- Admin interface:
  - At the "Admin page" interface, the admin can choose the management functions "Staff Account" or product management "Phone Management".



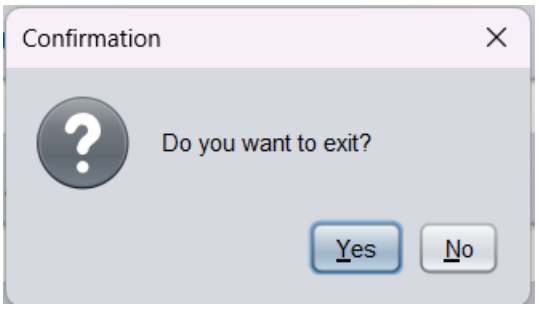
- When the Administrator choose Staff Account button, the program will switch to Staff Account interface:



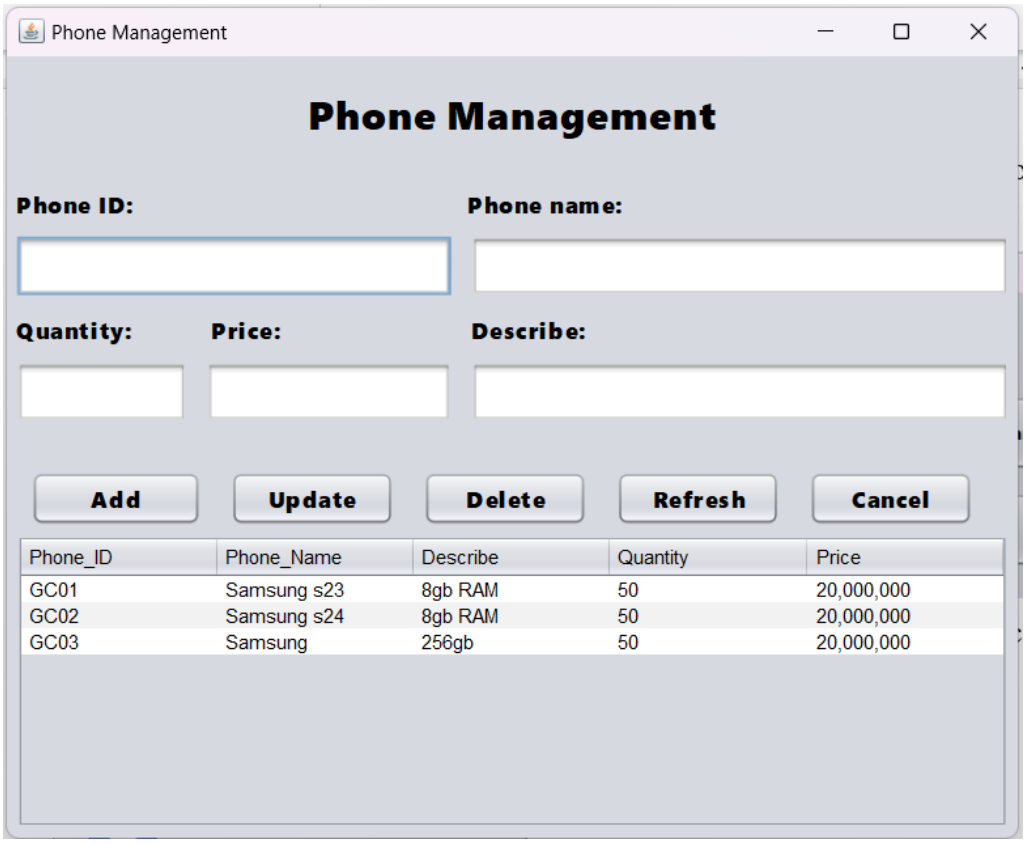
- When the Administrator choose Phone Management button, the program will switch to Phone Management interface:



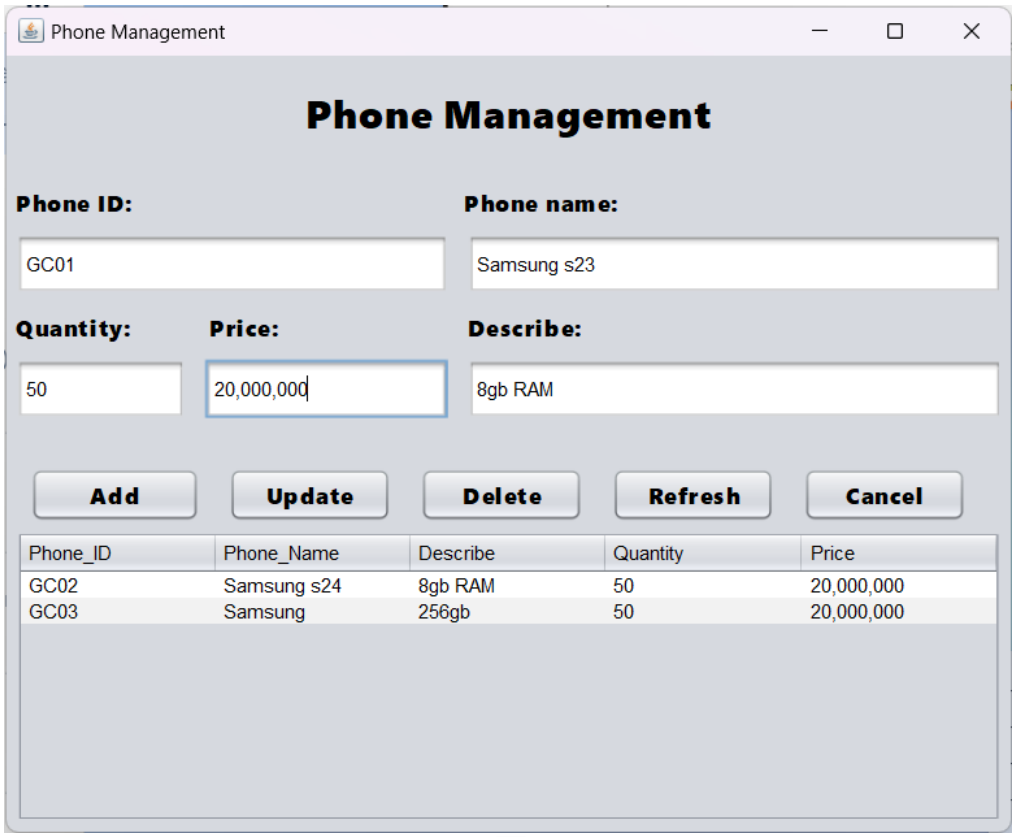
- When the Administrator or Staff want to exit to login interface, press “Cancel” button:



- Phone Management interface:
  - This is the product management interface called "Phone Management". Here, Administrator and Employees can perform the functions of Add new products, Update product information and Delete products.

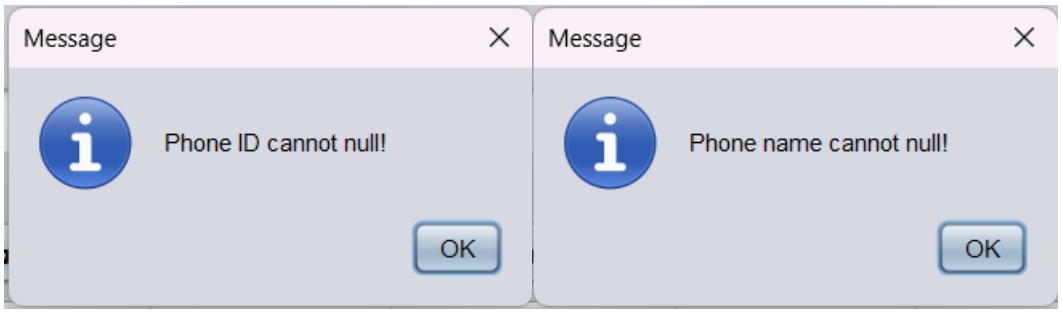


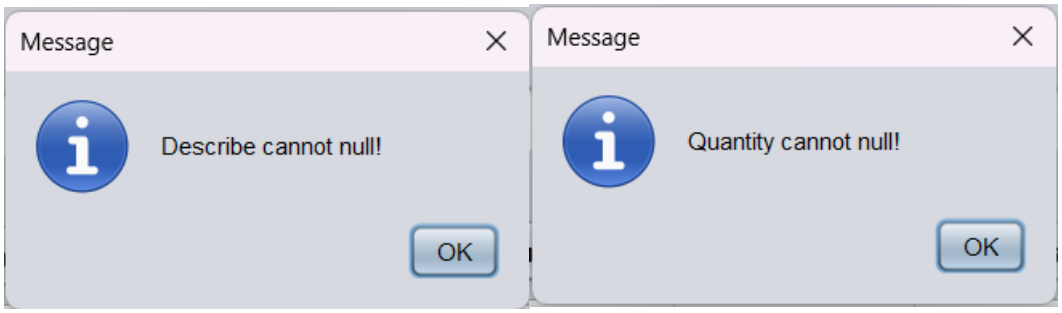
- ***Add button:***
- When the Administrator or Staff enter the correct product information, press the “Add” button, the product data will be saved in the information table below:

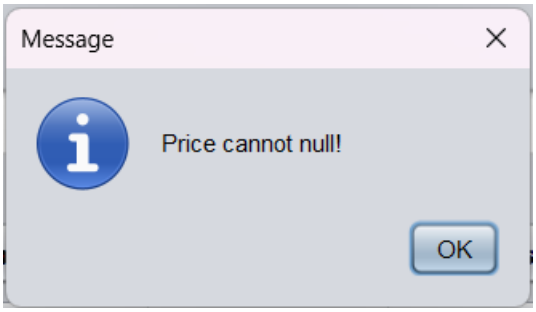

Phone Management
Phone Management
Phone ID:
Phone name:
GC01
Samsung s23
Quantity:
Price:
Describe:
50
20,000,000
8gb RAM
Add
Update
Delete
Refresh
Cancel
Phone_ID
Phone_Name
Describe
Quantity
Price
GC02
Samsung s24
8gb RAM
50
20,000,000
GC03
Samsung
256gb
50
20,000,000


