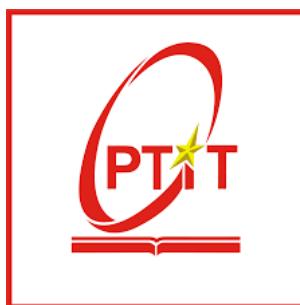


HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA VIỄN THÔNG 2



BÁO CÁO MÔN HỌC
LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG
ĐỀ TÀI: QUẢN LÝ KHÁCH HÀNG

Giảng viên: Th.S Nguyễn Văn Hiền

Sinh viên thực hiện:

Huỳnh Quang Triệu – N19DCVT069

Võ Hoài Thanh - N19DCVT060

Lê Nguyễn Đức Duy - N19DCVT006

Chu Đình Huấn - N19DCVT013

Nguyễn Duy Phúc - N19DCVT044

TP. HỒ CHÍ MINH – 2023

MỤC LỤC

LỜI CẢM ƠN	4
PHẦN I: ĐỀ TÀI	5
PHẦN II: NỘI DUNG THỰC HIỆN	5
1. Sơ đồ lớp	5
2. Xây dựng cơ sở dữ liệu	5
a. Tạo một database	6
b. Tạo một table trong database	6
c. Liên kết cơ sở dữ liệu với project	8
3. Xây dựng code	8
a. Database	8
b. Mô hình MVC (Model, View, Controller)	9
c. Test	35
PHẦN III: KẾT QUẢ CHẠY CHƯƠNG TRÌNH	36
1. Login Admin	36
2. Ngày và giờ trên giao diện	38
3. Chức năng lưu	38
4. Chức năng chỉnh sửa	40
5. Chức năng tìm kiếm	41
a. Tìm kiếm khách hàng bằng họ tên	41
b. Tìm kiếm khách hàng bằng số điện thoại	43
6. Chức năng hiển thị lại danh sách khách hàng	44
7. Chức năng xoá	44
8. Hiển thị giao diện danh sách admin	46
9. Chức năng trên thanh menu	47
a. Menu About	47
b. Menu file	48
c. Back	52

PHẦN IV: SOURCE CODE	53
1. ConnectDatabase.java	53
2. Customer.java	53
3. CustomerModify.java	56
4. Login_Admin.java	62
5. Ds_Admin.java	67
6. CustomerManagement.java	70
7. Test.java	90
KẾT LUẬN CHUNG	90

MỤC LỤC HÌNH ẢNH

Hình 1: Sơ đồ lớp quản lý khách hàng	5
Hình 2: Logo hệ quản trị cơ sở dữ liệu MySQL	6
Hình 3: Database customermanager	6
Hình 4: Mô tả cơ sở dữ liệu với thực thể Customer	7
Hình 5: Bảng customer với các cột được chỉ định	8
Hình 6: Thư viện MySQL JDBC Driver JAR	8
Hình 7: Thư viện tham chiếu	8

LỜI CẢM ƠN

Lời đầu tiên, nhóm em xin gửi lời cảm ơn chân thành nhất đến Thầy Nguyễn Văn Hiền. Trong quá trình học tập và tìm hiểu bộ môn “*Lập trình hướng đối tượng*”, nhóm em đã nhận được sự quan tâm giúp đỡ, hướng dẫn rất tận tình, tâm huyết của Thầy. Thầy đã giúp nhóm em tích lũy thêm nhiều kiến thức để có cái nhìn sâu sắc và hoàn thiện hơn trong quá trình nghiên cứu đề tài. Từ những kiến thức mà Thầy truyền tải, nhóm em đã học hỏi và trau dồi thêm nhiều kỹ năng như tìm hiểu, phân tích vấn đề, tìm kiếm tài liệu, hiểu cấu trúc và hoạt động của đề tài hơn, từ đó đưa ra hướng giải quyết vấn đề tốt hơn. Đây chắc chắn là những kiến thức bổ ích sẽ giúp nhóm em rất nhiều trong quá trình học tập sau này.

Do chưa có nhiều kinh nghiệm làm đề tài cũng như những hạn chế về kiến thức, trong bài báo cáo chắc chắn sẽ không tránh khỏi những thiếu sót. Nhóm em rất mong nhận được sự nhận xét, ý kiến đóng góp, phê bình từ phía Thầy để bài báo cáo được hoàn thiện hơn.

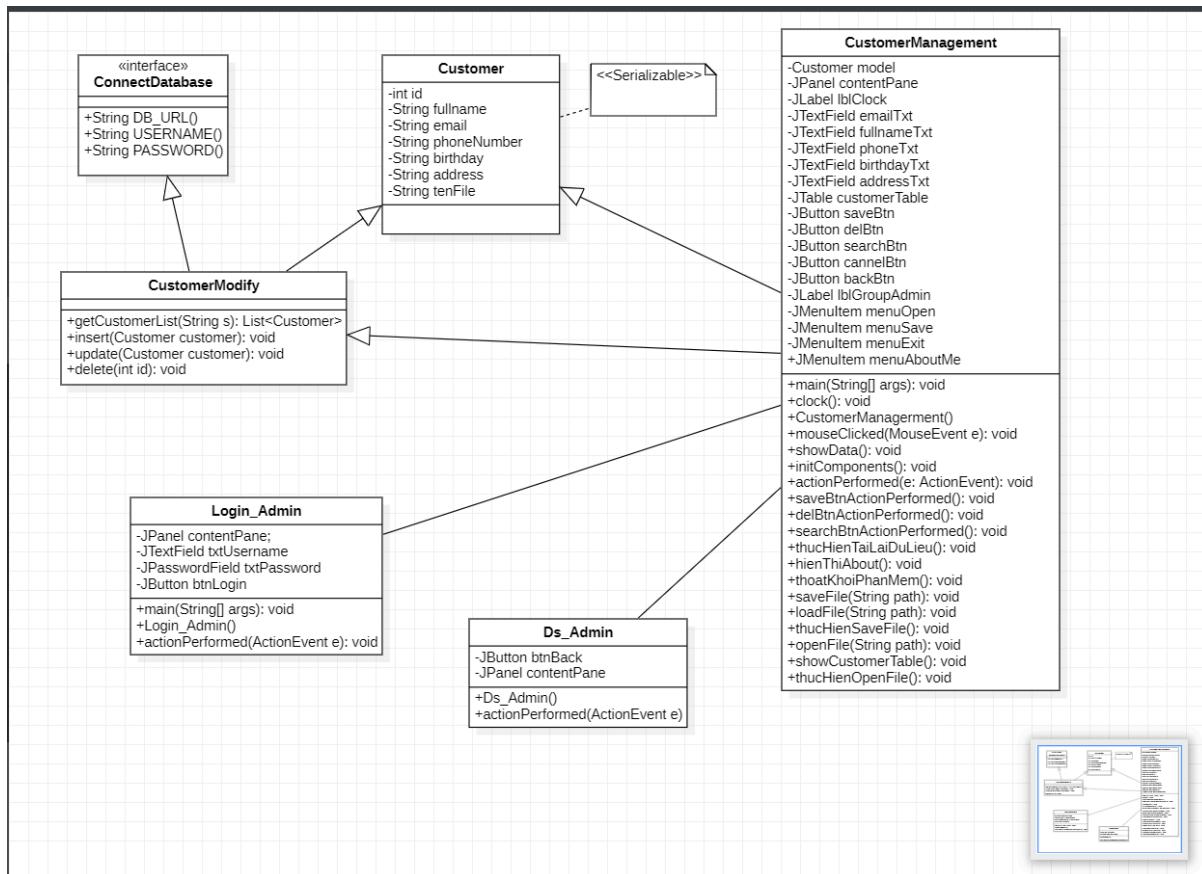
Lời cuối cùng, nhóm em xin kính chúc Thầy nhiều sức khỏe, thành công và hạnh phúc.

PHẦN I: ĐỀ TÀI

Quản lý khách hàng: Yêu cầu đầu vào: thông tin khách hàng (tên, địa chỉ, số điện thoại, email,...), các bước thực hiện: thêm, xóa, sửa, hiển thị danh sách, tìm kiếm khách hàng theo tên hoặc số điện thoại.

PHẦN II: NỘI DUNG THỰC HIỆN

1. Sơ đồ lớp



Hình 1: Sơ đồ lớp quản lý khách hàng

2. Xây dựng cơ sở dữ liệu

- Cơ sở dữ liệu là hệ thống bao gồm rất nhiều thông tin, dữ liệu được xây dựng theo một cấu trúc nhất định nhằm đáp ứng nhu cầu khai thác, sử dụng của nhiều người hay chạy nhiều chương trình ứng dụng cùng một lúc. Khi áp dụng hình thức lưu trữ này, nó sẽ giúp khắc phục được những điểm yếu của việc lưu file thông thường trên máy tính. Các thông tin lưu trữ sẽ đảm bảo được nhất quán, hạn chế tình trạng trùng lặp thông tin. Có nhiều cơ sở dữ liệu hiện nay thường dùng như Oracle, MySQL , MariaDB , MongoDB,....

- Trong bài này nhóm em sẽ sử dụng MySQL:



Hình 2: Logo hệ quản trị cơ sở dữ liệu MySQL

a. Tạo một database

Sử dụng lệnh “CREATE DATABASE customermanager” để tạo một database có tên là “customermanager”. Sau khi thực hiện lệnh này, có thể tạo các bảng và các đối tượng khác trong database để quản lý các thông tin liên quan đến khách hàng.

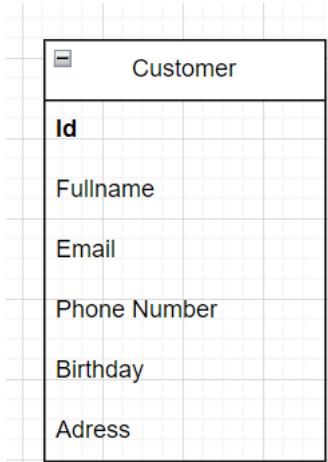
The screenshot shows the MySQL Workbench interface. In the Navigator pane, under the SCHEMAS section, the 'customermanager' database is listed and highlighted with a red box. In the central Query Editor window, the SQL command `create database customermanager` is entered and highlighted with a red box. Below the query editor, the Output pane shows the execution log with a single entry: `1 21:45:44 create database customermanager`, which is also highlighted with a red box. The bottom status bar indicates `No object selected`.

Hình 3: Database customermanager

b. Tạo một table trong database

Mô tả cơ sở dữ liệu với thực thể Customer (khách hàng) bao gồm các thuộc tính như sau:

Customer : ID , Fullname , Email , PhoneNumber , Birthday , Adress.

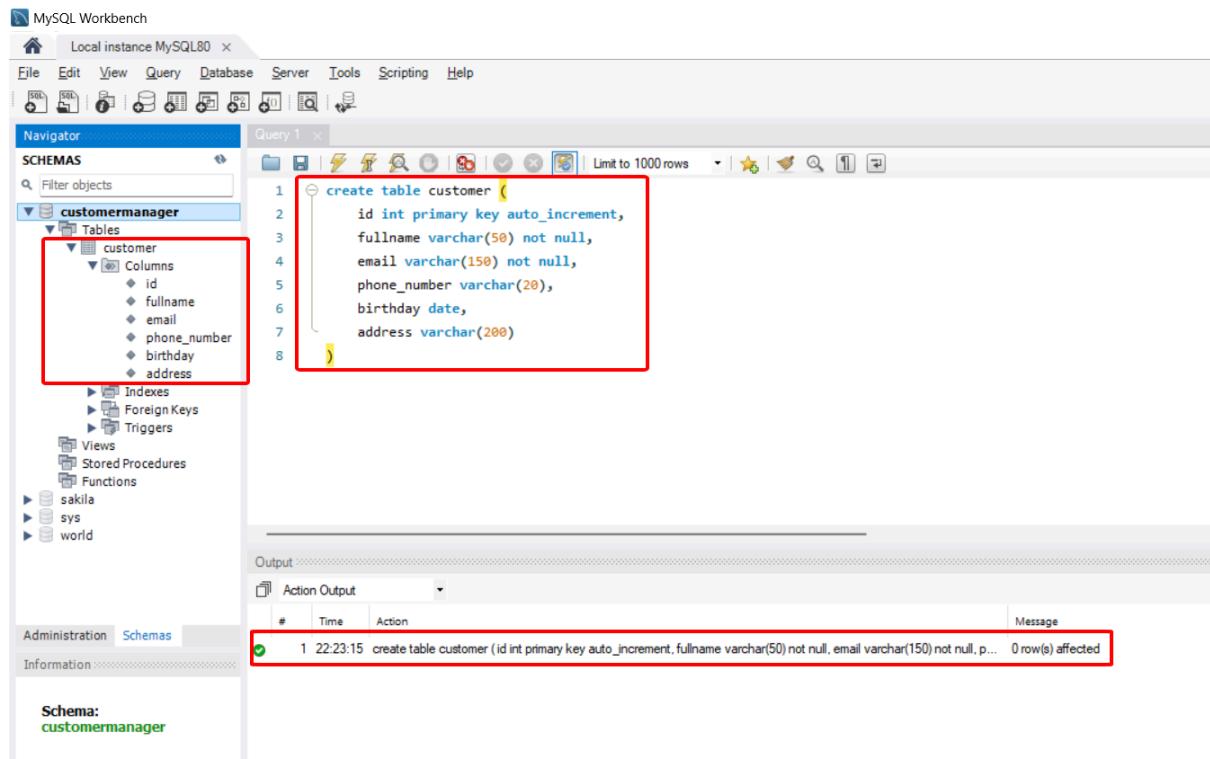


Hình 4: Mô tả cơ sở dữ liệu với thực thể Customer

Sử dụng lệnh sau để tạo ra một bảng có tên là “customer” với các cột được chỉ định:

```
CREATE TABLE customer (
    id int primary key auto_increment,
    fullname varchar(50) not null,
    email varchar(150) not null,
    phone_number varchar(20),
    birthday date,
    address varchar(200)
)
```

Nhóm 3 - Báo cáo môn học – Quản lý khách hàng



Hình 5: Bảng customer với các cột được chỉ định

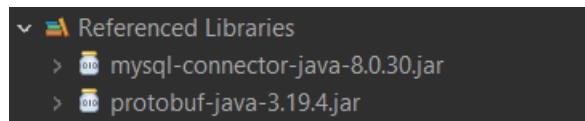
c. Liên kết cơ sở dữ liệu với project

- Trong Eclipse IDE, tạo một project “quanLyKhachHang”
- Để liên kết cơ sở dữ liệu với project “quanLyKhachHang”, ta cần cài đặt thư viện “MySQL JDBC Driver JAR”



Hình 6: Thư viện MySQL JDBC Driver JAR

- Tạo một package “Library”, copy thư viện JDBC vào Library
- Tiến hành xây dựng đường dẫn, sau khi xây dựng ta sẽ được:

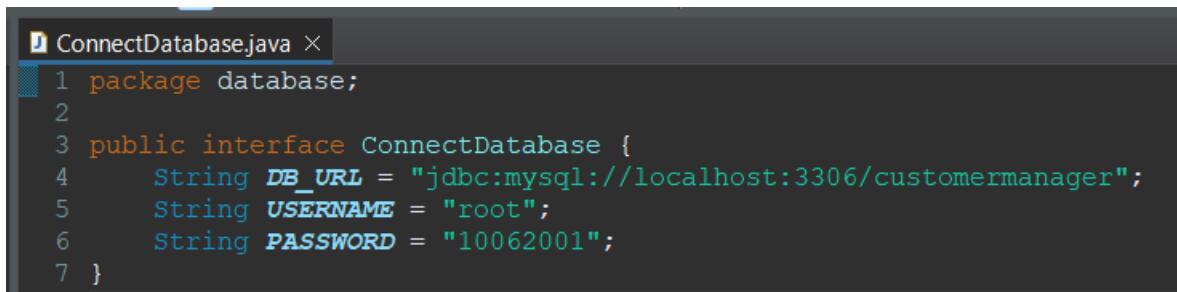


Hình 7: Thư viện tham chiếu

3. Xây dựng code

a. Database

Tại project tạo package “database”, sau đó tạo một interface “ConnectDatabase.java” trong package để khai báo thông tin đường dẫn kết nối tới MySQL (DB_URL), thông tin người dùng (USERNAME) và mật khẩu (PASSWORD).



```
1 package database;
2
3 public interface ConnectDatabase {
4     String DB_URL = "jdbc:mysql://localhost:3306/customermanager";
5     String USERNAME = "root";
6     String PASSWORD = "10062001";
7 }
```

b. Mô hình MVC (Model, View, Controller)

Mô hình MVC (Model-View-Controller) là một kiến trúc phát triển phần mềm được sử dụng rộng rãi trong Java Swing để xây dựng các ứng dụng có giao diện đồ họa.

❖ Model

- Là phần dữ liệu của ứng dụng
- Chứa các thuộc tính, phương thức để truy xuất, lưu trữ và xử lý dữ liệu
- Không có liên kết trực tiếp với View và Controller
 - Tại project tạo package “model”, sau đó tạo một class “Customer.java”. Lớp này đại diện cho một khách hàng trong một hệ thống quản lý khách hàng, gồm các thuộc tính và phương thức sau:

- Thuộc tính:

```
private int id;
private String fullname, email, phoneNumber, birthday, address, tenFile;
```

- o “id”: định danh của khách hàng, là một số nguyên
- o “fullname”: tên đầy đủ của khách hàng, là một chuỗi
- o “email”: địa chỉ email của khách hàng, là một chuỗi
- o “phoneNumber”: số điện thoại của khách hàng, là một chuỗi

- o “birthday”: ngày sinh của khách hàng, là một chuỗi
- o “address”: địa chỉ của khách hàng, là một chuỗi
- o “tenFile”: tên của tệp tin đính kèm, là một chuỗi
- Phương thức:

```
public Customer(int id, String fullname, String email, String phoneNumber, String birthday, String address) {  
    this.id = id;  
    this.fullname = fullname;  
    this.email = email;  
    this.phoneNumber = phoneNumber;  
    this.birthday = birthday;  
    this.address = address;  
    this.tenFile = "";  
}
```

“Customer(int id, String fullname, String email, String phoneNumber, String birthday, String address)”: phương thức khởi tạo lớp Customer, với các tham số là các thuộc tính của khách hàng.

```
public String getTenFile() {  
    return tenFile;  
}  
public void setTenFile(String tenFile) {  
    this.tenFile = tenFile;  
}
```

- o “getTenFile()”: phương thức trả về tên của tệp tin đính kèm
- o “setTenFile(String tenFile)”: phương thức thiết lập tên của tệp tin đính kèm

```
public int getId() {  
    return id;  
}  
public void setId(int id) {  
    this.id = id;  
}
```

- o “getId()”: phương thức trả về định danh của khách hàng
- o “setId(int id)”: phương thức thiết lập định danh của khách hàng

```
public String getFullscreen() {  
    return fullname;  
}  
public void setFullscreen(String fullname) {  
    this.fullname = fullname;  
}
```

- o “getFullscreen()”: phương thức trả về tên đầy đủ của khách hàng

- o “setFullscreen(String fullname)”: phương thức thiết lập tên đầy đủ của khách hàng

```
public String getEmail() {  
    return email;  
}  
public void setEmail(String email) {  
    this.email = email;  
}
```

- o “getEmail()”: phương thức trả về địa chỉ email của khách hàng
- o “setEmail(String email)”: phương thức thiết lập địa chỉ email của khách hàng

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

- o “getPhoneNumber()”: phương thức trả về số điện thoại của khách hàng
- o “setPhoneNumber(String phoneNumber)”: phương thức thiết lập số điện thoại của khách hàng

```
public String getBirthday() {  
    return birthday;  
}  
public void setBirthday(String birthday) {  
    this.birthday = birthday;  
}
```

- o “getBirthday()”: phương thức trả về ngày sinh của khách hàng
- o “setBirthday(String birthday)”: phương thức thiết lập ngày sinh của khách hàng

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

- o “getAddress()”: phương thức trả về địa chỉ của khách hàng

- o “setAddress(String address)”: phương thức thiết lập địa chỉ của khách hàng

```
@Override  
public String toString() {  
    return "Customer [id=" + id + ", fullname=" + fullname + ", email=" + email + ", phoneNumber=" + phoneNumber  
           + ", birthday=" + birthday + ", address=" + address + "]";  
}
```

- Lớp Customer cũng có một phương thức “toString()” để trả về thông tin đầy đủ về đối tượng khách hàng, bao gồm các giá trị của các trường id, fullname, email, phoneNumber, birthday, address

```
public class Customer implements Serializable
```

- Ngoài ra, lớp Customer được triển khai giao diện “Serializable”, cho phép đối tượng của lớp này có thể được tuần tự hóa và ghi vào một tệp. Điều này giúp cho việc lưu trữ và quản lý thông tin khách hàng trở nên tiện lợi hơn, đặc biệt là khi cần thiết phải lưu trữ thông tin khách hàng trong một tệp tin để tái sử dụng hoặc chuyển giao cho những người khác

❖ Controller

- Là phần điều khiển hành vi của ứng dụng.
 - Chứa các phương thức xử lý sự kiện được gửi từ View
 - Nhận dữ liệu từ Model
 - Thực hiện việc cập nhật View khi Model thay đổi và ngược lại
 - Liên kết giữa Model và View
- Tại project tạo package “controller”, tạo class “CustomerModify.java” trong package. Công việc chính của lớp “CustomerModify” là cung cấp các phương thức để tương tác với bảng “Customer” trong cơ sở dữ liệu. Nó bao gồm các phương thức để lấy danh sách khách hàng dựa trên tiêu chí tìm kiếm, chèn (insert) một bản ghi khách hàng mới, cập nhật (update) một bản ghi khách hàng hiện có và xóa (delete) một bản ghi khách hàng dựa trên ID. Lớp này sử dụng “JDBC API” để thiết lập kết nối với cơ sở dữ liệu và thực hiện các truy vấn SQL

```
CustomerModify.java ×
1 package controller;
2
3 import java.sql.Connection;
4
5 public class CustomerModify {
6
7     public static List<Customer> getCustomerList(String s) {
8         List<Customer> dataList = new ArrayList<>();
```

- Phương thức “getCustomerList” được sử dụng để truy vấn danh sách khách hàng từ cơ sở dữ liệu. Phương thức này nhận vào một tham số là chuỗi s kiểu String
 - Khởi tạo một danh sách “List<Customer>” với tên là “dataList” để chứa danh sách khách hàng. Khi có một khách hàng được truy vấn từ cơ sở dữ liệu, sẽ tạo một đối tượng khách hàng (Customer), sau đó đưa đối tượng này vào danh sách “dataList”. Cuối cùng, phương thức “getCustomerList” sẽ trả về danh sách “dataList” này cho người dùng sử dụng

□ Thiết lập database và tìm kiếm

• Thiết lập database

- Khởi tạo biến “conn” kiểu Connection và biến “statement” kiểu PreparedStatement để kết nối và thực thi câu lệnh SQL đến cơ sở dữ liệu:

```
Connection conn = null;
PreparedStatement statement = null;
```

- Trong khôi “try”, tạo kết nối với cơ sở dữ liệu thông qua DriverManager.getConnection(). Cụ thể, phương thức getConnection() của lớp DriverManager được gọi với 3 tham số:

- + ConnectDatabase.DB_URL: địa chỉ URL của cơ sở dữ liệu
- + ConnectDatabase.USERNAME: tên đăng nhập của người dùng để kết nối tới cơ sở dữ liệu
- + ConnectDatabase.PASSWORD: mật khẩu để kết nối tới cơ sở dữ liệu.

```
try {
    conn = DriverManager.getConnection(ConnectDatabase.DB_URL, ConnectDatabase.USERNAME, ConnectDatabase.PASSWORD);
```

• Truy vấn lấy thông tin khách hàng

```
String sql = "select * from customer where 1 = 1";
if(s != null && !s.isEmpty()) {
    sql += " and (fullname like ? or phone_number like ?)";
}
statement = conn.prepareStatement(sql);
if(s != null && !s.isEmpty()) {
    statement.setString(1, "%" + s + "%");
    statement.setString(2, "%" + s + "%");
}
ResultSet resultSet = statement.executeQuery();
```

- Khởi tạo chuỗi truy vấn sql với điều kiện WHERE luôn đúng (where 1 = 1). Điều này giúp cho việc thêm các điều kiện tìm kiếm sau này dễ dàng hơn.
- Kiểm tra xem tham số “s” có được truyền vào hay không. Nếu tham số “s” khác “null” và không trống (empty), sẽ thêm vào câu truy vấn điều kiện tìm kiếm theo fullname hoặc phone_number của khách hàng.
- Đối tượng PreparedStatement được tạo ra với chuỗi truy vấn sql. Nếu có điều kiện tìm kiếm, đối tượng này sẽ được thêm vào với hai tham số truyền vào tương ứng với fullname và phone_number được bao quanh bởi ký tự %, đại diện cho một chuỗi bất kỳ.
- Cuối cùng, đối tượng ResultSet được trả về khi thực hiện truy vấn bằng phương thức executeQuery() của đối tượng PreparedStatement.

- **Thiết lập lấy danh sách khách hàng (Customer)**

```
while(resultSet.next()) {
    Customer customer = new Customer(
        resultSet.getInt("id"),
        resultSet.getString("fullname"),
        resultSet.getString("email"),
        resultSet.getString("phone_number"),
        resultSet.getString("birthday"),
        resultSet.getString("address")
    );
    dataList.add(customer);
}
```

- Sử dụng vòng lặp While để lặp lại cho đến khi không còn bản ghi nào trong “resultSet”. Sau đó khai báo một đối tượng Customer để lưu trữ thông tin của mỗi khách hàng trong CSDL. Đối tượng này được khởi tạo bằng cách truyền các giá trị từ resultSet vào các tham số tương ứng trong hàm tạo của đối tượng Customer.

- Sau đó, sử dụng phương thức add để thêm đối tượng Customer vào danh sách dataList, nơi sẽ lưu trữ thông tin của tất cả các khách hàng trong CSDL.

- **Xử lý lỗi và đóng kết nối CSDL**

```

} catch (SQLException ex) {
    Logger.getLogger(CustomerModify.class.getName()).log(Level.SEVERE, null, ex);
} finally {
    if(statement != null) {
        try {
            statement.close();
        } catch (SQLException ex) {
            Logger.getLogger(CustomerModify.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    if(conn != null) {
        try {
            conn.close();
        } catch (SQLException ex) {
            Logger.getLogger(CustomerModify.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
return dataList;
}

```

- Sau khi truy vấn và lấy dữ liệu thành công thì khôi “catch” nơi xử lý các ngoại lệ (exceptions) của kiểu SQLException. SQLException là một ngoại lệ được lấy ra khi có lỗi xảy ra trong quá trình truy vấn CSDL. Phương thức getLogger() được sử dụng để tạo ra một đối tượng Logger để ghi log lỗi. Đối tượng Logger được truyền vào tên của lớp (CustomerModify) để đánh dấu nơi phát sinh lỗi.

- Khối “finally” kiểm tra nếu biến “statement” khác “null” thì sẽ đóng nó bằng cách gọi phương thức close(). Đây là một thao tác quan trọng để giải phóng tài nguyên và giúp tối ưu hóa hiệu suất ứng dụng. Tương tự kiểm tra nếu biến “conn” khác “null” thì sẽ đóng kết nối với CSDL bằng cách gọi phương thức close(). Việc đóng kết nối là quan trọng để giải phóng tài nguyên và tránh gây ra lỗi trong quá trình chạy ứng dụng.

- Cuối cùng, phương thức trả về danh sách dataList chứa thông tin của tất cả các khách hàng được truy vấn từ CSDL

- **Chèn (Insert) dữ liệu vào cơ sở dữ liệu**

```

public static void insert(Customer customer) {
    Connection conn = null;
    PreparedStatement statement = null;

    try {
        conn = DriverManager.getConnection(ConnectDatabase.DB_URL, ConnectDatabase.USERNAME, ConnectDatabase.PASSWORD);

```

- Khai báo “insert” với đối số là một đối tượng Customer, đối tượng này chứa thông tin về khách hàng mới sẽ được thêm vào CSDL. Sau đó, Khởi tạo hai biến “conn” và “statement” để sử dụng cho kết nối và truy vấn CSDL
- Tương tự “Thiết lập database” ở mục a, tạo kết nối cơ sở dữ liệu thông qua “DriverManager.getConnection” với 3 tham số: ConnectDatabase.DB_URL, ConnectDatabase.USERNAME và ConnectDatabase.PASSWORD

```

String sql = "insert into customer(fullname, email, phone_number, birthday, address) "
    + "values (?, ?, ?, ?, ?)";
statement = conn.prepareStatement(sql);
statement.setString(1, customer.getFullscreen());
statement.setString(2, customer.getEmail());
statement.setString(3, customer.getPhoneNumber());
statement.setString(4, customer.getBirthday());
statement.setString(5, customer.getAddress());
statement.execute();

```

- Khai báo một chuỗi SQL để insert dữ liệu vào bảng “customer” các trường dữ liệu tương ứng với thông tin của một đối tượng khách hàng, bao gồm: fullname, email, phone_number, birthday, và address. Sau đó, sử dụng phương thức “prepareStatement” để trả về một đối tượng preparedStatement thực hiện các thao tác trên cơ sở dữ liệu.
- Đối với mỗi trường dữ liệu của khách hàng, ta sẽ gọi phương thức “setString” của đối tượng preparedStatement để thiết lập giá trị của tham số tương ứng. Đôi số thứ nhất của phương thức này là chỉ số của tham số trong câu lệnh SQL(?) (bắt đầu từ 1)
- Phương thức “execute” của đối tượng PreparedStatement thực thi câu lệnh SQL thêm khách hàng mới vào cơ sở dữ liệu

```

} catch (SQLException ex) {
    Logger.getLogger(CustomerModify.class.getName()).log(Level.SEVERE, null, ex);
} finally {
    if(statement != null) {
        try {
            statement.close();
        } catch (SQLException ex) {
            Logger.getLogger(CustomerModify.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    if(conn != null) {
        try {
            conn.close();
        } catch (SQLException ex) {
            Logger.getLogger(CustomerModify.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
}

```

- Tương tự như “Xử lý lỗi và đóng kết nối CSDL” ở mục a, thì khôi “catch” xử lý ngoại lệ SQLException bằng cách ghi log lỗi và khôi “finally” đóng các đối tượng PreparedStatement và Connection để giải phóng tài nguyên và đóng kết nối CSDL. Nếu có lỗi xảy ra khi đóng, sẽ ghi log lỗi.

□ Cập nhật (update) thông tin khách hàng trong cơ sở dữ liệu

```
public static void update(Customer customer) {  
    Connection conn = null;  
    PreparedStatement statement = null;  
  
    try {  
        conn = DriverManager.getConnection(ConnectDatabase.DB_URL, ConnectDatabase.USERNAME, ConnectDatabase.PASSWORD);  
  
        String sql = "update customer set fullname = ?, email = ?, phone_number = ?, birthday = ?, address = ? where id = ?";  
        statement = conn.prepareStatement(sql);  
        statement.setString(1, customer.getFullscreen());  
        statement.setString(2, customer.getEmail());  
        statement.setString(3, customer.getPhoneNumber());  
        statement.setString(4, customer.getBirthday());  
        statement.setString(5, customer.getAddress());  
        statement.setInt(6, customer.getId());  
  
        statement.execute();  
    } catch (SQLException e) {  
        e.printStackTrace();  
    } finally {  
        if (statement != null) {  
            try {  
                statement.close();  
            } catch (SQLException e) {  
                e.printStackTrace();  
            }  
        }  
        if (conn != null) {  
            try {  
                conn.close();  
            } catch (SQLException e) {  
                e.printStackTrace();  
            }  
        }  
    }  
}
```

- Cũng tương tự như “insert”, khai báo sql để cập nhật thông tin khách hàng trong bảng “customer”

```
String sql = "update customer set fullname = ?, email = ?, phone_number = ?, birthday = ?, address = ? where id = ?";  
statement = conn.prepareStatement(sql);  
statement.setString(1, customer.getFullscreen());  
statement.setString(2, customer.getEmail());  
statement.setString(3, customer.getPhoneNumber());  
statement.setString(4, customer.getBirthday());  
statement.setString(5, customer.getAddress());  
statement.setInt(6, customer.getId());  
  
statement.execute();
```

- Sử dụng các cột: fullname, email, phone_number, birthday, address trong bảng customer để update các giá trị mới cho một khách hàng được xác định bằng cột “id”. Các giá trị mới được truyền vào câu truy vấn thông qua các tham số “?” được đặt tên theo thứ tự của các cột tương ứng trong câu truy vấn.

- Thiết lập các tham số cho câu lệnh truy vấn bằng các phương thức setString() và setInt() của đối tượng “PreparedStatement” và tham chiếu đến sql.

- Sau đó, thực thi câu lệnh truy vấn với phương thức execute() của đối tượng PreparedStatement để cập nhật thông tin khách hàng trong cơ sở dữ liệu.

- Tương tự “xử lý lỗi” thì khôi “catch”, xử lý ngoại lệ SQLException bằng cách ghi log lỗi, khôi “finally” đóng các đối tượng “PreparedStatement” và Connection để giải phóng tài nguyên và đóng kết nối CSDL. Nếu có lỗi xảy ra khi đóng sẽ ghi log lỗi.

□ Xoá (delete) thông tin khách hàng

```
public static void delete(int id) {  
    Connection conn = null;  
    PreparedStatement statement = null;  
  
    try {  
        conn = DriverManager.getConnection(ConnectDatabase.DB_URL, ConnectDatabase.USERNAME, ConnectDatabase.PASSWORD);  
  
        String sql = "delete from customer where id = ?";  
        statement = conn.prepareStatement(sql);  
        statement.setInt(1, id);  
  
        statement.executeUpdate();  
    } catch (SQLException e) {  
        e.printStackTrace();  
    } finally {  
        if (statement != null) {  
            try {  
                statement.close();  
            } catch (SQLException e) {  
                e.printStackTrace();  
            }  
        }  
        if (conn != null) {  
            try {  
                conn.close();  
            } catch (SQLException e) {  
                e.printStackTrace();  
            }  
        }  
    }  
}
```

- Tương tự như “insert” và “update” sau khi khai báo “delete” (xóa khách hàng khỏi cơ sở dữ liệu theo ID truyền vào) , Khởi tạo hai biến “conn” và “statement” để sử dụng cho kết nối và truy vấn CSDL và thiết lập kết nối đến cơ sở dữ liệu thông qua DriverManager.getConnection()

```
String sql = "delete from customer where id = ?";
statement = conn.prepareStatement(sql);
statement.setInt(1, id);

statement.execute();
```

- Khởi tạo biểu thức SQL với chuỗi “delete from customer where id = ?”. Đây là biểu thức SQL để xóa một bản ghi khách hàng với điều kiện là “id” trùng với tham số được truyền vào. Sử dụng conn.prepareStatement() để chuẩn bị biểu thức SQL.