Phone Management
Phone Management
Phone ID:
Phone name:
GC01
Samsung s23
Quantity:
Price:
Describe:
50
20,000,000
8gb RAM
Add
Update
Delete
Refresh
Cancel
Phone_ID
Phone_Name
Describe
Quantity
Price
GC02
Samsung s24
8gb RAM
50
20,000,000
GC03
Samsung
256gb
50
20,000,000
GC01
Samsung s23
8gb RAM
50
20,000,000

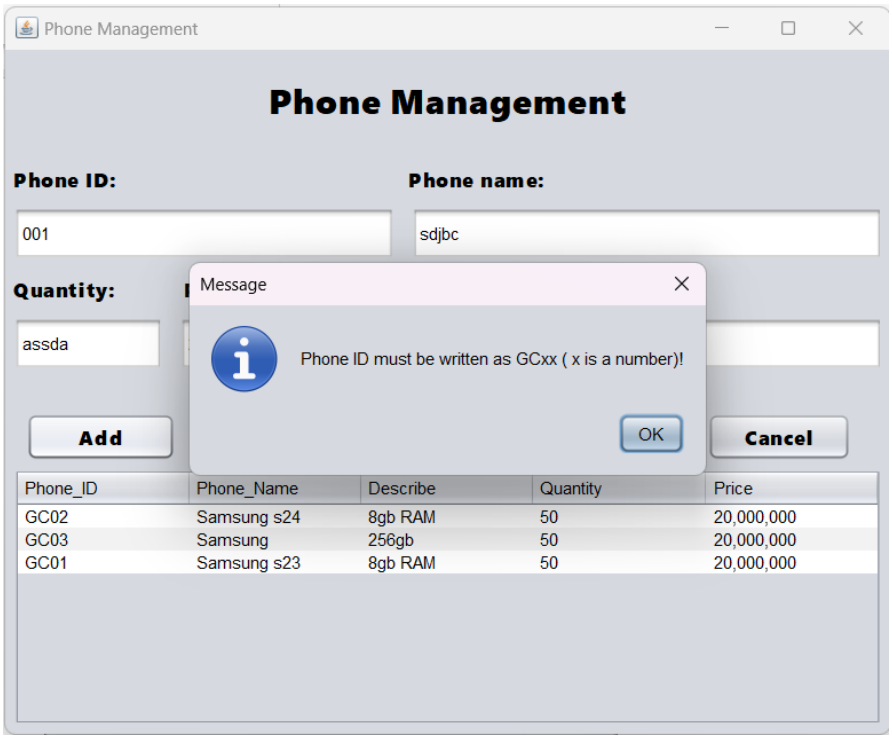
- If the Administrator or Staff leaves 1 or more blank cells, the system will display an error message as follows:



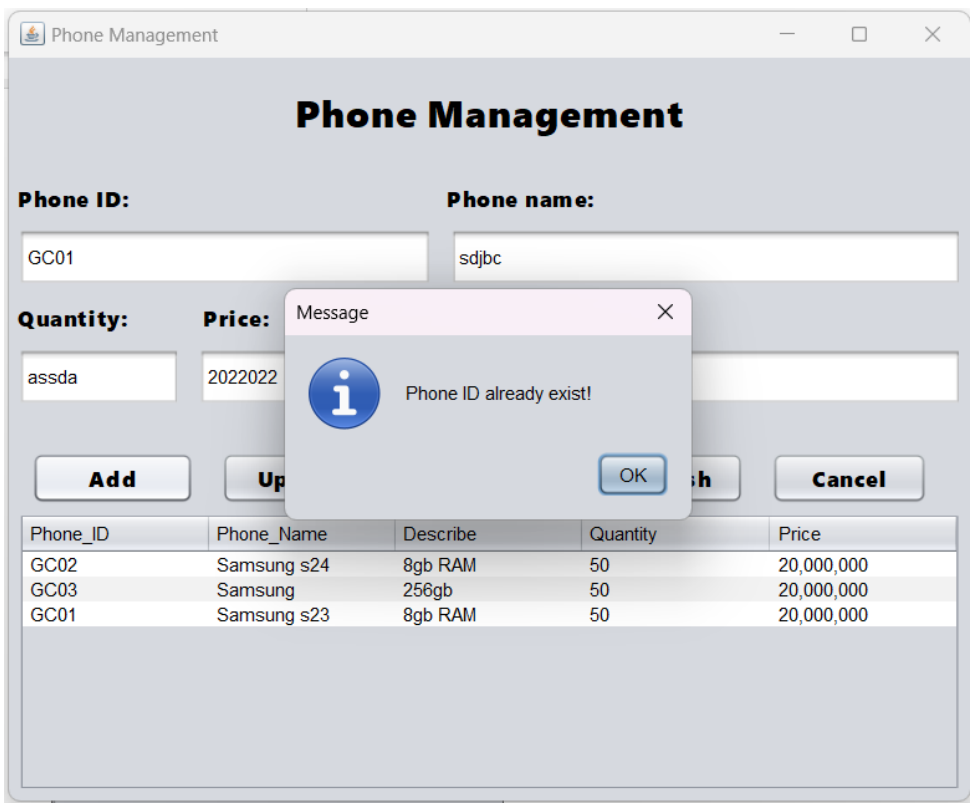




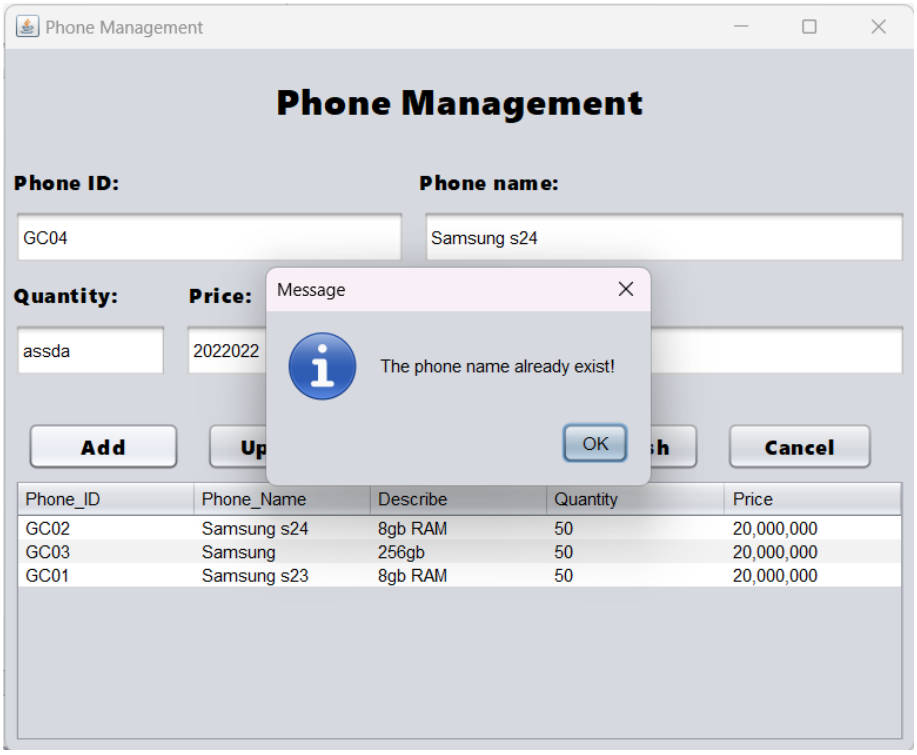
- If the Administrator or Staff enter wrong ID format, the program will show error message:



- If the Administrator or Staff enter an existed ID, the program will show error message:

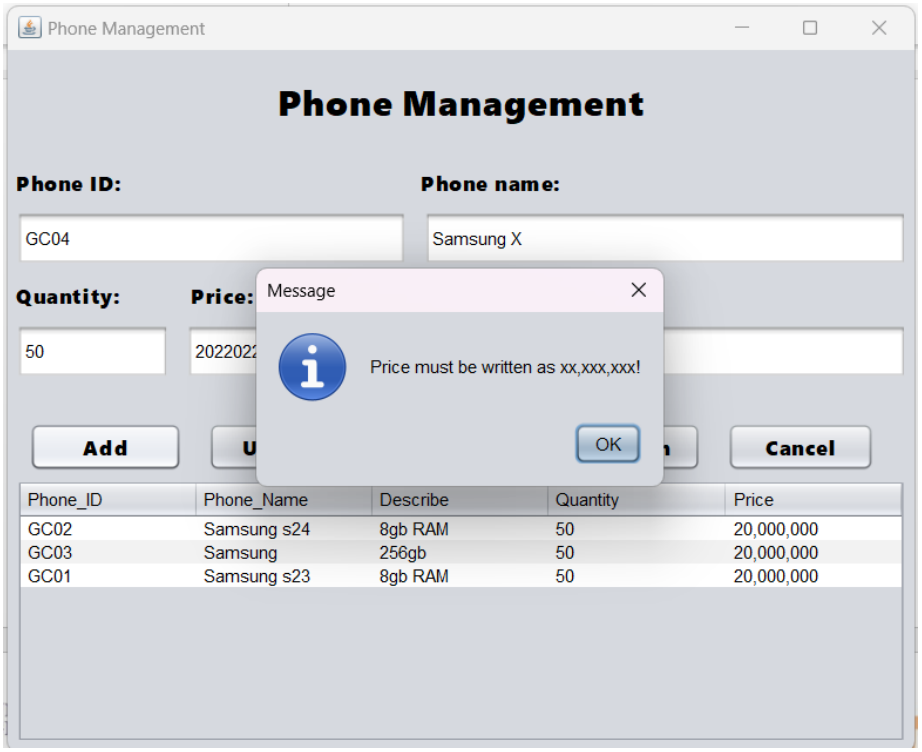


- If the Administrator or Staff enter an existed phone name, the program will show error message:

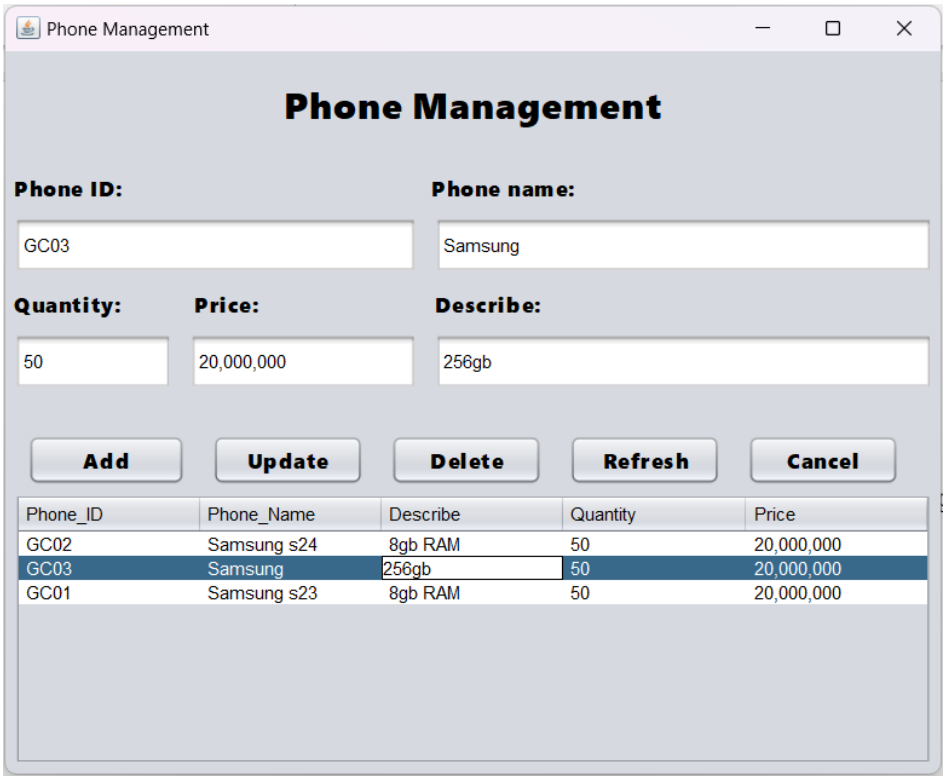


- If the Administrator or Staff enters the quantity as letters instead of numbers, the program will show error message:

- If the Administrator or Staff enters wrong price format, the program will show error message:

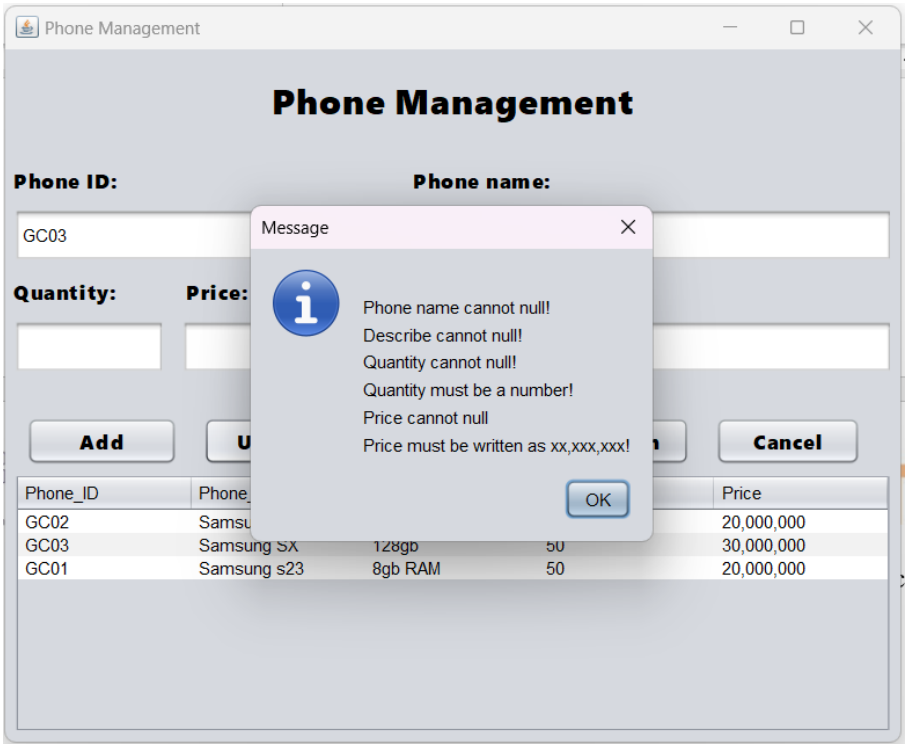


- ***Update button:***
- When the Administrator or Staff enter the correct product information need to update, press "Update" button, the product data will save to the table below:

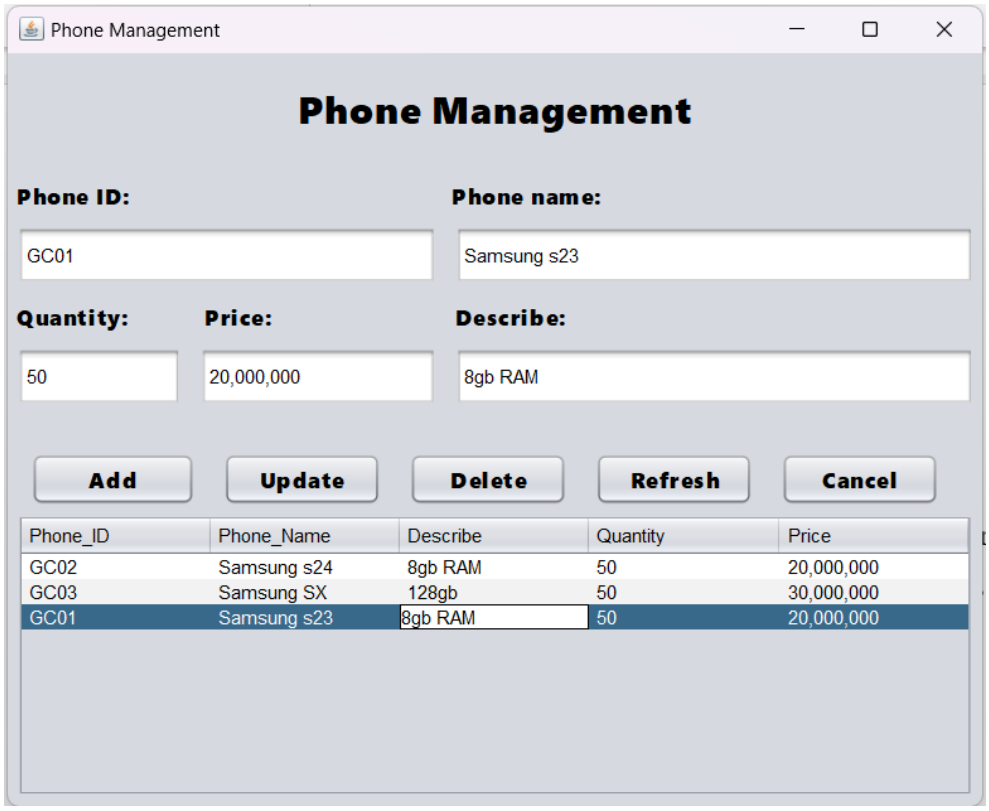

Phone Management
Phone Management
Phone ID:
GC03
Phone name:
Samsung
Quantity:
50
Price:
20,000,000
Describe:
256gb
Add
Update
Delete
Refresh
Cancel
Phone_ID | Phone_Name | Describe | Quantity | Price
GC02 | Samsung s24 | 8gb RAM | 50 | 20,000,000
GC03 | Samsung | 256gb | 50 | 20,000,000
GC01 | Samsung s23 | 8gb RAM | 50 | 20,000,000

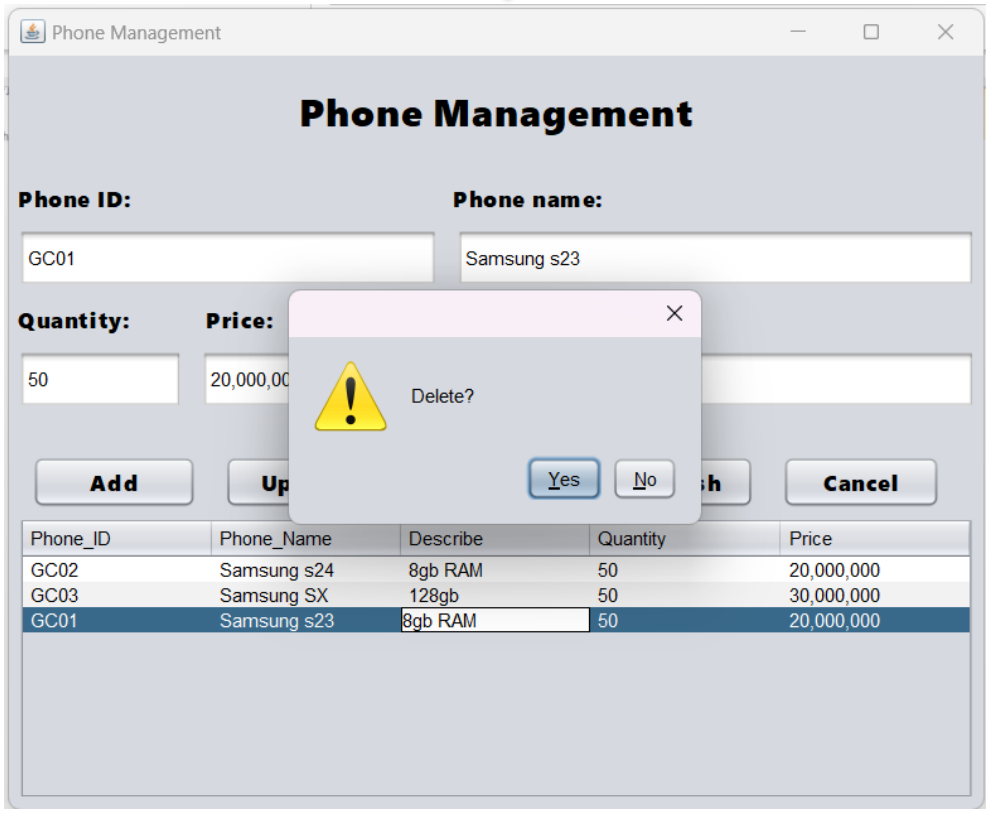

Phone Management
Phone Management
Phone ID:
GC03
Phone name:
Samsung SX
Quantity:
50
Price:
30,000,000
Describe:
128gb
Add
Update
Delete
Refresh
Cancel
Phone_ID | Phone_Name | Describe | Quantity | Price
GC02 | Samsung s24 | 8gb RAM | 50 | 20,000,000
GC03 | Samsung SX | 128gb | 50 | 30,000,000
GC01 | Samsung s23 | 8gb RAM | 50 | 20,000,000