- Sử dụng statement.setInt() để đặt tham số vào biểu thức SQL. Tham số này chính là “id” của khách hàng cần xóa và sử dụng statement.execute() để thực thi biểu thức SQL và xóa bản ghi khách hàng.

```
} catch (SQLException ex) {
    Logger.getLogger(CustomerModify.class.getName()).log(Level.SEVERE, null, ex);
} finally {
    if(statement != null) {
        try {
            statement.close();
        } catch (SQLException ex) {
            Logger.getLogger(CustomerModify.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    if(conn != null) {
        try {
            conn.close();
        } catch (SQLException ex) {
            Logger.getLogger(CustomerModify.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

- Sử dụng khối “catch”, xử lý ngoại lệ SQLException bằng cách ghi log lỗi và khối “finally” đóng các đối tượng preparedStatement và Connection để giải phóng tài nguyên và đóng kết nối CSDL. Nếu có lỗi xảy ra khi đóng sẽ ghi log lỗi.

❖ View

- Là phần giao diện người dùng (GUI) của ứng dụng.
- Chứa các thành phần như JButton, JLabel, JTable,..

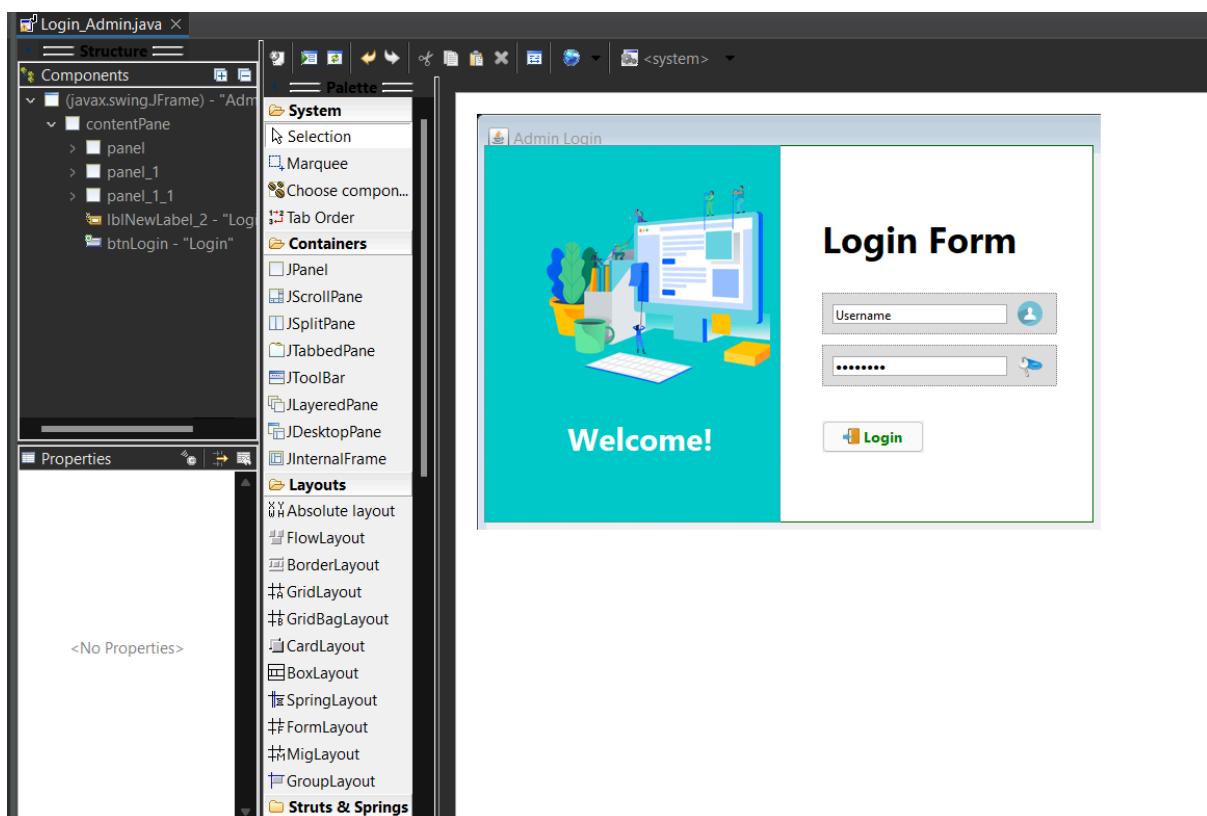
- Được sử dụng để hiển thị thông tin từ Model và tương tác với người dùng.
- Không có liên kết trực tiếp với Model và Controller
 - Tại project tạo package “view”, tạo 3 JFrame “Login_Admin.java”, “Ds_Admin.java” và “CustomerManagement.java” trong package.

□ **Login_Admin.java**

Đây là một JFrame được định nghĩa để tạo một giao diện đăng nhập cho người dùng đăng nhập vào tài khoản quản trị viên.

✓ **Giao diện GUI**

Cài đặt WindowsBuilder trên Eclipse IDE và tiến hành design, khi design sẽ hiển thị những đoạn code tương ứng bên phần “Source code”



✓ **Xử lý chức năng**

```

public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                Login_Admin frame = new Login_Admin();
                frame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}

```

- Lớp chứa một phương thức main() để khởi chạy ứng dụng. Phương thức này sử dụng một đối tượng EventQueue để đưa các tác vụ vào hàng đợi để được thực hiện trong tương lai. Trong phần khởi tạo, đối tượng Login_Admin được tạo ra và hiển thị lên màn hình. Nếu có bất kỳ lỗi nào xảy ra, nó sẽ được in ra trong bản ghi nhật ký của chương trình.

- **Thực hiện thiết lập đăng nhập:**

```

btnLogin.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String username = txtUsername.getText();
        String password = txtPassword.getPassword().toString();

        // login
        if (username.contains("Quang Trieu") || password.contains("1234567890")) {
            txtUsername.setText(null);
            txtPassword.setText(null);

            // link
            CustomerManagement.main(null);
        }
        else if (username.contains("Admin") || password.contains("1234567890")){
            txtUsername.setText(null);
            txtPassword.setText(null);

            // đóng form hiện tại
            JFrame currentFrame = (JFrame) SwingUtilities.getWindowAncestor(btnLogin);
            currentFrame.dispose();
        }
        // link
        CustomerManagement.main(null);
    }
    else {
        JOptionPane.showMessageDialog(null, "Invalid Login Details", "Login Error", JOptionPane.ERROR_MESSAGE);
        txtUsername.setText(null);
        txtPassword.setText(null);
    }
}
));

```

- Phương thức “actionPerformed(ActionEvent e)” là phương thức xử lý sự kiện khi người dùng nhấn vào nút đăng nhập. Khi người dùng bấm vào button Login, một hành động được thực hiện bởi phương thức actionPerformed() của ActionListener.

- Lấy giá trị đầu vào từ hai trường văn bản trên giao diện người dùng: username và password. Sau đó, nó kiểm tra xem tên đăng nhập và mật khẩu có phù hợp với thông tin được đặt sẵn hay không.

- Nếu thông tin đăng nhập hợp lệ, chúng ta mở giao diện quản lý khách hàng (CustomerManagement).
- Nếu không hợp lệ, chúng ta hiển thị một hộp thoại lỗi và xóa thông tin tài khoản và mật khẩu để người dùng có thể thử lại.

```
    ...
    btnLogin.setBounds(325, 265, 98, 31);
    contentPane.add(btnLogin);

    setLocationRelativeTo(null);

    this.setVisible(true);
```

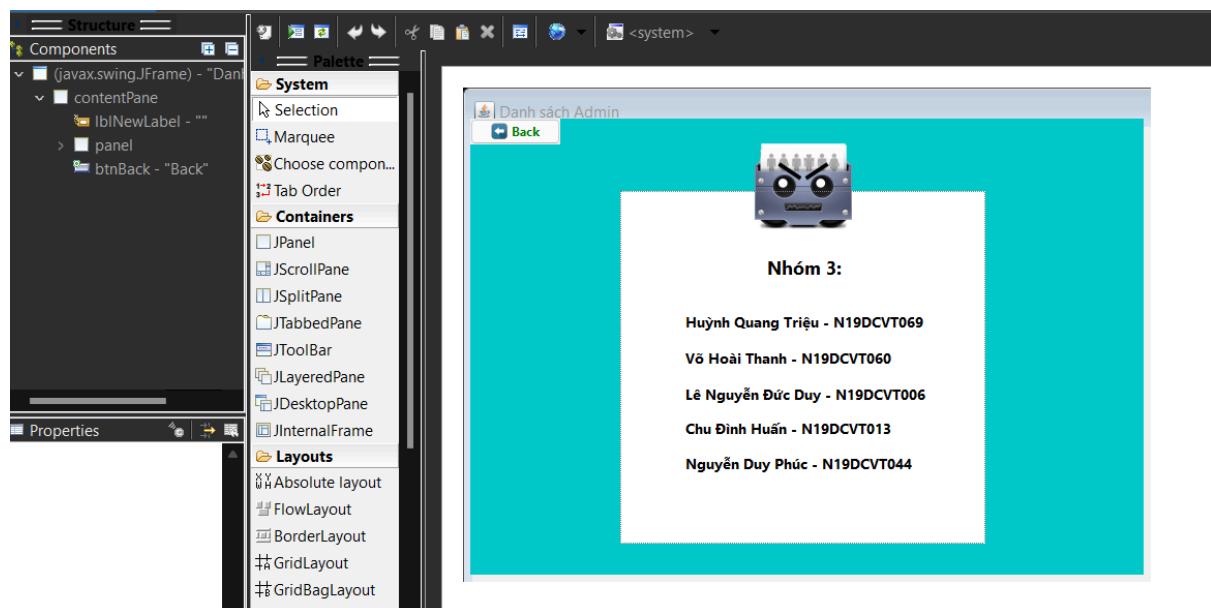
- Phương thức “setLocationRelativeTo(null)” được sử dụng để cửa sổ hiện tại được đặt ở giữa màn hình.

□ Ds_Admin.java

Đây là một JFrame được định nghĩa để hiển thị danh sách các thành viên trong nhóm.

✓ Giao diện GUI

Cài đặt WindowsBuilder trên Eclipse IDE và tiến hành design, khi design sẽ hiển thị những đoạn code tương ứng bên phần “Source code”



✓ Xử lý chức năng

```
btnBack.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        CustomerManagement.main(null);
        JFrame currentFrame = (JFrame) SwingUtilities.getWindowAncestor(btnBack);
        currentFrame.dispose();
    }
});
```

Khi người dùng nhấn vào nút “Back”, đối tượng ActionListener được gắn vào nút sẽ thực hiện hành động, trong trường hợp này là chuyển đến giao diện chính của Ứng dụng bằng cách gọi phương thức main() của lớp CustomerManagement và đóng cửa sổ hiện tại bằng cách gọi phương thức dispose của JFrame.

```
});  
btnBack.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\back.png"));  
btnBack.setFont(new Font("Segoe UI", Font.BOLD, 12));  
btnBack.setBounds(0, 0, 87, 25);  
contentPane.add(btnBack);  
  
setLocationRelativeTo(null);  
}]|
```

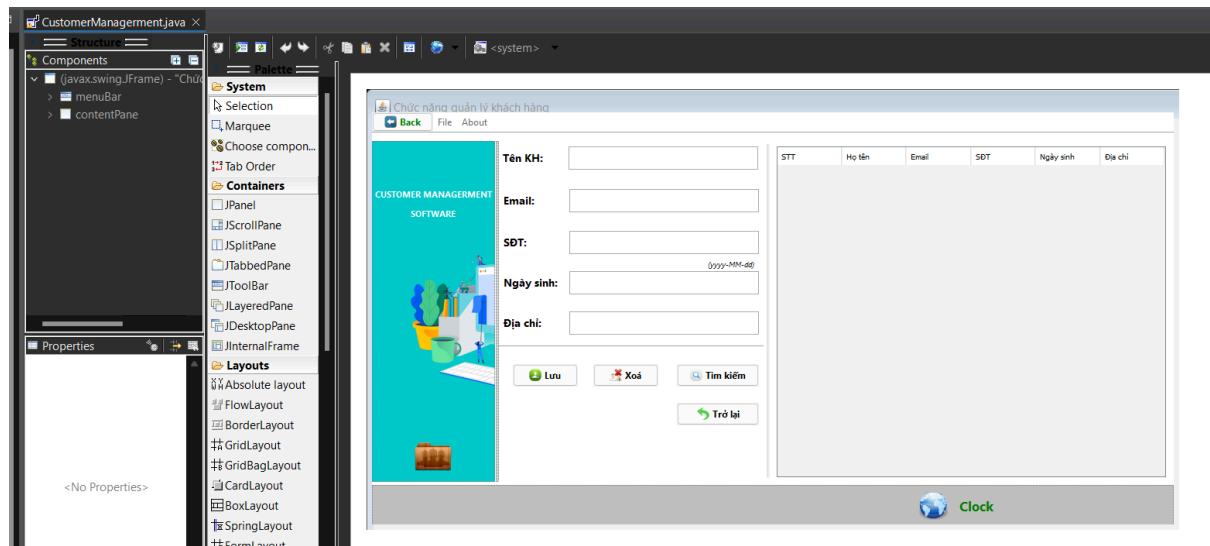
Phương thức “setLocationRelativeTo(null)” được sử dụng để cửa sổ hiện tại được đặt ở giữa màn hình.

□ **CustomerManagement.java**

Đây là JFrame CustomerManagement giúp cho người dùng quản lý khách hàng của họ, thực hiện các hành động: thêm, xoá, sửa, tìm kiếm, hiển thị,..

✓ **Giao diện GUI**

Cài đặt WindowsBuilder trên Eclipse IDE và tiến hành design, khi design sẽ hiển thị những đoạn code tương ứng bên phần “Source code”.



✓ Xử lý chức năng

```
//SQL
DefaultTableModel tableModel;
List<Customer> dataList;
int currentPos = -1;

public static void main(String[] args) {

    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                CustomerManagement frame = new CustomerManagement();
                frame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}
```

- Khai báo biến tableModel kiểu DefaultTableModel, sử dụng để lưu trữ dữ liệu cho một JTable.
- Khai báo biến dataList kiểu List<Customer>, giả định rằng đối tượng Customer chứa thông tin về khách hàng.
- Khai báo biến currentPos kiểu số nguyên và khởi tạo giá trị ban đầu là -1.
- Lớp cũng chứa một phương thức main() để khởi chạy ứng dụng. Phương thức này sử dụng một đối tượng EventQueue để đưa các tác vụ vào hàng đợi để được thực hiện trong tương lai. Trong phần khởi tạo, đối tượng

CustomerManagement được tạo ra và hiển thị lên màn hình. Nếu có bất kỳ lỗi nào xảy ra, nó sẽ được in ra trong bản ghi nhật ký của chương trình.

- **Hiển thị date, time**

```
public void clock() {
    Thread clock = new Thread() {
        public void run() {
            try {
                for(;;) {
                    Calendar cal = new GregorianCalendar();
                    int day = cal.get(Calendar.DAY_OF_MONTH);
                    int month = cal.get(Calendar.MONTH) + 1;
                    int year = cal.get(Calendar.YEAR);

                    int second = cal.get(Calendar.SECOND);
                    int minute = cal.get(Calendar.MINUTE);
                    int hour = cal.get(Calendar.HOUR);

                    lblClock.setText("Time " + hour + ":" + minute + ":" + second + " | " + " Date " + year + "/" + month + "/" + day);

                    sleep(1000);
                }
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    };
    clock.start();
}
```

- Phương thức clock() được sử dụng để hiển thị đồng hồ và ngày giờ hiện tại trên giao diện người dùng.

- Trong phương thức clock(), tạo một đối tượng luồng (Thread) bằng cách khởi tạo một lớp vô danh bên trong phương thức. Lớp vô danh này kế thừa lớp Thread và định nghĩa một phương thức run() để chạy luồng.

- Trong phương thức run(), sử dụng lớp Calendar để lấy thông tin về ngày, tháng, năm, giờ, phút và giây hiện tại từ hệ thống. Sau đó, định dạng và hiển thị thông tin này trên nhãn lblClock.

- Phương thức sleep() được gọi để đảm bảo rằng đồng hồ được cập nhật mỗi giây một lần.

- Cuối cùng, luồng được bắt đầu bằng cách gọi phương thức start() của đối tượng luồng Thread được tạo ra.

- **Khởi tạo lớp CustomerManagement**

```
public CustomerManagement() {
    this.model = new Customer(currentPos, getName(), getName(), getTitle(), getWarningString(), getName());
    initComponents();
    clock();

    //SQL
    tableModel = (DefaultTableModel) customerTable.getModel();
    dataList = CustomerModify.getCustomerList(null);

    showData();
}
```

- Khởi tạo của lớp CustomerManagement, được gọi khi một đối tượng CustomerManagement được tạo ra.
 - Trong hàm này, một đối tượng khách hàng mới được tạo ra bằng cách sử dụng constructor của lớp Customer.
 - Phương thức initComponents() được gọi để khởi tạo các thành phần giao diện người dùng.
 - Phương thức clock() cũng được gọi để bắt đầu đồng hồ hiển thị thời gian thực.
 - Sau đó, hàm tiếp tục khởi tạo các thành phần để truy xuất cơ sở dữ liệu SQL, bao gồm một đối tượng DefaultTableModel để hiển thị dữ liệu khách hàng trên bảng và một danh sách đối tượng khách hàng được lấy từ cơ sở dữ liệu bằng cách gọi phương thức getCustomerList(null) trong lớp CustomerModify.
 - Hàm showData() được gọi để hiển thị dữ liệu khách hàng trong danh sách lên bảng.

```
customerTable.addMouseListener(new MouseListener() {  
    public void mouseReleased(MouseEvent e) {  
  
    }  
    public void mousePressed(MouseEvent e) {  
  
    }  
    public void mouseExited(MouseEvent e) {  
  
    }  
    public void mouseEntered(MouseEvent e) {  
  
    }  
  
    public void mouseClicked(MouseEvent e) {  
        // TODO Auto-generated method stub  
        currentPos = customerTable.getSelectedRow();  
  
        fullnameTxt.setText(dataList.get(currentPos).getFullname());  
        emailTxt.setText(dataList.get(currentPos).getEmail());  
        phoneTxt.setText(dataList.get(currentPos).getPhoneNumber());  
        birthdayTxt.setText(dataList.get(currentPos).getBirthday());  
        addressTxt.setText(dataList.get(currentPos).getAddress());  
  
    }  
});  
}  
}
```

- Một “MouseListener” được thêm vào bảng khách hàng để theo dõi sự kiện nhấp chuột của người dùng trên bảng. Khi người dùng nhấp chuột vào một hàng trong bảng, các trường dữ liệu liên quan đến khách hàng được chọn sẽ được hiển thị trên các trường nhập liệu tương ứng, bao gồm tên đầy đủ, email, số điện thoại, ngày sinh và địa chỉ.

- **Hiển thị dữ liệu khách hàng lên bảng**

```
//ham hien thi du lieu len bang
public void showData() {
    tableModel.setRowCount(0);
    for (Customer customer : dataList) {
        tableModel.addRow(new Object[] {
            tableModel.getRowCount() + 1,
            customer.getFullname(),
            customer.getEmail(),
            customer.getPhoneNumber(),
            customer.getBirthday(),
            customer.getAddress()
        });
    }
}
```

- Phương thức showData() có tác dụng hiển thị dữ liệu khách hàng lên bảng đã tạo.

- Phương thức gọi hàm “setRowCount” của đối tượng “tableModel” để thiết lập số lượng hàng của bảng là 0. Điều này giúp xóa hết các hàng cũ trong bảng trước khi hiển thị dữ liệu mới.

- Sử dụng vòng lặp “for-each” để lấy thông tin của từng khách hàng, lặp qua danh sách khách hàng “dataList”

- Sau đó, phương thức gọi hàm “addRow” của đối tượng “tableModel” để thêm một hàng mới vào bảng. Giá trị của mỗi ô trong hàng mới được đặt trong một mảng đối tượng (Object[]). Giá trị đầu tiên trong mảng là số thứ tự hàng, được tính bằng cách lấy số lượng hàng hiện tại của bảng “tableModel.getRowCount()” cộng thêm 1.

- Các giá trị còn lại trong mảng đối tượng là các thuộc tính của đối tượng Customer, bao gồm: fullname, email, phoneNumber, birthday và address.

- Khi phương thức kết thúc, bảng sẽ hiển thị danh sách khách hàng mới nhất.

- Khởi tạo các thành phần giao diện (UI) của ứng dụng.

```
public void initComponents() {
    setTitle("Chức năng quản lý khách hàng");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 1050, 569);

    JMenuBar menuBar = new JMenuBar();
    setJMenuBar(menuBar);

    JButton BackBtn = new JButton("Back");
    BackBtn.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\back.png"));
    menuBar.add(BackBtn);
    BackBtn.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            // Đóng form hiện tại
            JFrame currentFrame = (JFrame) SwingUtilities.getWindowAncestor(BackBtn);
            currentFrame.dispose();

            // Mở form mới
            Login_Admin.main(null);
        }
    });
    BackBtn.setFont(new Font("Segoe UI", Font.BOLD, 12));
    BackBtn.setForeground(new Color(0, 128, 0));
    BackBtn.setBackground(UIManager.getColor("Button.background"));
}
```

- Đặt tiêu đề cho cửa sổ ứng dụng là “Chức năng quản lý khách hàng” và thiết lập hành động mặc định khi đóng cửa sổ (JFrame.EXIT_ON_CLOSE).
- Tạo một nút “Back”, nút này được thêm vào thanh menu và thiết lập một hành động thông qua một đối tượng ActionListener được tạo ra thông qua cú pháp “new ActionListener() {...}”. Phương thức actionPerformed sẽ được gọi khi nút được nhấn, nó sẽ đóng cửa sổ hiện tại và mở một cửa sổ mới với lớp Login_Admin.

```

JMenu menuFile = new JMenu("File");
menuBar.add(menuFile);

JMenuItem menuOpen = new JMenuItem("Open");
menuOpen.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_O, InputEvent.CTRL_DOWN_MASK));
menuOpen.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\Open.png"));
menuFile.add(menuOpen);
menuOpen.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
    }
});

JMenuItem menuSave = new JMenuItem("Save");
menuSave.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_S, InputEvent.CTRL_DOWN_MASK));
menuSave.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\Save.png"));
menuFile.add(menuSave);
menuSave.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        thucHienSaveFile();
    }
});

JSeparator separator = new JSeparator();
menuFile.add(separator);

JMenuItem menuExit = new JMenuItem("Exit");
menuExit.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_F4, InputEvent.ALT_DOWN_MASK));
menuExit.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\Exit.png"));
menuFile.add(menuExit);
menuExit.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        thoatKhoiPhanMem();
    }
});

```

- Tạo một menu “File” trong menu bar của ứng dụng. Menu này có các mục “Open”, “Save” và “Exit” cho phép người dùng thực hiện các thao tác liên quan đến tệp tin.

- Các ActionListener để bắt sự kiện click chuột vào từng menu item, sau đó thực hiện các phương thức tương ứng. Cụ thể, khi click vào “Open”, phương thức thucHienOpenFile() sẽ được gọi; khi người dùng click vào “Save”, phương thức thucHienSaveFile() sẽ được gọi và khi người dùng click vào “Exit”, phương thức thoatKhoiPhanMem() sẽ được gọi.

- Tiếp theo tạo một menu item “About” để hiển thị thông tin về chương trình.

```

JMenu menuAbout = new JMenu("About");
menuBar.add(menuAbout);

JMenuItem menuAboutMe = new JMenuItem("About me");
menuAboutMe.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\About.png"));
menuAbout.add(menuAboutMe);
menuAboutMe.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        hienThiAbout();
    }
});

```

- Trong menu “About”, tạo một menu item “About me” và thêm một ActionListener để xử lý sự kiện khi người dùng chọn “About me”.

```
JLabel lblGroupAdmin = new JLabel("");
lblGroupAdmin.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        Ds_Admin.main(null);
        JFrame frame = (JFrame) SwingUtilities.getWindowAncestor(lblGroupAdmin);
        frame.setVisible(false);
    }
});
lblGroupAdmin.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\group_admin.png"));
lblGroupAdmin.setBounds(55, 380, 48, 54);
panel_1.add(lblGroupAdmin);
```

- lblGroupAdmin là một JLabel chứa một biểu tượng hình ảnh, khi được nhấp chuột vào, sẽ mở ra một cửa sổ để hiển thị danh sách các quản trị viên.

```
JButton saveBtn = new JButton("Lưu");
saveBtn.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\Add.png"));
saveBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        saveBtnActionPerformed(e);
    }
});
saveBtn.setFont(new Font("Segoe UI", Font.BOLD, 12));
saveBtn.setBounds(182, 298, 84, 29);
contentPane.add(saveBtn);

JButton delBtn = new JButton("Xoá");
delBtn.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\delete.png"));
delBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        delBtnActionPerformed(e);
    }
});
delBtn.setFont(new Font("Segoe UI", Font.BOLD, 12));
delBtn.setBounds(286, 298, 84, 29);
contentPane.add(delBtn);
```

Đoạn mã trên tạo hai đối tượng JButton để thực hiện các hoạt động khác nhau trong giao diện người dùng:

- saveBtn: Nút lưu này có sự kiện ActionListener để xử lý khi người dùng nhấn nút này. Phương thức xử lý được gọi khi nút này được nhấn là saveBtnActionPerformed().

- delBtn: Nút xoá có biểu tượng hình ảnh được đặt ở phía trước. Nút này có sự kiện ActionListener để xử lý khi người dùng nhấn nút này. Phương thức xử lý được gọi khi nút này được nhấn là delBtnActionPerformed().

```
JButton searchBtn = new JButton("Tìm kiếm");
searchBtn.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\search.png"));
searchBtn.setFont(new Font("Segoe UI", Font.BOLD, 12));
searchBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        searchBtnActionPerformed(e);
    }
});

searchBtn.setBounds(394, 298, 106, 29);
contentPane.add(searchBtn);
JButton cannelBtn = new JButton("Trở lại");
cannelBtn.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\go_back.png"));
cannelBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        thucHienTaiLaiDuLieu(e);
    }
});
cannelBtn.setFont(new Font("Segoe UI", Font.BOLD, 12));
cannelBtn.setBounds(394, 348, 106, 29);
contentPane.add(cannelBtn);
```

Đoạn mã trên tạo thêm hai đối tượng JButton để thực hiện các hoạt động khác nhau trong giao diện người dùng:

- searchBtn: Khi nhấn vào nút tìm kiếm này, sẽ thực thi hàm searchBtnActionPerformed().
- cannelBtn: Khi nhấn vào nút trở lại này, sẽ thực thi hàm thucHienTaiLaiDuLieu().

- **Xử lý sự kiện save và update**

- Đây là phương thức xử lý sự kiện khi người dùng nhấn nút “Lưu”.
- Nếu currentPos lớn hơn 0, thì các trường thông tin của khách hàng tại vị trí này sẽ được cập nhật bằng giá trị mới từ các trường thông tin trên giao diện. Phương thức CustomerModify.update() được gọi để cập nhật thông tin khách hàng trong cơ sở dữ liệu. Nếu currentPos < 0 (không có khách hàng nào đang được chọn) thì một khách hàng mới sẽ được tạo từ các trường thông tin trên giao diện và được thêm vào cơ sở dữ liệu thông qua phương thức CustomerModify.insert(). Danh sách khách hàng được cập nhật từ cơ sở dữ liệu và được hiển thị trên giao diện bằng phương thức showData(). Cuối cùng, các trường thông tin trên giao diện được xóa để chuẩn bị cho việc thêm khách hàng mới.

```

public void saveBtnActionPerformed(java.awt.event.ActionEvent evt) {
    if (currentPos >= 0) {
        dataList.get(currentPos).setFullscreen(fullnameTxt.getText());
        dataList.get(currentPos).setEmail(emailTxt.getText());
        dataList.get(currentPos).setPhoneNumber(phoneTxt.getText());
        dataList.get(currentPos).setBirthday(birthdayTxt.getText());
        dataList.get(currentPos).setAddress(addressTxt.getText());

        CustomerModify.update(dataList.get(currentPos));
        currentPos = -1;
    } else {
        Customer customer = new Customer(
            0,
            fullnameTxt.getText(),
            emailTxt.getText(),
            phoneTxt.getText(),
            birthdayTxt.getText(),
            addressTxt.getText()
        );
        CustomerModify.insert(customer);
        dataList = CustomerModify.getCustomerList(null);
    }
    showData();

    fullnameTxt.setText("");
    emailTxt.setText("");
    phoneTxt.setText("");
    birthdayTxt.setText("");
    addressTxt.setText("");
}
}

```

- Xử lý sự kiện delete

```

public void delBtnActionPerformed(java.awt.event.ActionEvent evt) {
    if(currentPos == -1) {
        JOptionPane.showMessageDialog(rootPane, "Chưa chọn khách hàng cần xoá, vui lòng kiểm tra lại.");
        return;
    }
    int option = JOptionPane.showConfirmDialog(rootPane, "Bạn có chắc muốn xoá khách hàng này không?");
    if(option == 0) {
        CustomerModify.delete(dataList.get(currentPos).getId());
        dataList.remove(currentPos);
        currentPos = -1;
        showData();
    }

    fullnameTxt.setText("");
    emailTxt.setText("");
    phoneTxt.setText("");
    birthdayTxt.setText("");
    addressTxt.setText("");
}
}

```

- Đây là phương thức xử lý sự kiện khi người dùng nhấn nút “Xoá” trên giao diện. Đầu tiên, phương thức kiểm tra xem người dùng đã chọn khách hàng cần xoá chưa bằng cách kiểm tra giá trị của biến currentPos.

- Nếu currentPos = -1 tức là chưa có khách hàng nào được chọn, phương thức sẽ hiển thị một hộp thoại thông báo lỗi và kết thúc phương thức. Nếu đã

chọn khách hàng, phương thức sẽ hiển thị một hộp thoại xác nhận việc xoá khách hàng. Nếu người dùng chọn “Đồng ý”, phương thức sẽ xoá khách hàng khỏi cơ sở dữ liệu thông qua lớp CustomerModify, cập nhật lại danh sách khách hàng và hiển thị danh sách khách hàng mới trên giao diện. Cuối cùng, phương thức đặt lại các giá trị của các trường nhập liệu về rỗng để chuẩn bị cho việc thêm khách hàng mới hoặc sửa đổi thông tin khách hàng khác.

- **Xử lý sự kiện search**

```
public void searchBtnActionPerformed(java.awt.event.ActionEvent evt) {  
    String s = JOptionPane.showInputDialog("Nhập tên khách hàng cần tìm kiếm.");  
    if(!s.isEmpty()) {  
        s = "%" + s + "%";  
    }  
    dataList = CustomerModify.getCustomerList(s);  
    showData();  
}
```

Đoạn code trên thực hiện việc tìm kiếm khách hàng theo tên hoặc số điện thoại. Khi người dùng click vào nút tìm kiếm, một hộp thoại yêu cầu người dùng nhập tên hoặc số điện thoại khách hàng cần tìm kiếm được hiển thị lên. Sau đó, chuỗi tìm kiếm sẽ được định dạng bằng cách thêm ký tự % ở đầu và cuối chuỗi, điều này sẽ giúp tìm kiếm được các khách hàng có tên hoặc số điện thoại chứa chuỗi nhập vào. Sau khi chuỗi tìm kiếm được định dạng, hàm getCustomerList sẽ được gọi với chuỗi này như là tham số để lấy danh sách khách hàng và hiển thị dữ liệu trên bảng.

- **Xử lý sự kiện tải lại dữ liệu**

```
public void thucHienTaiLaiDuLieu(java.awt.event.ActionEvent evt) {  
    dataList = CustomerModify.getCustomerList("");  
    showData();  
    JOptionPane.showMessageDialog(this, "Đã Loading!");  
}
```

Phương thức “thucHienTaiLaiDuLieu” dùng để tải lại toàn bộ dữ liệu khách hàng từ cơ sở dữ liệu và hiển thị lên giao diện. Khi được gọi, phương thức này sẽ gọi đến phương thức “getCustomerList” của lớp “CustomerModify” để lấy lại toàn bộ dữ liệu khách hàng. Sau đó, phương thức “showData” sẽ được gọi để hiển thị dữ liệu lên giao diện và một thông báo sẽ được hiển thị để thông báo cho người dùng biết là đã tải lại thành công.

- Xử lý sự kiện hiển thị About

```
public void hienThiAbout() {
    JOptionPane.showMessageDialog(this, "Phần mềm quản lý khách hàng của Nhóm 3.");
    showData();
}
```

Phương thức “hienThiAbout” được sử dụng để hiển thị thông tin về phần mềm cho người dùng. Khi được gọi, phương thức này sẽ hiển thị một hộp thoại “JOptionPane” chứa thông tin về phần mềm và sau đó gọi đến phương thức “showData” để hiển thị dữ liệu lên giao diện.

- Xử lý sự kiện thoát khỏi phần mềm

```
public void thoatKhoiPhanMem() {
    int option = JOptionPane.showConfirmDialog(rootPane, "Bạn có chắc chắn muốn thoát khỏi phần mềm quản lý khách hàng không?");
    if(option == 0) {
        System.exit(0);
        showData();
    }
}
```

Phương thức “thoatKhoiPhanMem”: được sử dụng để thoát khỏi chương trình. Khi được gọi, phương thức này sẽ hiển thị một hộp thoại “JOptionPane” để xác nhận việc thoát khỏi chương trình với người dùng. Nếu người dùng chọn “Yes”, chương trình sẽ thoát và đồng thời phương thức “showData” sẽ được gọi để hiển thị dữ liệu lên giao diện.

- Xử lý sự kiện save file

```
public void saveFile(String path) {
    try {
        this.model.setTenFile(path);
        FileWriter fileWriter = new FileWriter(path);
        BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);
        for (Customer customer : dataList) {
            StringBuilder builder = new StringBuilder();
            builder.append(customer.getFullname()).append(", ");
            builder.append(customer.getEmail()).append(", ");
            builder.append(customer.getPhoneNumber()).append(", ");
            builder.append(customer.getBirthday()).append(", ");
            builder.append(customer.getAddress());
            bufferedWriter.write(builder.toString());
            bufferedWriter.newLine();
        }
        bufferedWriter.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Phương thức “saveFile”: lưu dữ liệu trong danh sách “dataList” vào một file được chỉ định bởi đường dẫn “path”. Phương thức cập nhật đường dẫn tệp được lưu trữ trong đối tượng “model”. Sau đó, phương thức sử dụng một “FileWriter” để mở tệp và một “BufferedWriter” để ghi dữ liệu vào tệp. Với mỗi đối tượng “Customer” trong danh sách “dataList”, phương thức sử dụng một “StringBuilder” để tạo một chuỗi đại diện cho thông tin khách hàng và ghi chuỗi đó vào tệp bằng cách sử dụng phương thức “write” của “BufferedWriter”.

```
public void loadFile(String path) {
    try {
        dataList.clear();
        Scanner scanner = new Scanner(new File(path));
        while (scanner.hasNextLine()) {
            String line = scanner.nextLine();
            String[] fields = line.split(",");
            Customer customer = new Customer(currentPos, fields[0], fields[1], fields[2], fields[3], fields[4]);
            dataList.add(customer);
            currentPos++;
        }
        scanner.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Phương thức “loadFile”: nó đọc dữ liệu từ một file được chỉ định bởi đường dẫn “path” và lưu trữ chúng trong danh sách “dataList”. Các dòng trong file được lưu trữ trong một mảng các chuỗi “fields”. Sau đó, các phần tử của mảng “fields” được sử dụng để tạo một đối tượng “Customer” và được thêm vào danh sách “dataList”. Nếu có lỗi xảy ra trong quá trình đọc file, phương thức sẽ in ra thông báo lỗi.