- If Administrator or Staff leave blank phone name, describe, quantity, product price, wrong price format and quantity format, the program will show error message:



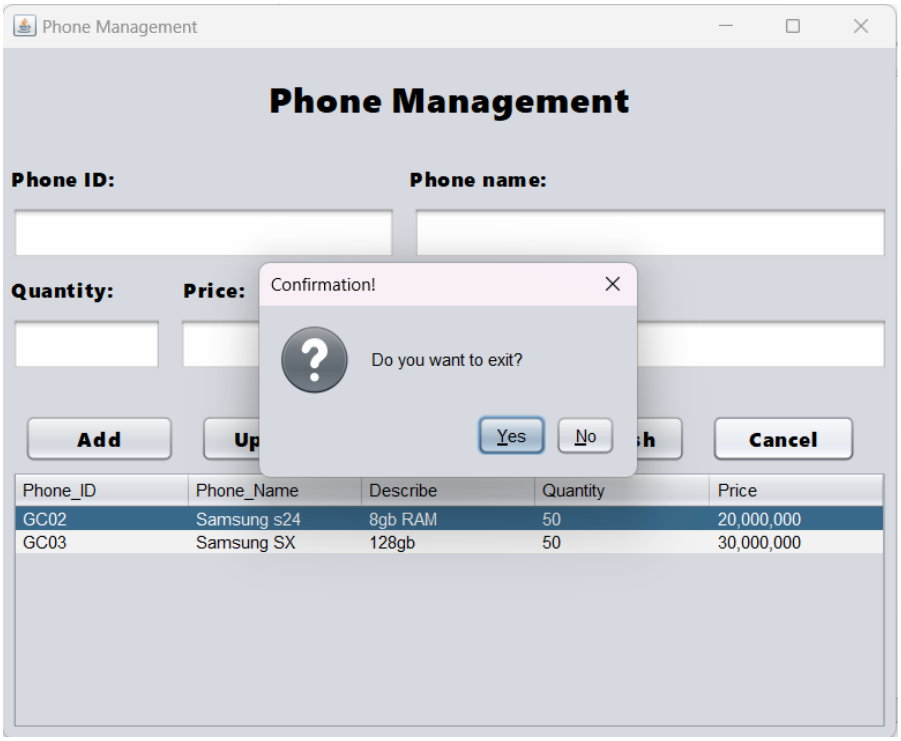
- ***Delete button:***
- To delete a product, Administrator or Staff need to select the product line they want to delete, then press the “Delete” button. The program will ask the Admin or Staff if they want to delete, press "Ok" to delete:

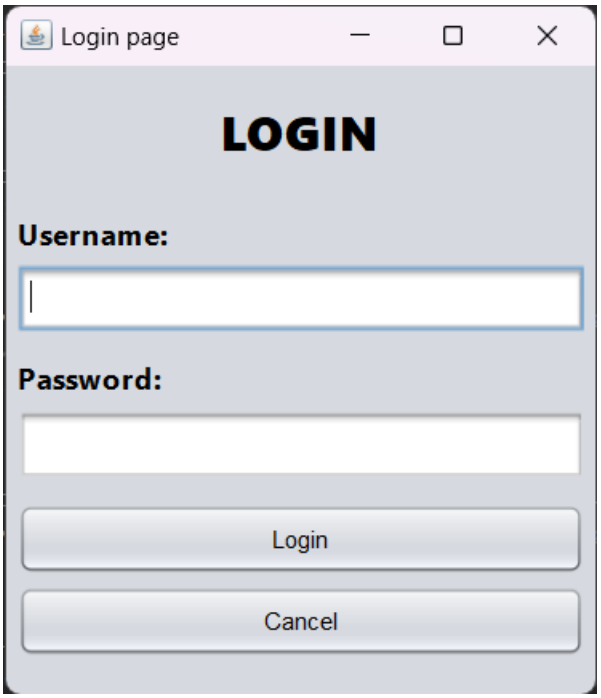

Phone Management
Phone Management
Phone ID:
Phone name:
GC01
Samsung s23
Quantity:
Price:
Describe:
50
20,000,000
8gb RAM
Add
Update
Delete
Refresh
Cancel
Phone_ID
Phone_Name
Describe
Quantity
Price
GC02
Samsung s24
8gb RAM
50
20,000,000
GC03
Samsung SX
128gb
50
30,000,000
GC01
Samsung s23
8gb RAM
50
20,000,000


Phone Management
Phone Management
Phone ID:
Phone name:
GC01
Samsung s23
Quantity:
Price:
50
Delete?
Yes
No
Add
Cancel
Phone_ID
Phone_Name
Describe
Quantity
Price
GC02
Samsung s24
8gb RAM
50
20,000,000
GC03
Samsung SX
128gb
50
30,000,000
GC01
Samsung s23
8gb RAM
50
20,000,000

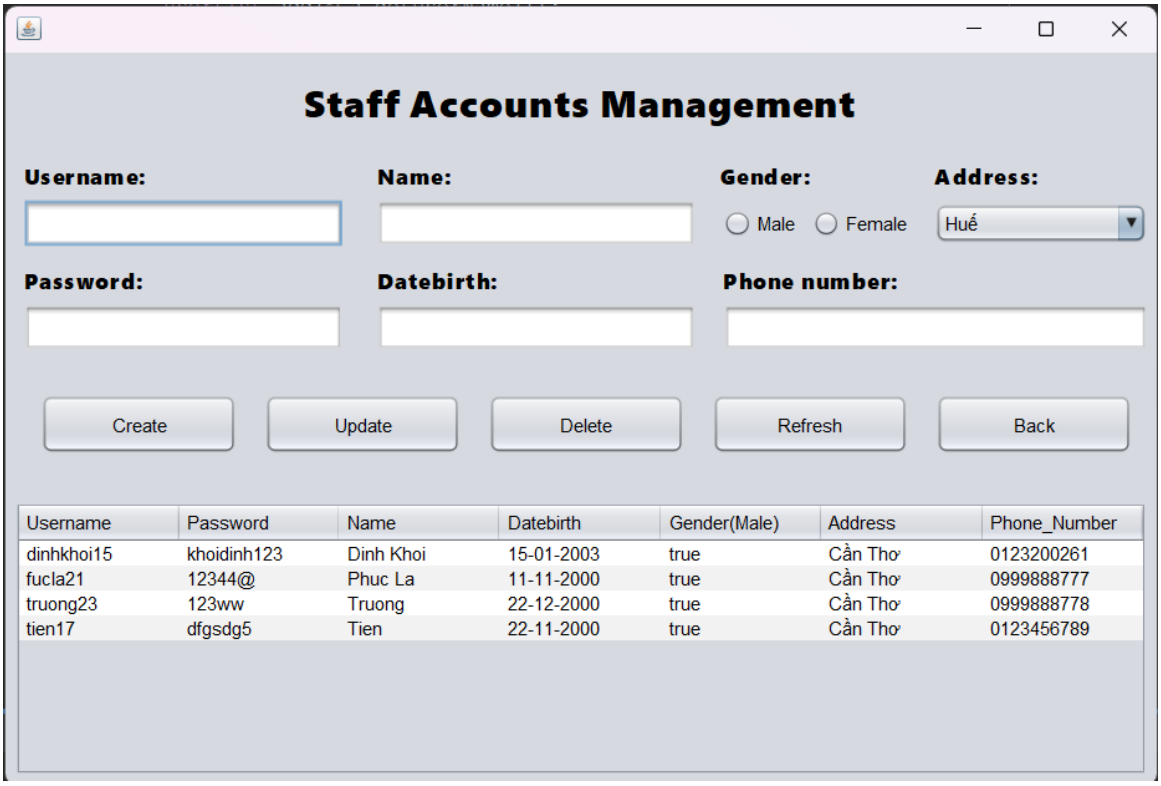
- ***Refresh button:***
- "Refresh" button is used to clear all input data in the text fields:

- ***Cancel button:***
- If the Administrator or Staff want to exit to login interface, press "Cancel" button:

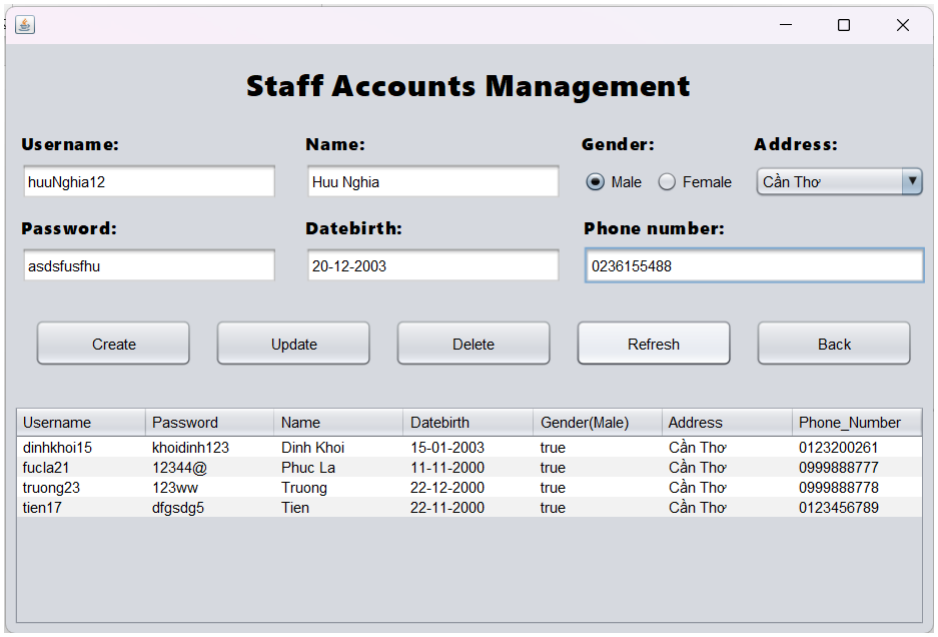


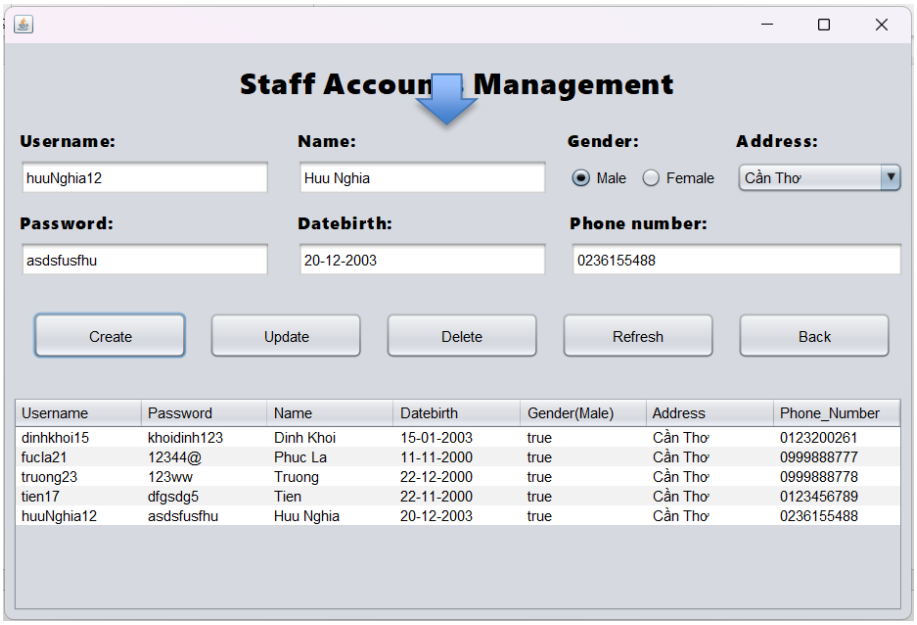


- Staff Account interface:
  - This is the staff account management interface called "Staff Account". Here, Administrator can perform the functions of Create new account, Update account information and Delete accounts.

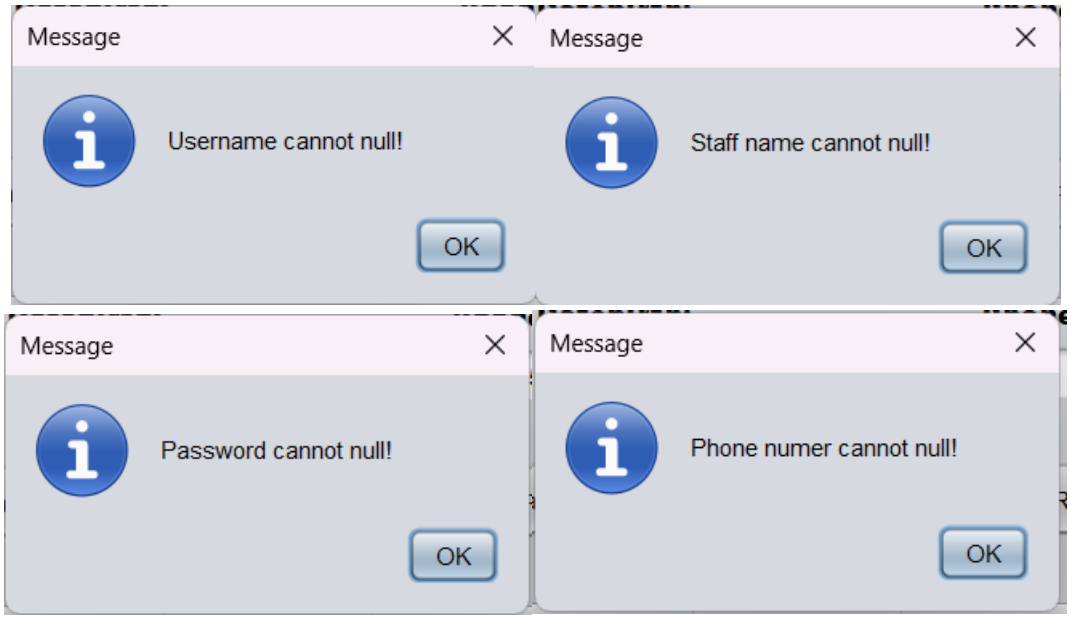


- ***Create button:***
- When the Administrator enters the correct account information, press the "Create" button, the account data will be saved in the information table below:

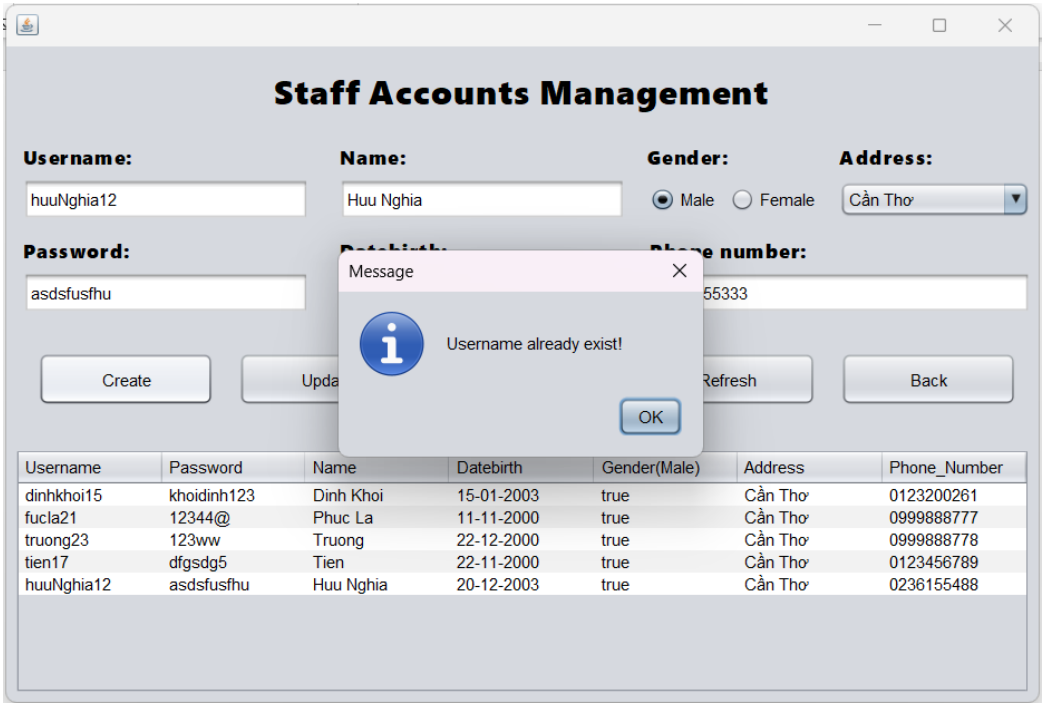




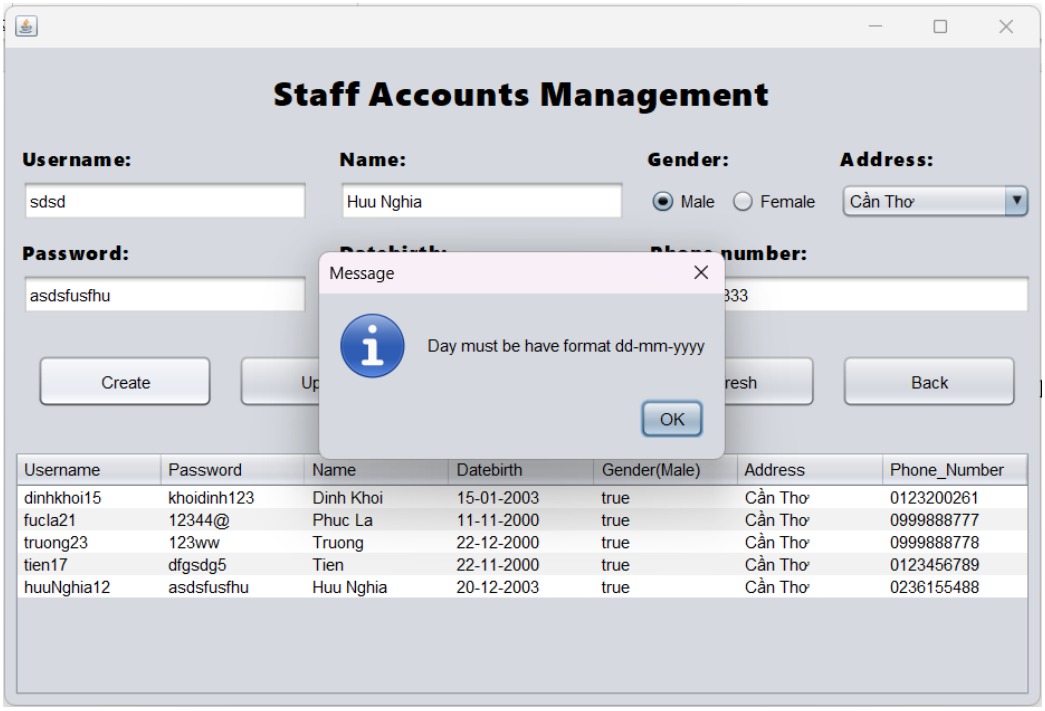
- If the Administrator leaves 1 or more blank cells, the system will display an error message as follows:



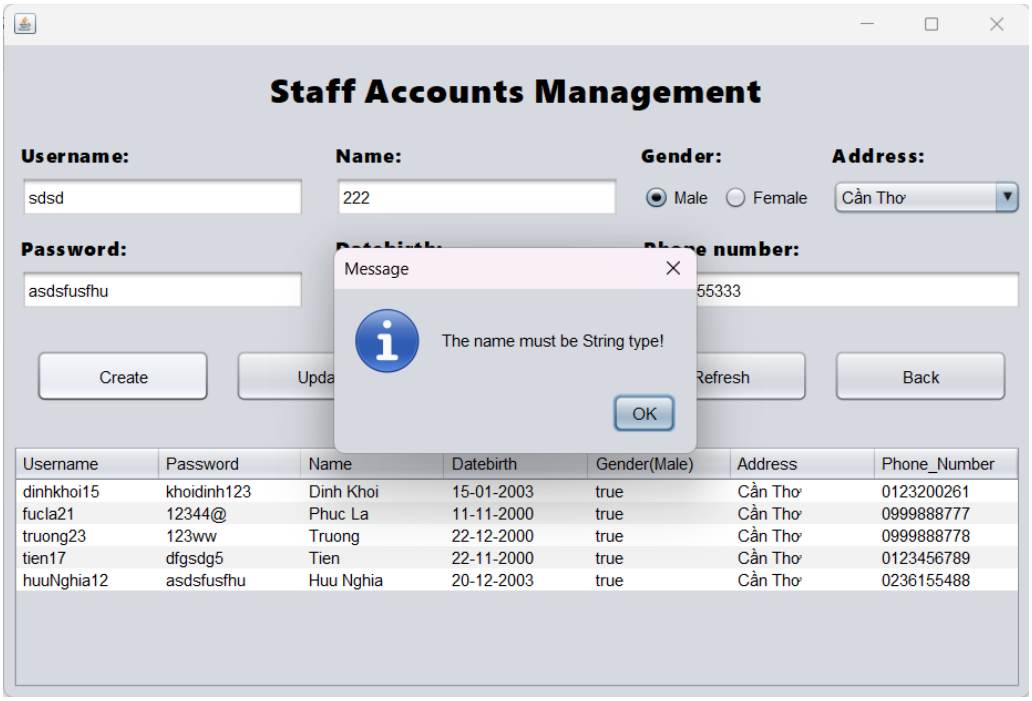
- If the Administrator enters an existed Username, the program will show error message:



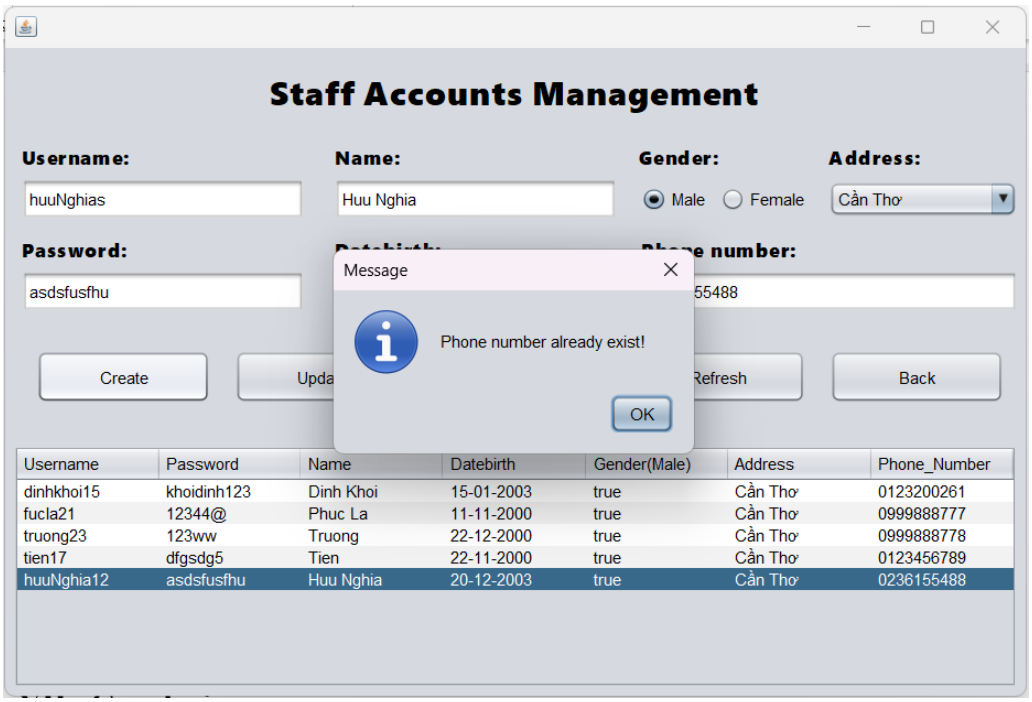
- If the Administrator leaves the Date birth blank or enters wring format date, the program will show error message:



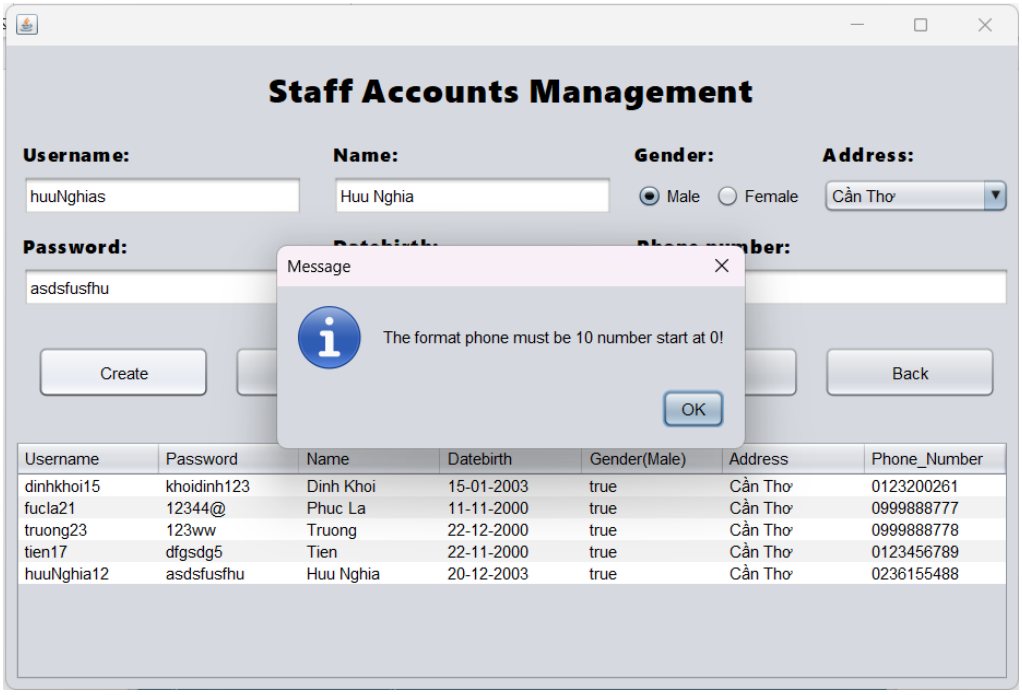
- If the Administrator enters the staff name as numbers instead of letters, the program will show error message:



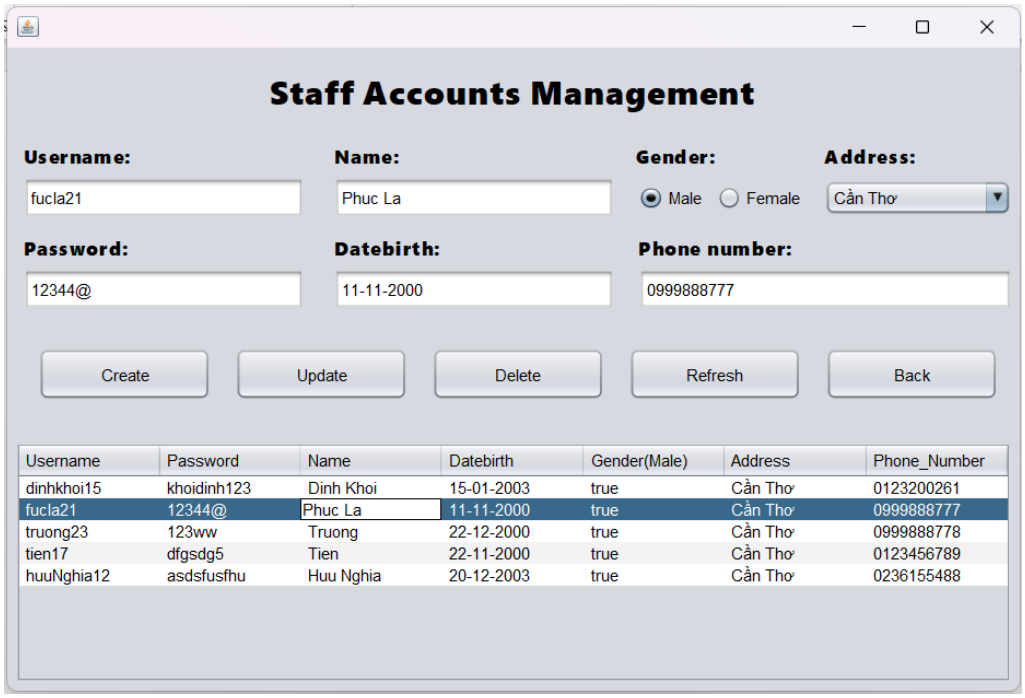
- If the Administrator enters an existed phone number, the program will show error message:

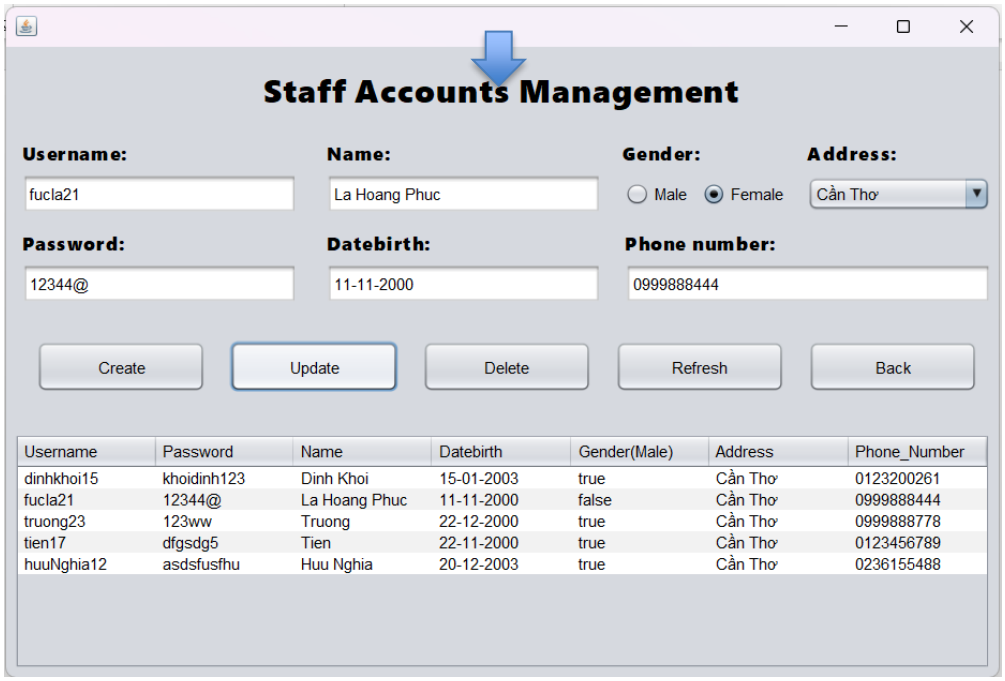


- If the Administrator enters wrong phone number format, the program will show error message:

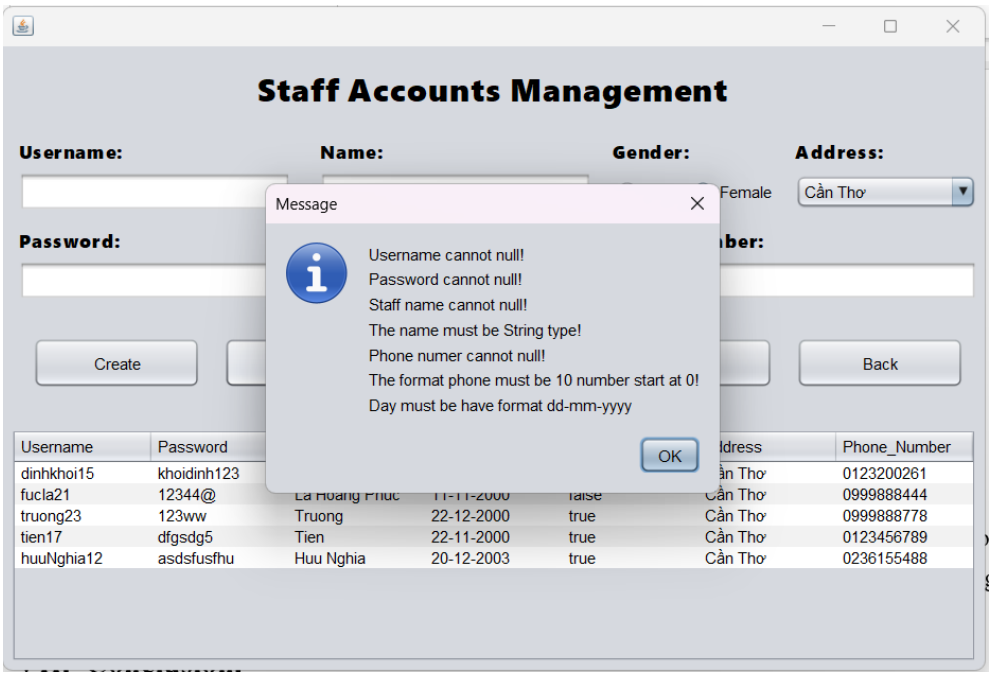


- ***Update button:***
- When the Administrator enters the correct staff account information need to update, press "Update" button, the account data will save to the table below:

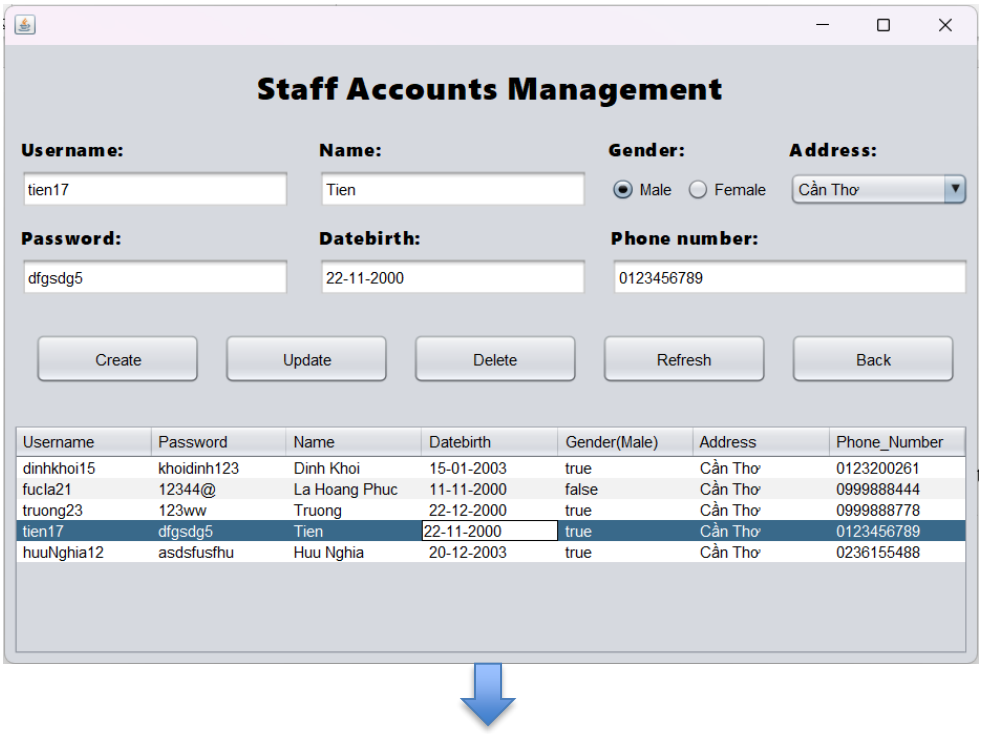


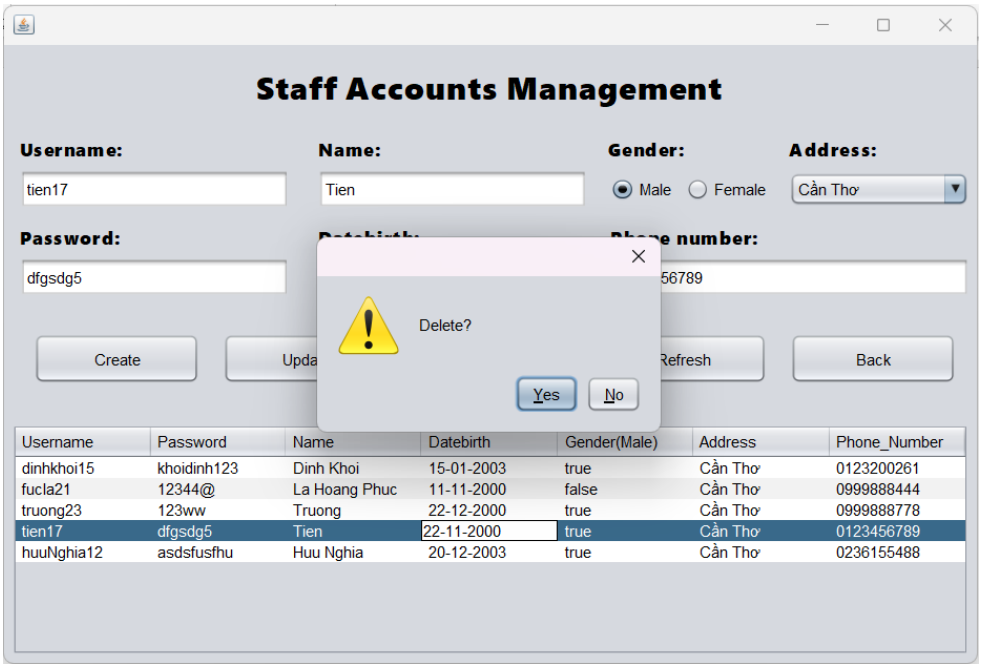


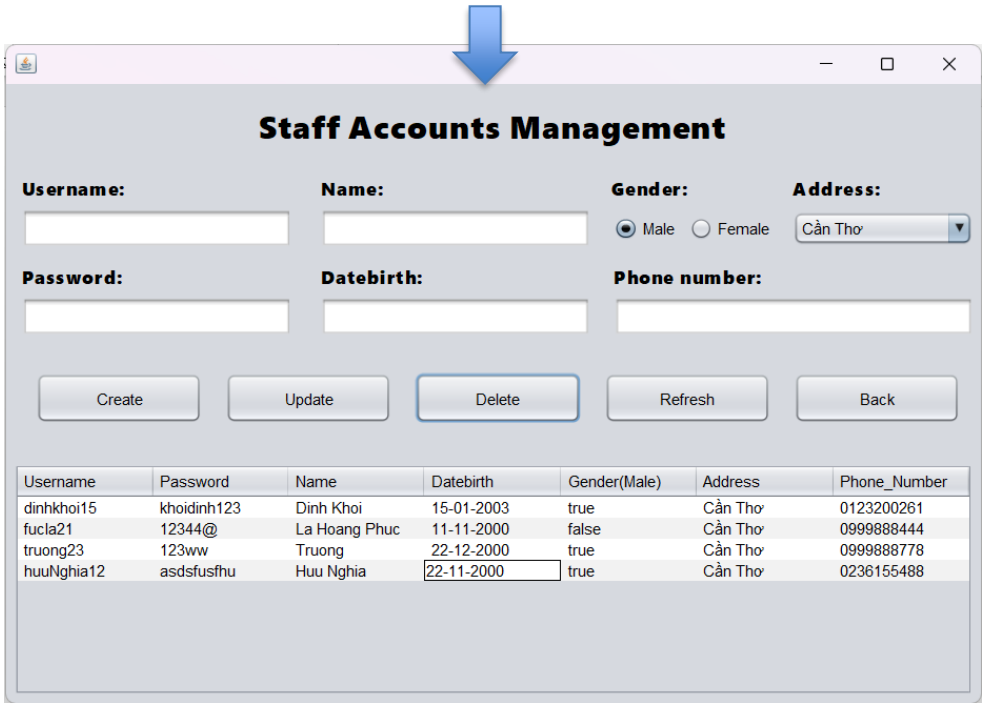
- If Administrator leaves blank username, staff name, password, date birth, phone number, wrong date format, staff name format, and phone number format, the program will show error message:



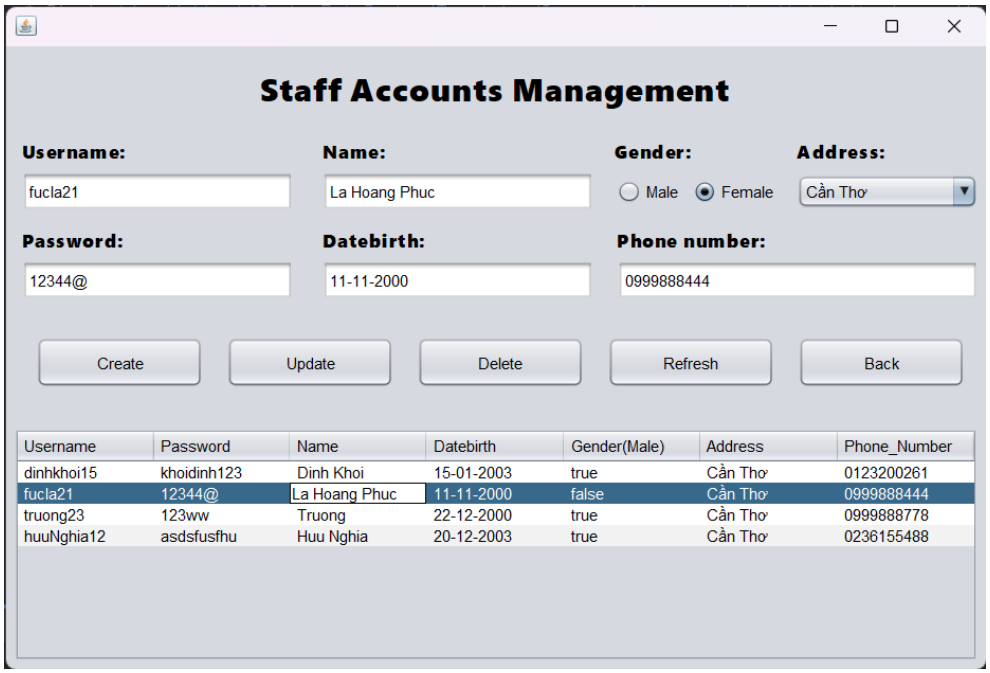
- ***Delete button:***
- To delete a account, Administrator needs to select the account they want to delete, then press the "Delete" button. The program will ask the Admin if they want to delete, press "Ok" to delete:

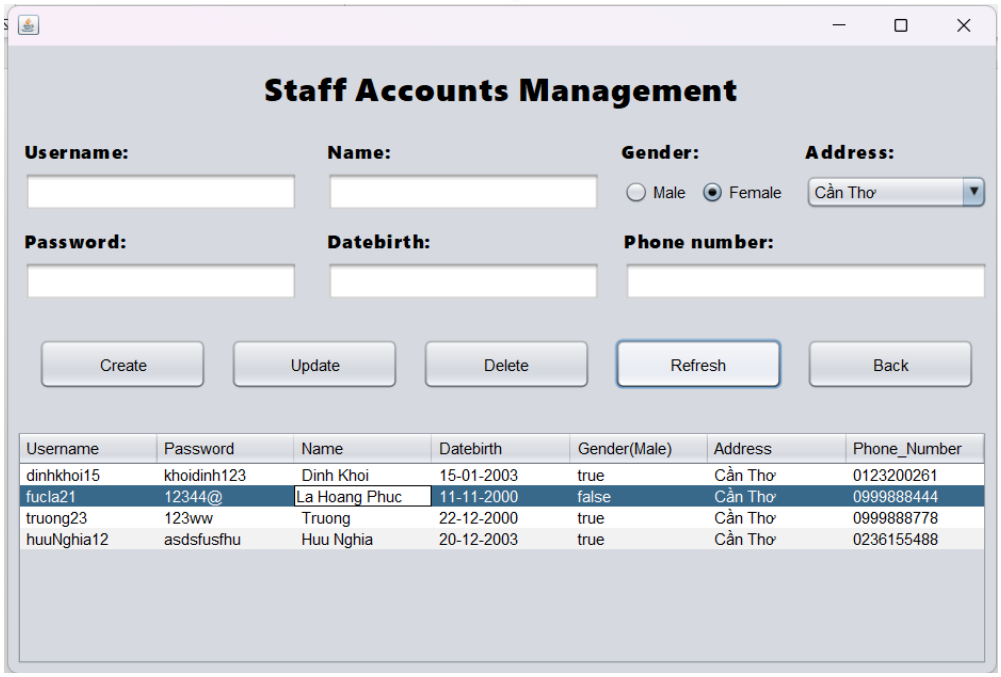




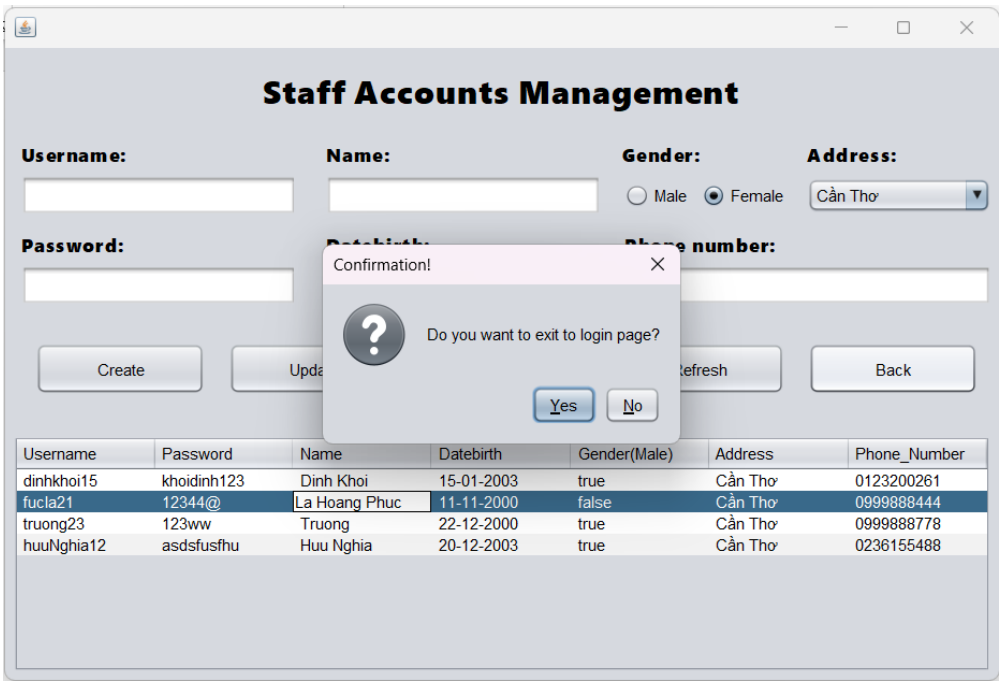


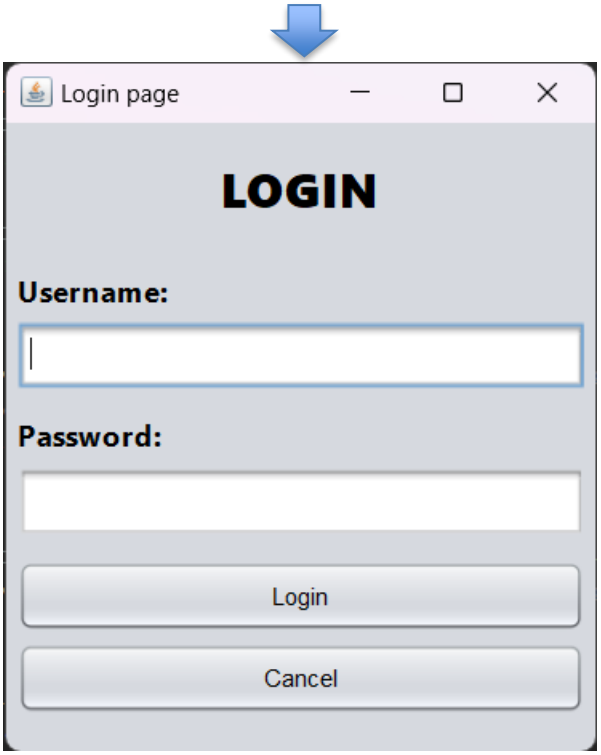
- ***Refresh button:***
- "Refresh" button is used to clear all input data in the text fields:





- ***Back button:***
- If the Administrator want to exit to login interface, press “Back” button:





VII. Conclusion.

- Strength: Accomplish most of the main functions of the application: Sign in, Add, Update, Delete, etc. Understand the basics of creating an application using the Java programming language. Several

regular expressions can be used to check for formatting errors of some data fields. Understand the basics of the Java language.

- Weakness:
    - When logging in, the password from the file cannot be checked.
    - There is no data connection between the interfaces.
    - The design is sketchy.

VIII. References.

FLM, n.d. [Online]
Available at: https://flm.greenwich.edu.vn/gui/role/student/SyllabusDetails?sylID=2581