```
public void thucHienSaveFile() {
    if (this.model.getTenFile().length() > 0) {
        String path = this.model.getTenFile() + ".txt";
        saveFile(path);
    } else {
        JFileChooser fs = new JFileChooser(new File("d:\\\\"));
        fs.setDialogTitle("Save a File");
        int result = fs.showSaveDialog(this);
        if (result == JFileChooser.APPROVE_OPTION) {
            File fi = fs.getSelectedFile();
            String path = fi.getAbsolutePath() + ".txt";
            saveFile(path);
        }
    }
}
```

Phương thức “thucHienSaveFile”: kiểm tra xem tên file đã được nhập vào chưa:

- Nếu tên file đã được nhập, phương thức sử dụng đối tượng “model” để lấy tên file và sau đó nối thêm đuôi mở rộng “.txt” vào cuối tên file để tạo ra đường dẫn đầy đủ. Sau đó, phương thức gọi phương thức “saveFile” với đường dẫn này.

- Nếu tên file chưa được nhập, phương thức tạo một đối tượng “JfileChooser” và hiển thị hộp thoại cho phép người dùng chọn đường dẫn lưu file. Nếu người dùng chọn một đường dẫn hợp lệ, phương thức lấy đường dẫn này và thêm đuôi mở rộng “.txt” vào cuối để tạo ra đường dẫn đầy đủ. Sau đó, phương thức gọi phương thức “saveFile” với đường dẫn này. Nếu người dùng không chọn một đường dẫn hợp lệ, phương thức không thực hiện bất kỳ thao tác nào.

• Xử lý sự kiện open file

```
public void openFile(String path) {
    try {
        FileInputStream fis = new FileInputStream(path);
        ObjectInputStream ois = new ObjectInputStream(fis);
        Object obj;
        List<Customer> customers = new ArrayList<Customer>();
        while ((obj = ois.readObject()) != null) {
            if (obj instanceof Customer) {
                Customer customer = (Customer) obj;
                customers.add(customer);
            }
        }
        ois.close();
        showCustomerTable(customers);
    } catch (EOFException e) {
        // End of file reached
    } catch (StreamCorruptedException e) {
        JOptionPane.showMessageDialog(this, "File không hợp lệ hoặc bị hỏng", "Lỗi", JOptionPane.ERROR_MESSAGE);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Phương thức “openFile”: nhận vào một đường dẫn tới file và mở file này để đọc danh sách khách hàng. Nó sử dụng “ObjectInputStream” để đọc từng đối tượng trong file, kiểm tra xem đối tượng có phải là một “Customer” và thêm đối tượng khách hàng này vào một danh sách. Sau khi đọc xong, phương thức gọi phương thức “showCustomerTable” để hiển thị danh sách khách hàng trong bảng trên giao diện. Nếu file không hợp lệ, phương thức sẽ hiển thị một thông báo lỗi.

```
public void showCustomerTable(List<Customer> customers) {
    JTable customerTable = new JTable();
    customerTable.setBackground(new Color(175, 238, 238));
    customerTable.setFont(new Font("Segoe UI", Font.PLAIN, 14));
    customerTable.setModel(new DefaultTableModel(
        new Object[][] {},
        new String[] {"STT", "Họ tên", "Email", "SĐT", "Ngày sinh", "Địa chỉ"})
    );
    DefaultTableModel model = (DefaultTableModel) customerTable.getModel();
    int i = 1;
    for (Customer customer : customers) {
        model.addRow(new Object[] {i, customer.getFullname(), customer.getEmail(),
            customer.getPhoneNumber(), customer.getBirthday(), customer.getAddress()});
        i++;
    }
    JScrollPane scrollPane = new JScrollPane(customerTable);
    JFrame frame = new JFrame("Danh sách khách hàng");
    frame.getContentPane().add(scrollPane, BorderLayout.CENTER);
    frame.setSize(800, 600);
    frame.setLocationRelativeTo(null);
    frame.setVisible(true);
}
```

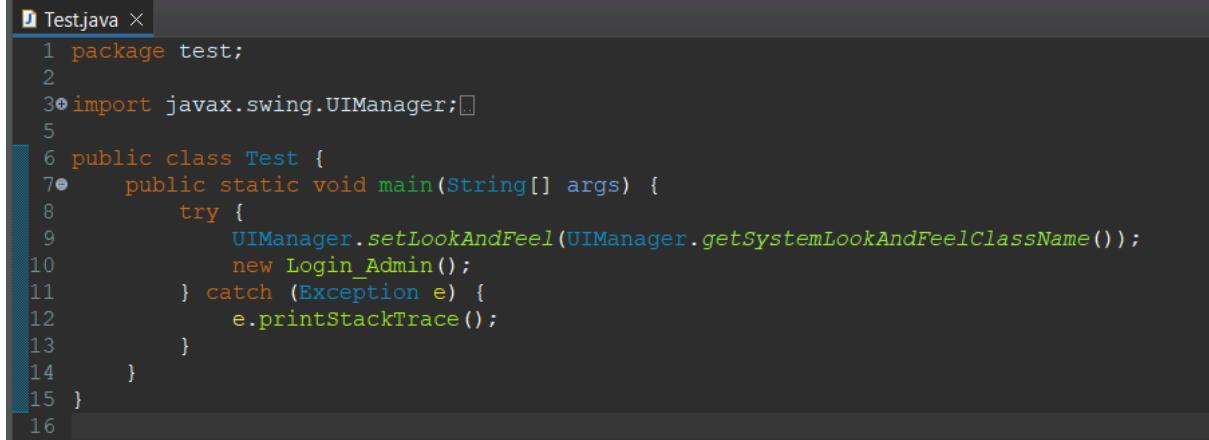
Phương thức “showCustomerTable”: nhận vào một danh sách khách hàng và hiển thị chúng trên một bảng trong một cửa sổ JFrame. Bảng sử dụng lớp JTable để hiển thị dữ liệu. Phương thức tạo ra một bảng JTable với các cột “STT”, “Họ tên”, “Email”, “SĐT”, “Ngày sinh”, “Địa chỉ” và thêm các hàng tương ứng với thông tin của từng khách hàng vào bảng. Cuối cùng, phương thức tạo ra một cửa sổ JFrame mới và chèn bảng vào đó để hiển thị danh sách khách hàng.

```
public void thucHienOpenFile() {
    JFileChooser fs = new JFileChooser(new File("d:\\\\"));
    fs.setDialogTitle("Open a File");
    int result = fs.showOpenDialog(this);
    if (result == JFileChooser.APPROVE_OPTION) {
        File fi = fs.getSelectedFile();
        loadFile(fi.getAbsolutePath());
        showCustomerTable(dataList);
    }
}
```

Phương thức “thucHienOpenFile”: dùng để mở một tập tin chứa danh sách khách hàng đã được lưu trữ trước đó. Khi người dùng nhấn vào nút “Open” trong giao diện, phương thức sẽ hiển thị một hộp thoại “Open a File” để người dùng chọn tập tin cần mở. Nếu người dùng chọn một tập tin và nhấn OK, phương thức sẽ gọi hai phương thức khác là loadFile() để tải danh sách khách hàng từ tập tin đó và showCustomerTable() để hiển thị danh sách đó trên một bảng.

c. Test

Tại project tạo package “test”, tạo class “Test.java” trong package có chức năng là khởi chạy chương trình.



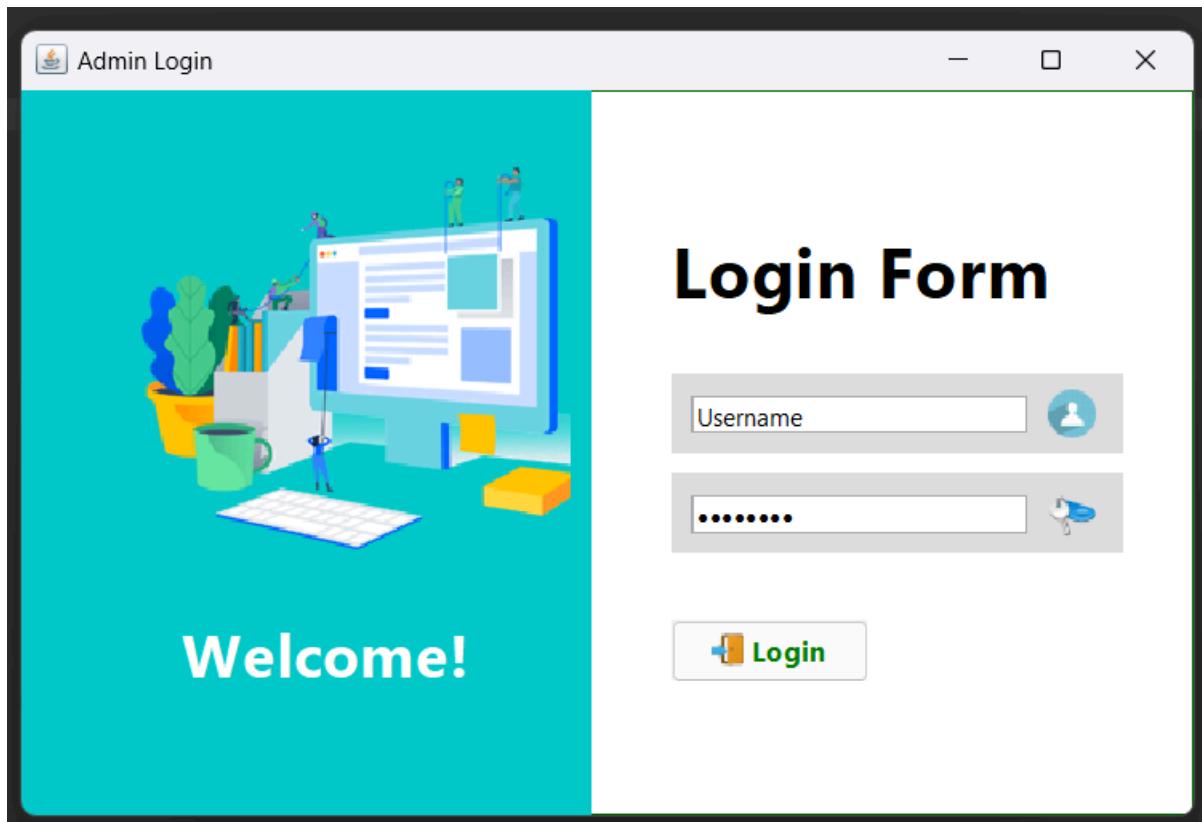
```
Test.java ×
1 package test;
2
3 import javax.swing.UIManager;
4
5
6 public class Test {
7     public static void main(String[] args) {
8         try {
9             UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
10            new Login_Admin();
11        } catch (Exception e) {
12            e.printStackTrace();
13        }
14    }
15}
16
```

Phương thức main trong class Test được sử dụng để bắt đầu chương trình. Trong phương thức này, đầu tiên sử dụng UIManager để đặt giao diện của ứng dụng thành giao diện hệ thống (system look and feel). Sau đó, tạo một đối tượng của lớp Login_Admin, đây là giao diện đăng nhập của quản trị viên của chương trình.

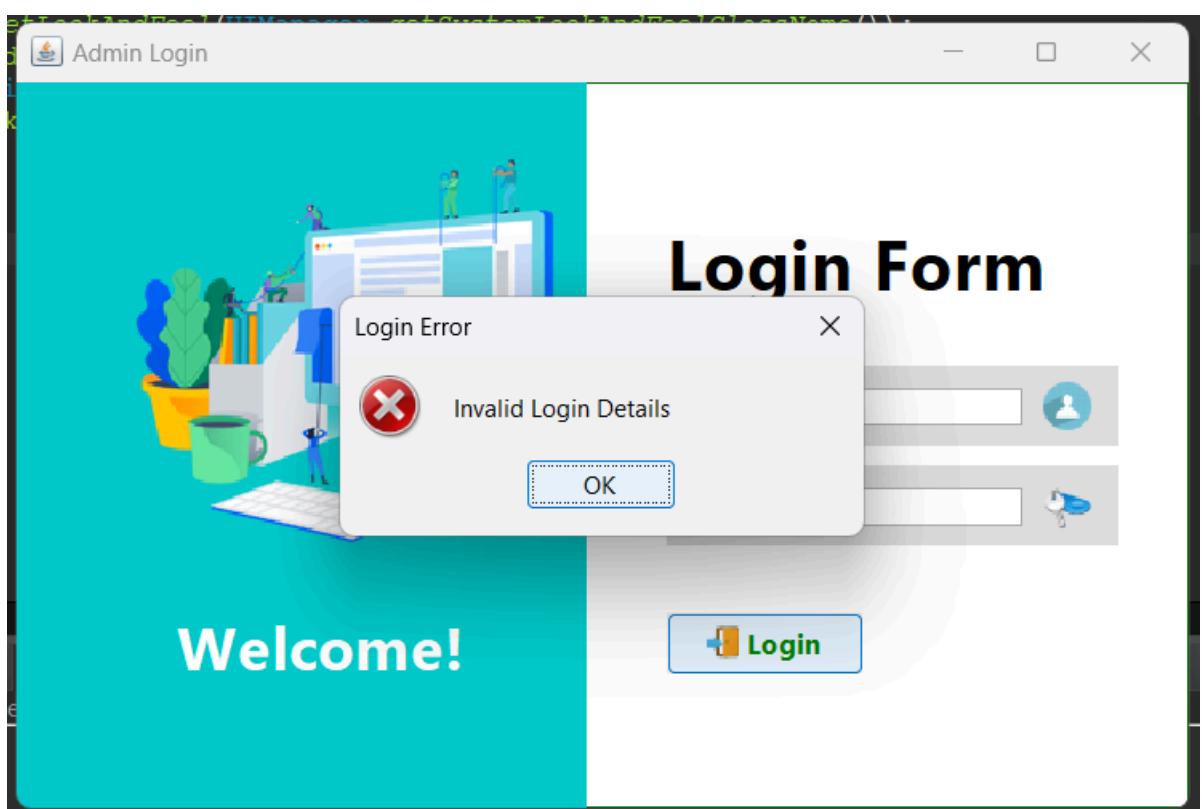
PHẦN III: KẾT QUẢ CHẠY CHƯƠNG TRÌNH

1. Login Admin

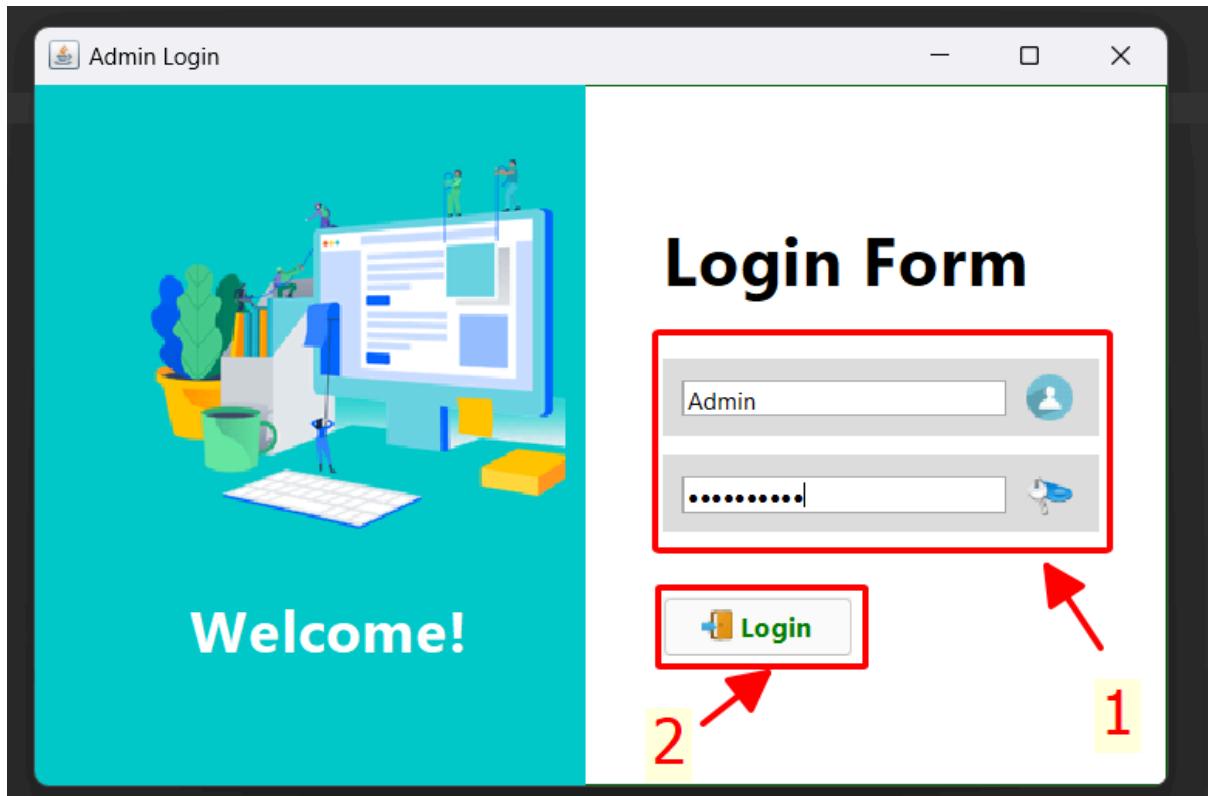
- Khi chạy chương trình “Test.java”, giao diện đăng nhập của quản trị viên “Login_Admin” sẽ xuất hiện:



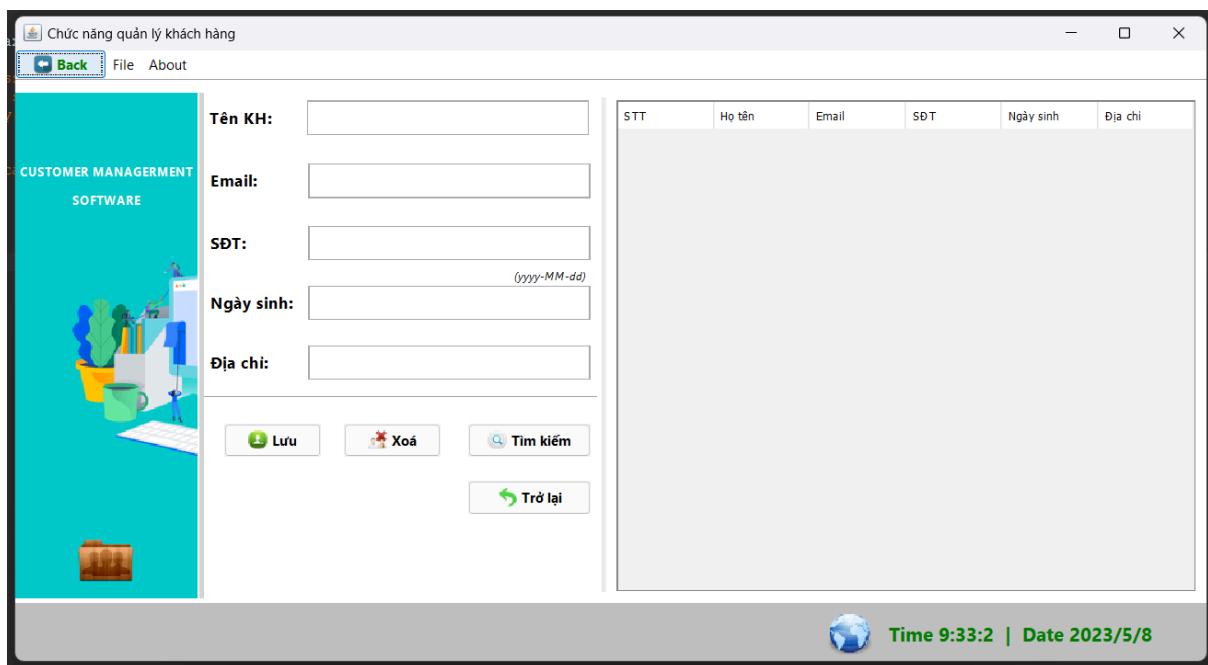
- Nếu chưa nhập “Username” và “Password” mà bấm nút “Login” sẽ hiện thông báo lỗi:



- Tiến hành nhập “Username” và “Password” sau đó nhấn nút “Login”:



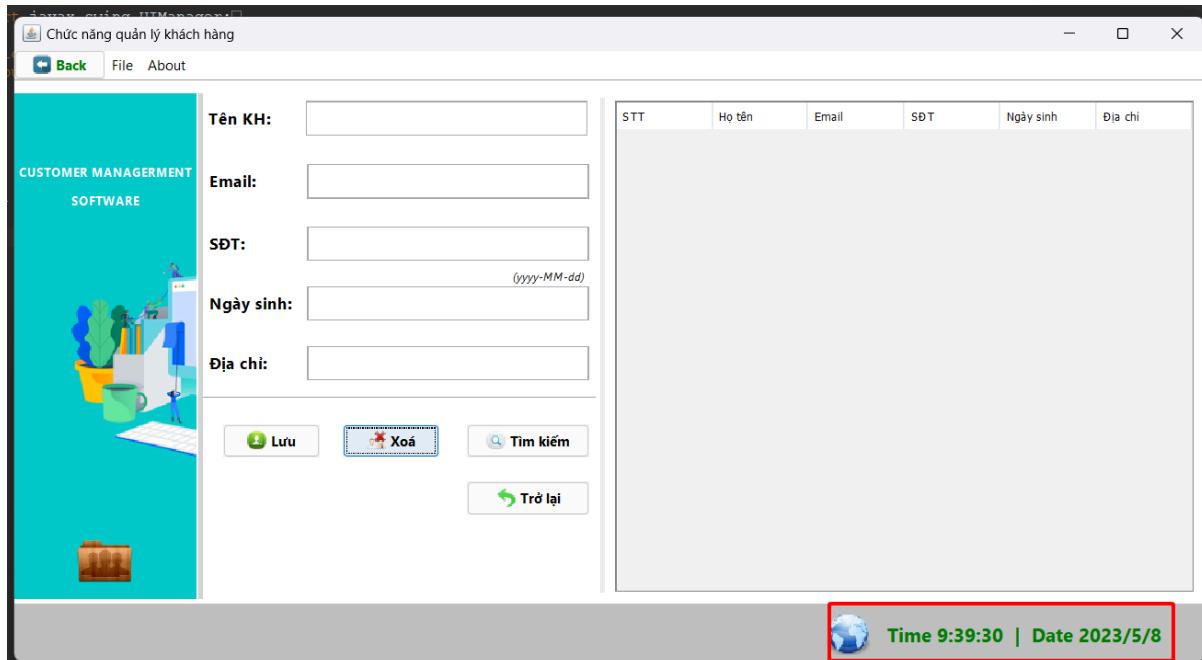
- Lúc này sẽ hiển thị giao diện “Chức năng quản lý khách hàng”:



2. Ngày và giờ trên giao diện

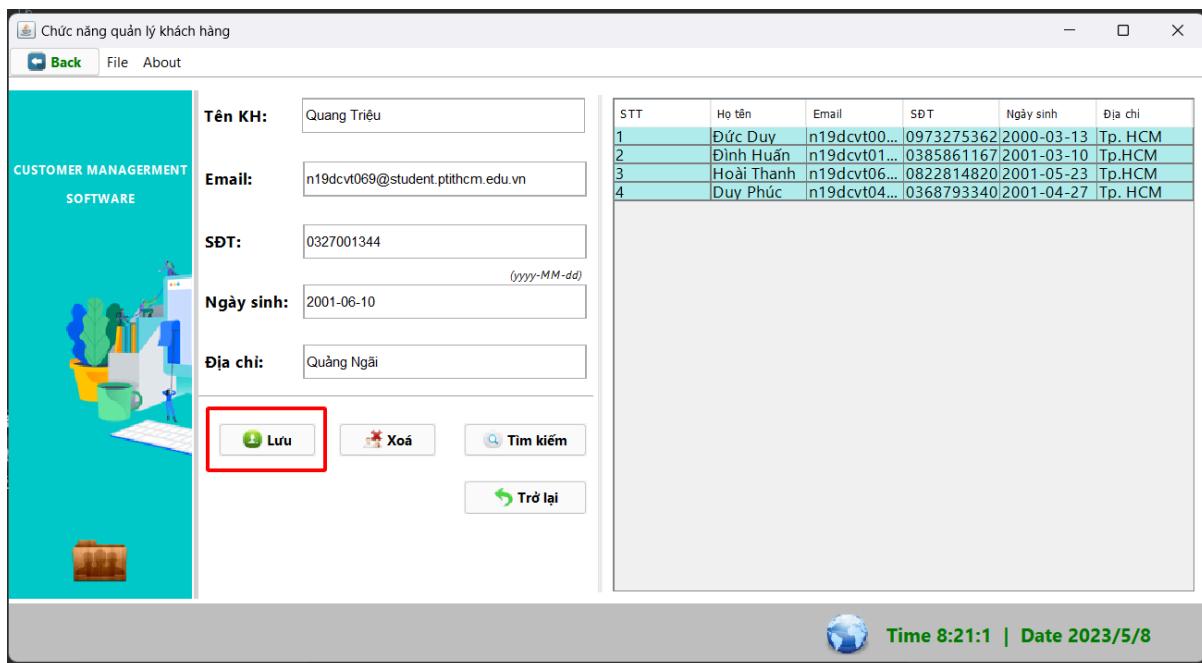
- Tại cuối góc phải giao diện hiển thị Time và Date:

Nhóm 3 - Báo cáo môn học – Quản lý khách hàng



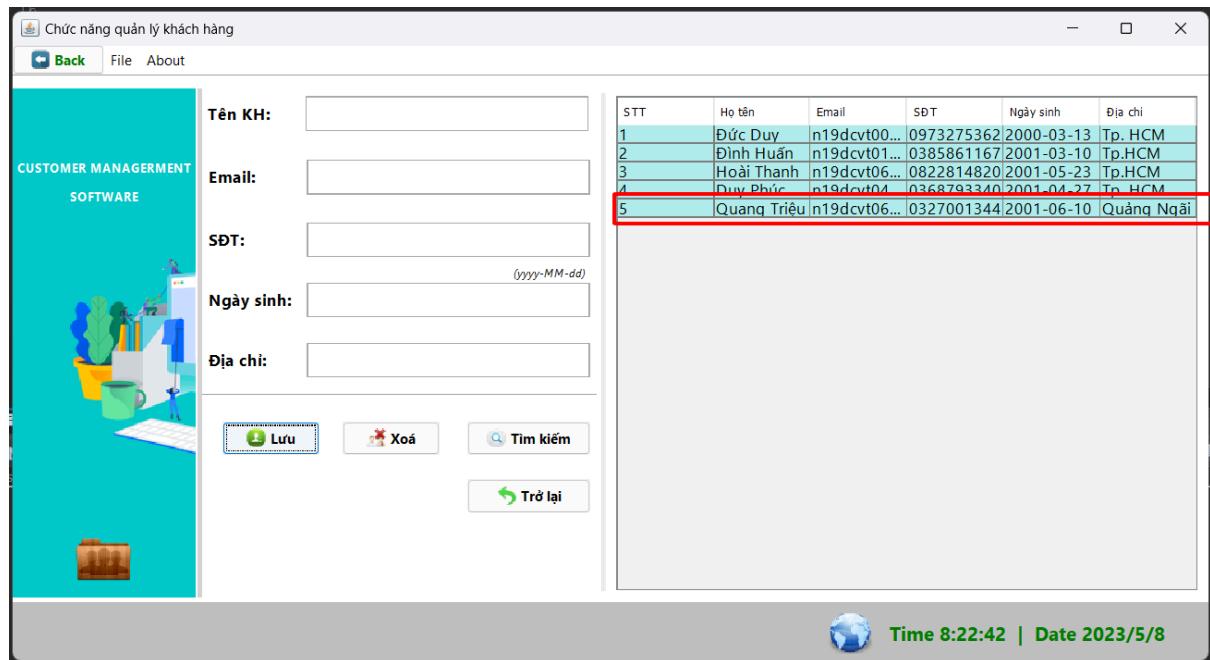
3. Chức năng lưu

- Nhập thông tin khách hàng và bấm nút “Lưu”:



- Thông tin khách hàng sẽ hiển thị trên bảng bên phải giao diện

Nhóm 3 - Báo cáo môn học – Quản lý khách hàng



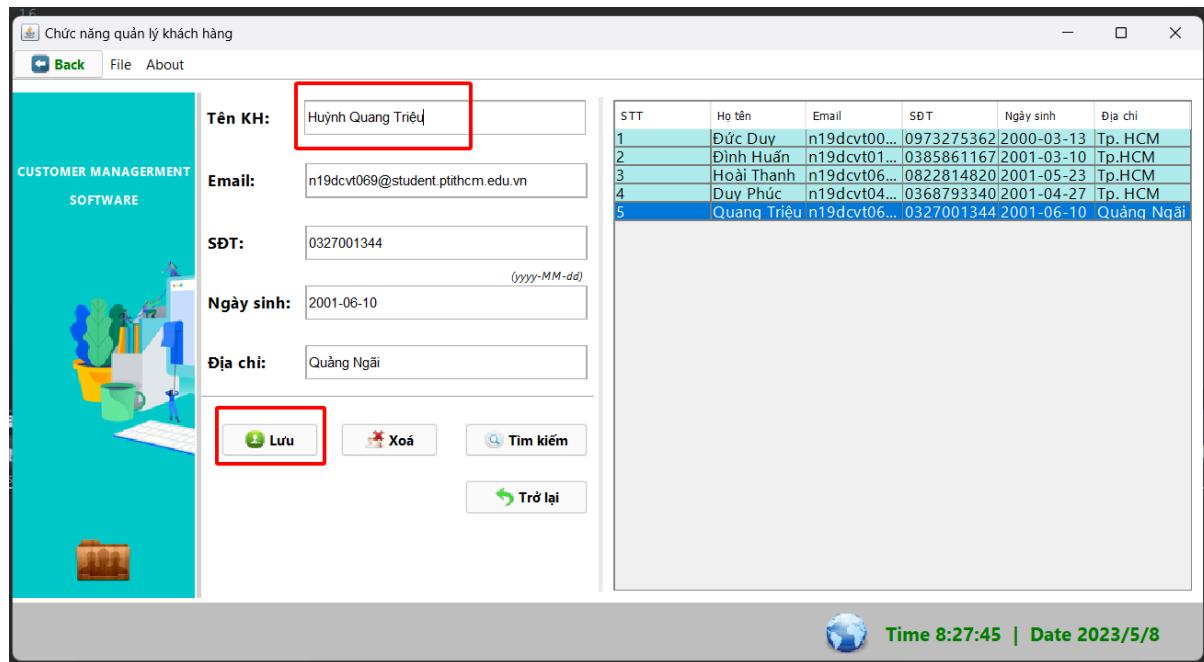
- Thông tin khách hàng cũng sẽ được lưu lại trong MySQL:

#	Time	Action
1	20:31:02	SELECT * FROM customermanager.customer LIMIT 0, 1000

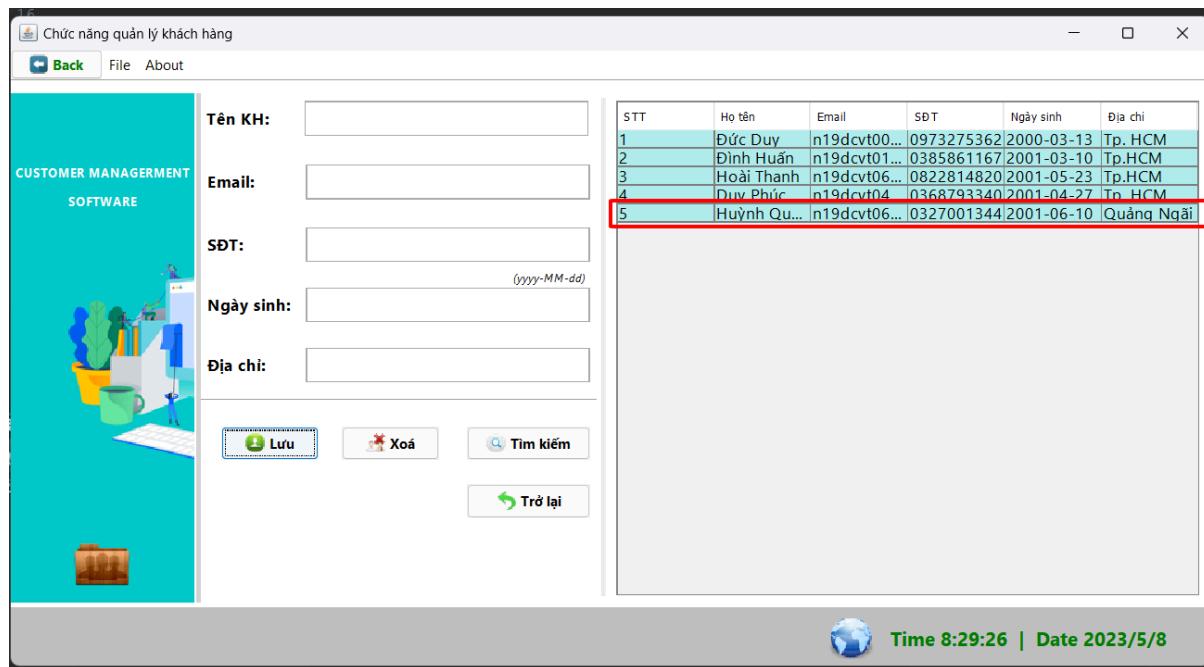
4. Chức năng chỉnh sửa

- Để chỉnh sửa thông tin khách hàng, ta bấm vào khách hàng cần chỉnh sửa, tiến hành chỉnh sửa và nhấn nút “Lưu”:

Nhóm 3 - Báo cáo môn học – Quản lý khách hàng



- Lúc này, thông tin khách hàng sẽ được cập nhật trên bảng bên phải:



- Thông tin khách hàng cũng sẽ được cập nhật trong MySQL:

	id	fullname	email	phone_number	birthday	address
▶	3	Đức Duy	n19dcvt006@student.ptithcm.edu.vn	0973275362	2000-03-13	Tp. HCM
▶	4	Đinh Huân	n19dcvt013@student.ptithcm.edu.vn	0385861167	2001-03-10	Tp.HCM
▶	5	Hoài Thanh	n19dcvt060@student.ptithcm.edu.vn	0822814820	2001-05-23	Tp.HCM
▶	6	Duy Phúc	n19dcvt044@student.ptithcm.edu.vn	0368793340	2001-04-27	Tp. HCM
▶	7	Huỳnh Quang Triệu	n19dcvt069@student.ptithcm.edu.vn	0327001344	2001-06-10	Quảng Ngãi

customer 1 x
Output
Action Output

5. Chức năng tìm kiếm

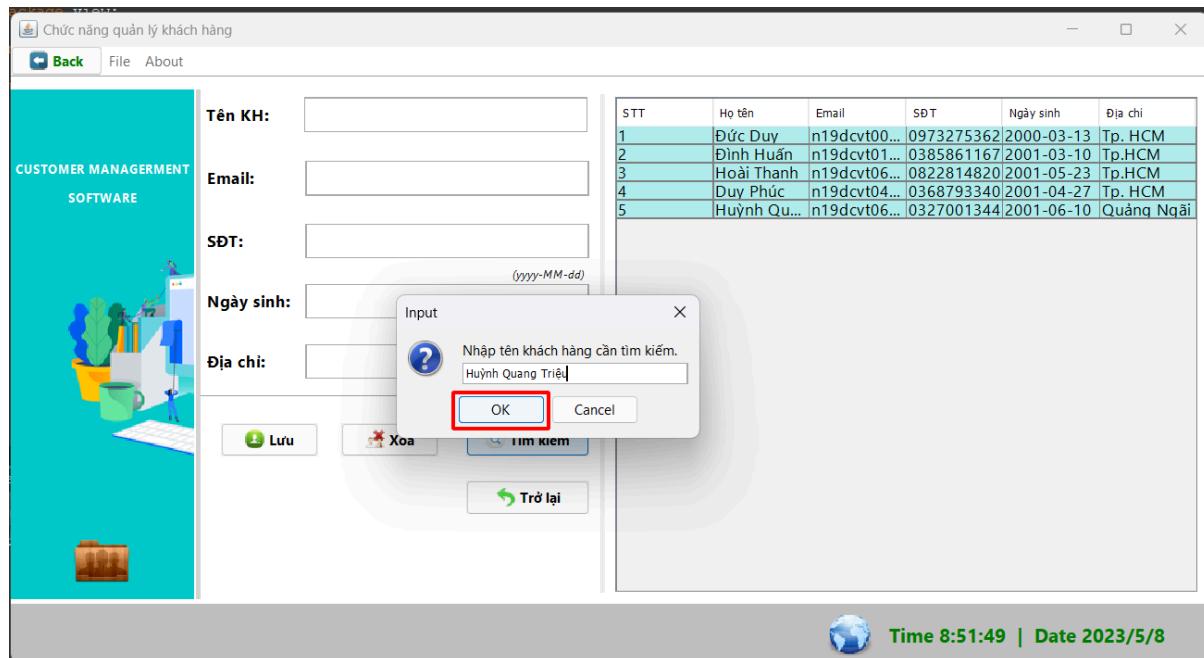
a. Tìm kiếm khách hàng bằng họ tên

- Khi bấm vào nút “Tìm kiếm”, hộp thoại “Input” sẽ hiện ra:

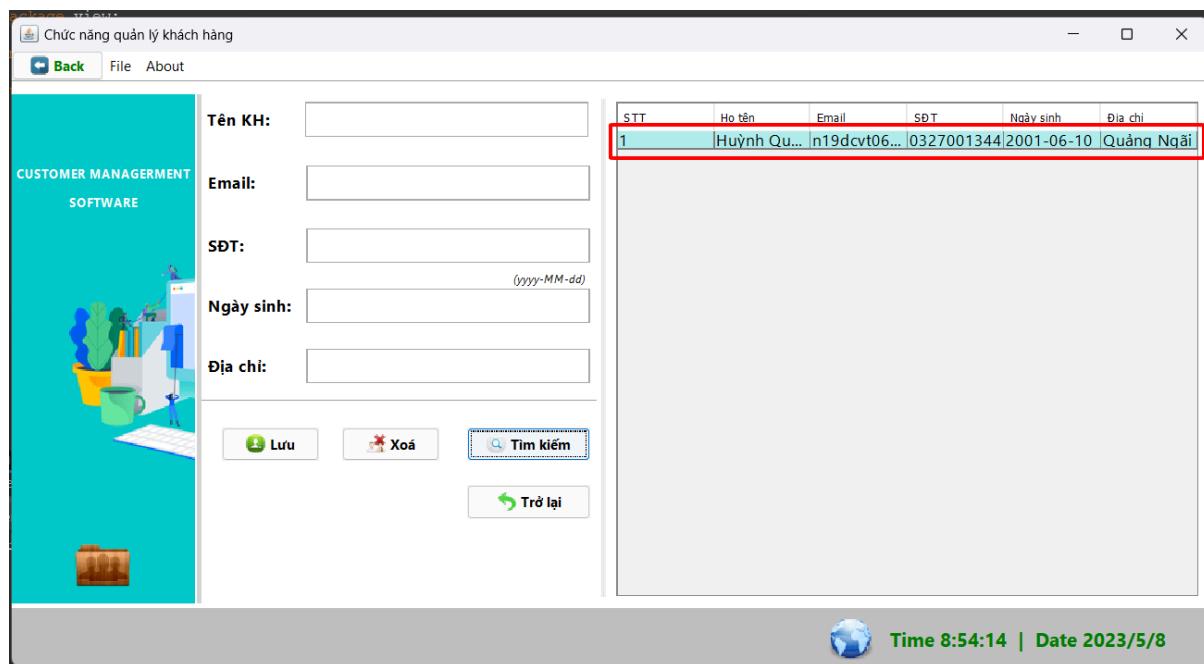
The screenshot shows the software's main window with a sidebar titled 'CUSTOMER MANAGEMENT SOFTWARE'. On the right, there is a table of customer data. A search dialog box is overlaid on the screen, containing a question mark icon, a text input field with placeholder text 'Nhập tên khách hàng cần tìm kiếm.', and two buttons: 'OK' and 'Cancel'. The bottom right corner of the window displays the time and date: 'Time 8:46:17 | Date 2023/5/8'.

- Nhập họ tên khách hàng muốn tìm kiếm rồi bấm nút “OK”:

Nhóm 3 - Báo cáo môn học – Quản lý khách hàng



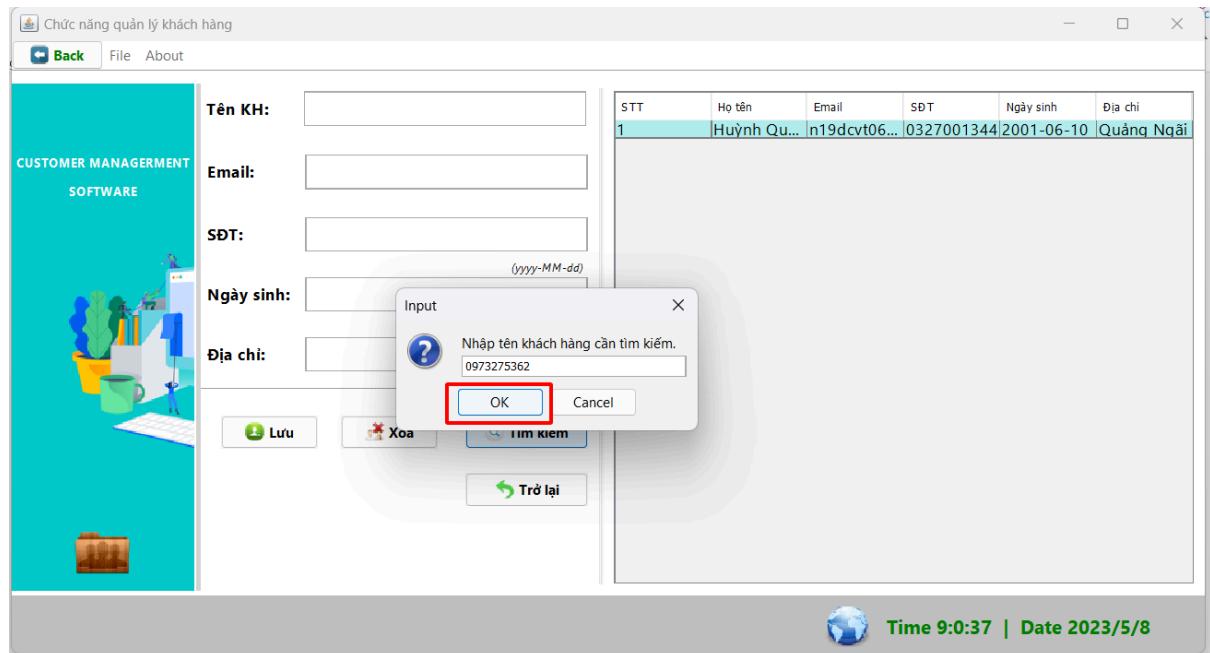
- Lúc này, thông tin khách hàng cần tìm sẽ hiển thị trên bảng bên phải:



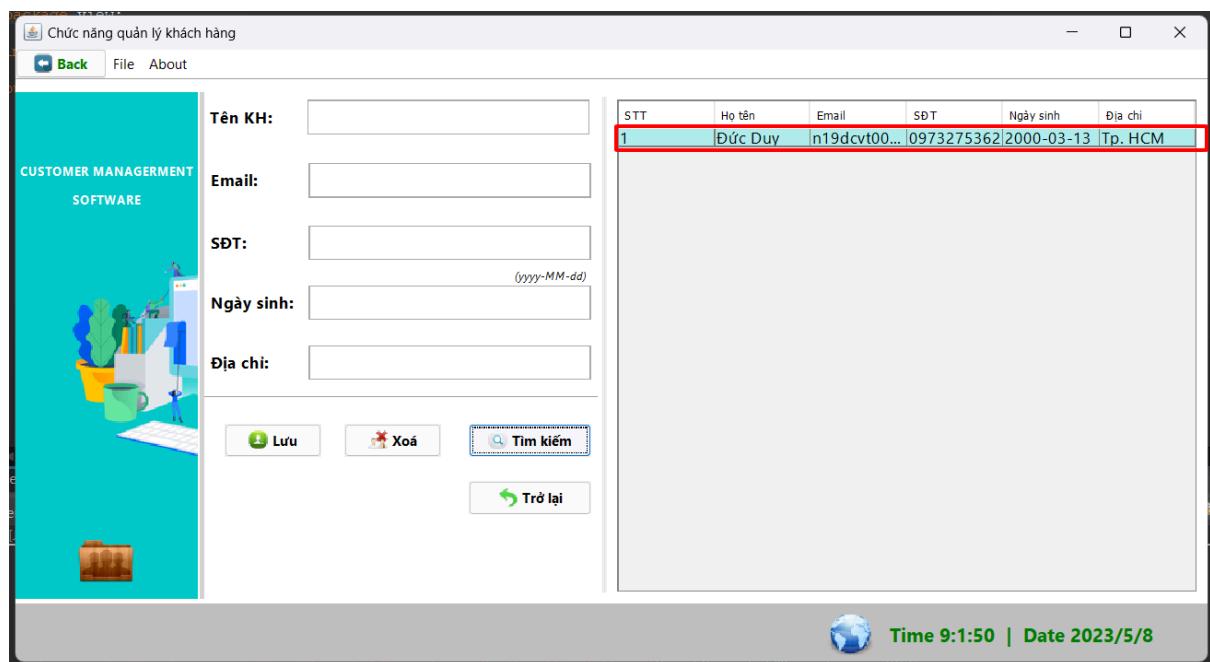
b. Tìm kiếm khách hàng bằng số điện thoại

- Tương tự tìm kiếm bằng họ tên, khi nhấn vào nút “Tìm kiếm” hộp thoại “Input” sẽ hiện ra, nhập số điện thoại khách hàng cần tìm và nhấn “OK”:

Nhóm 3 - Báo cáo môn học – Quản lý khách hàng



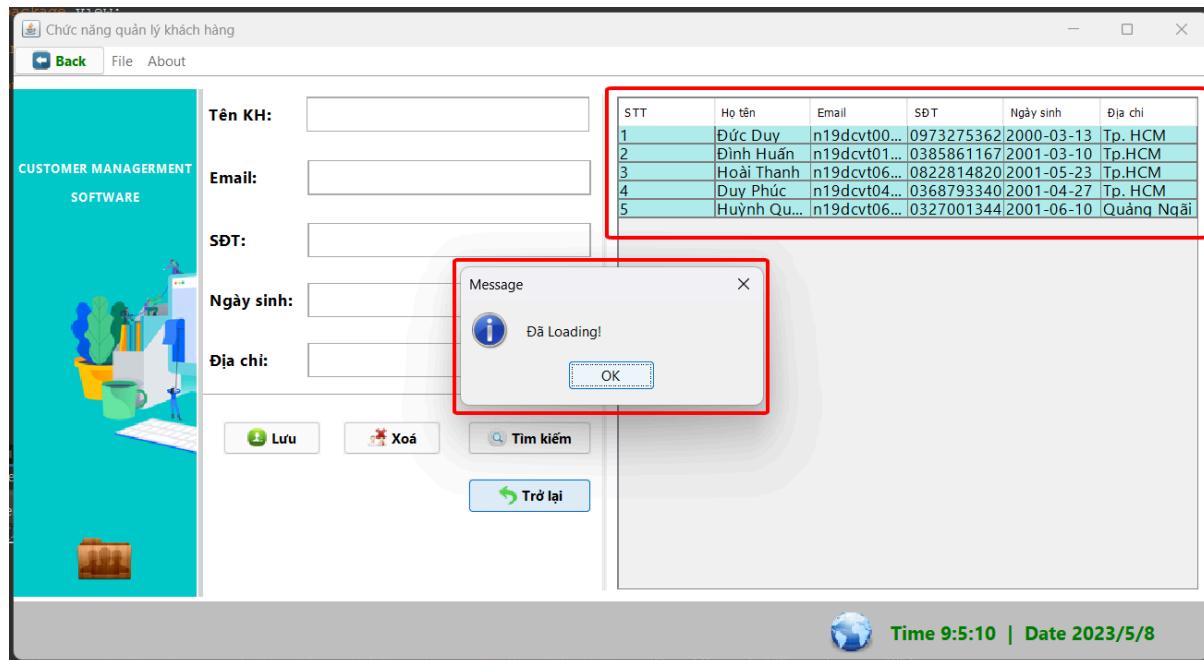
- Lúc này, thông tin khách hàng muốn tìm kiếm bằng số điện thoại sẽ hiện trên bảng:



6. Chức năng hiển thị lại danh sách khách hàng

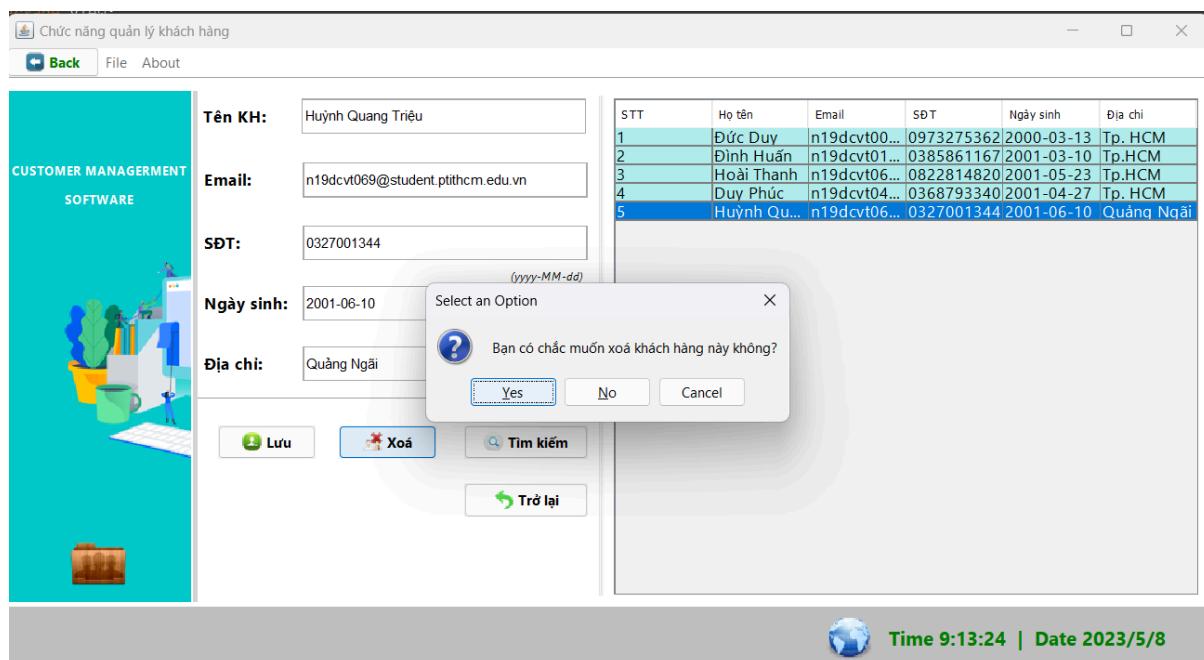
- Để hiển thị lại danh sách khách hàng trên bảng, ta nhấn nút “Trở lại”, lúc này sẽ hiển thị thông báo “Đã Loading!” và danh sách khách hàng sẽ hiển thị trên bảng:

Nhóm 3 - Báo cáo môn học – Quản lý khách hàng



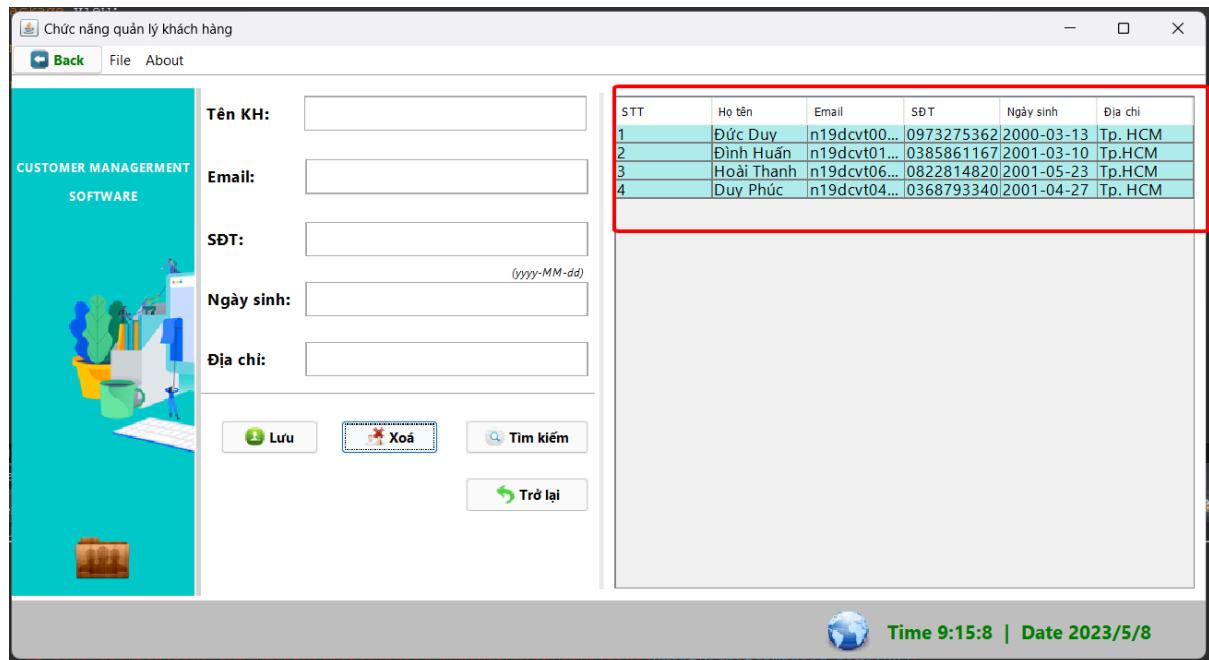
7. Chức năng xoá

- Khi bạn muốn xoá 1 khách hàng, bạn bấm vào khách hàng đó và bấm vào nút “Xoá” sẽ xuất hiện thông báo “Bạn có chắc muốn xoá khách hàng này không”



- Nếu bạn chọn “Yes”, thông tin khách hàng sẽ bị xoá khỏi bảng

Nhóm 3 - Báo cáo môn học – Quản lý khách hàng



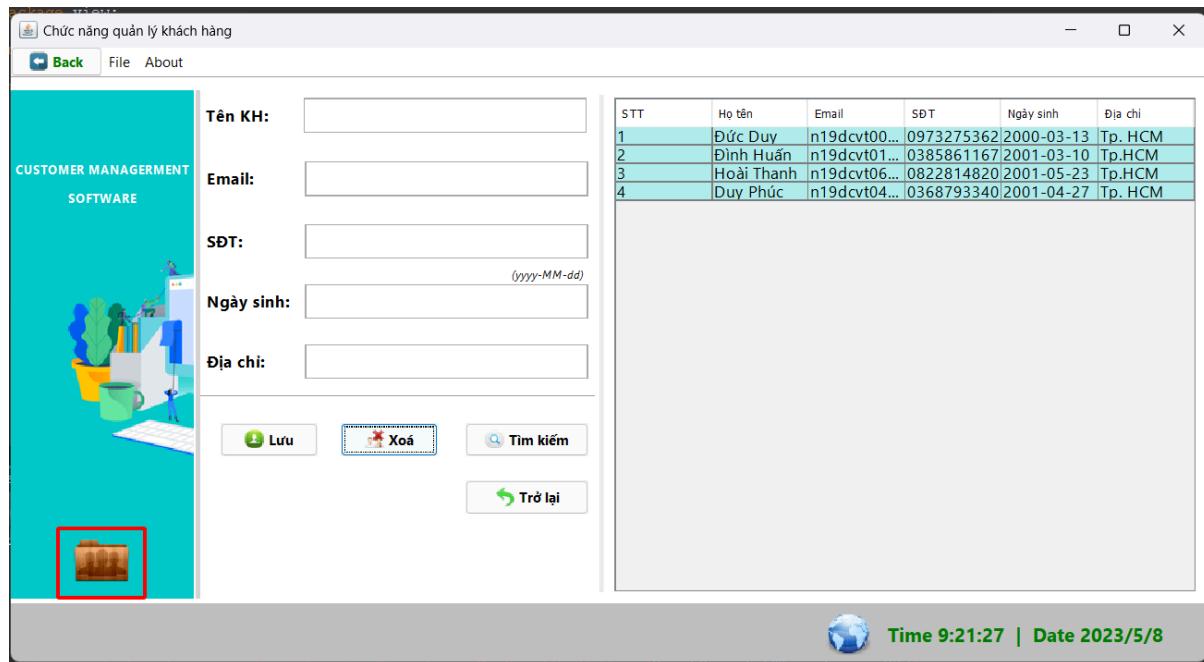
- Thông tin khách hàng cũng sẽ bị xoá khỏi MySQL

	id	fullname	email	phone_number	birthday	address
▶	3	Đức Duy	n19dcvt006@student.ptithcm.edu.vn	0973275362	2000-03-13	Tp. HCM
▶	4	Đình Huân	n19dcvt013@student.ptithcm.edu.vn	0385861167	2001-03-10	Tp.HCM
▶	5	Hoài Thanh	n19dcvt060@student.ptithcm.edu.vn	0822814820	2001-05-23	Tp.HCM
*	6	Duy Phúc	n19dcvt044@student.ptithcm.edu.vn	0368793340	2001-04-27	Tp. HCM
	NULL	NULL	NULL	NULL	NULL	NULL

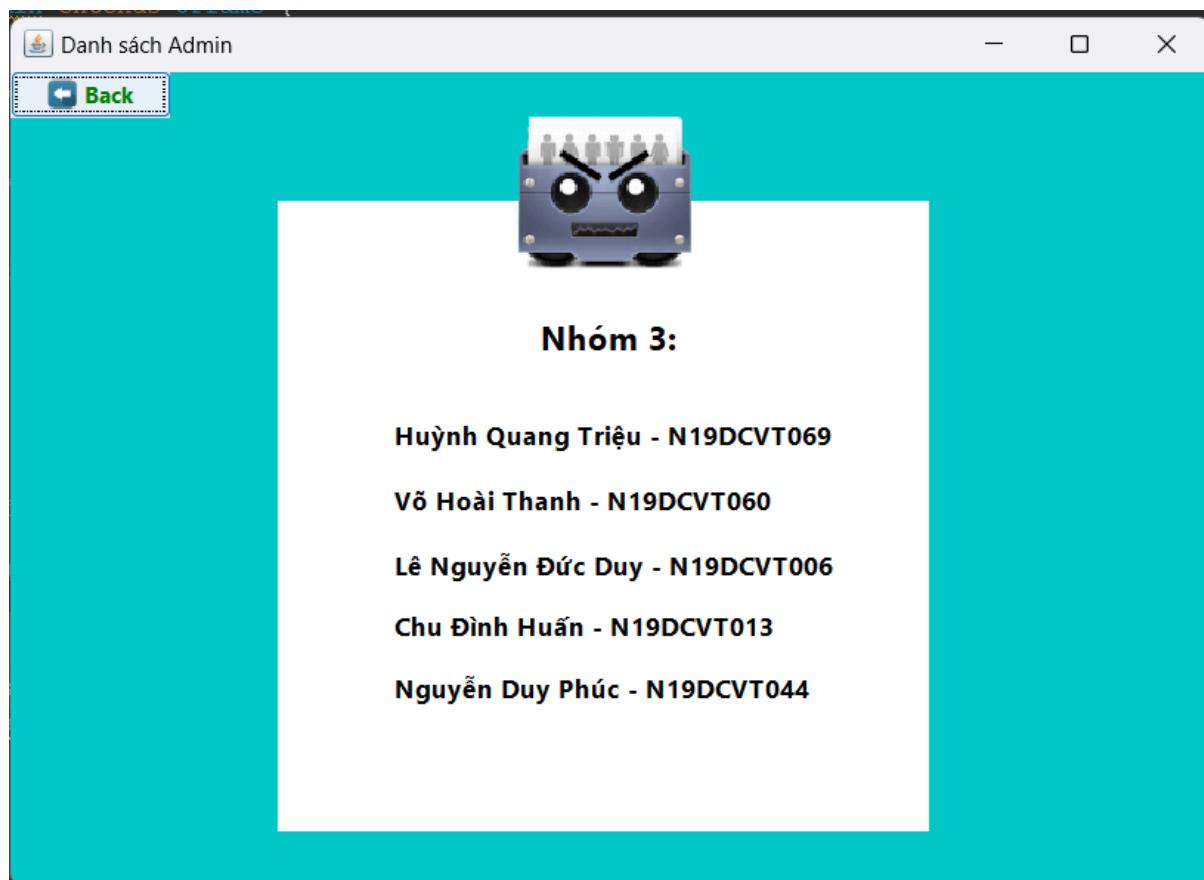
8. Hiển thị giao diện danh sách admin

- Bấm vào biểu tượng phía cuối màn hình bên trái:

Nhóm 3 - Báo cáo môn học – Quản lý khách hàng

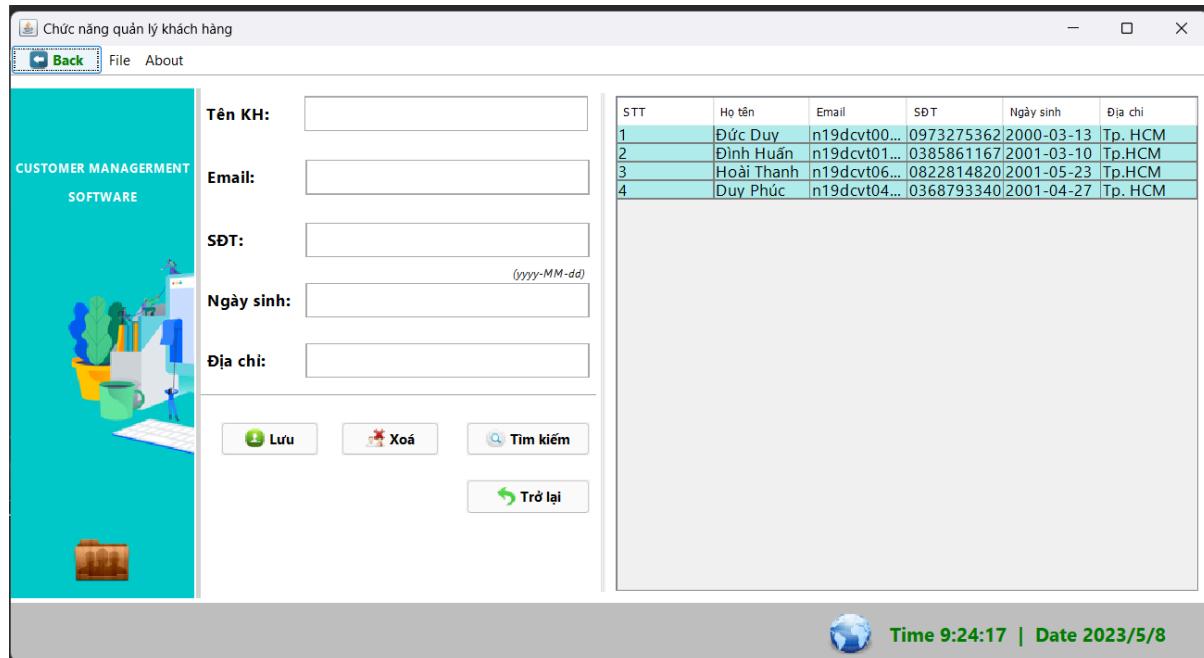


- Lúc này sẽ hiển thị giao diện danh sách những thành viên quản trị:



- Khi bạn bấm vào nút “Back”, sẽ quay lại giao diện “Chức năng quản lý khách hàng”

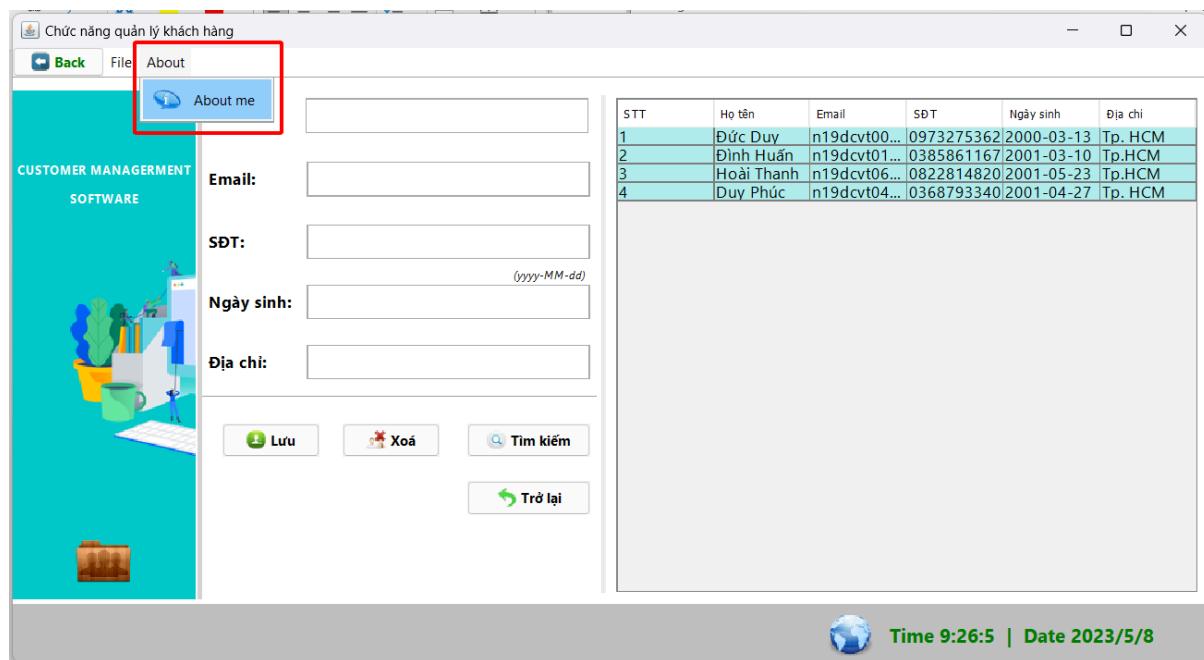
Nhóm 3 - Báo cáo môn học – Quản lý khách hàng



9. Chức năng trên thanh menu

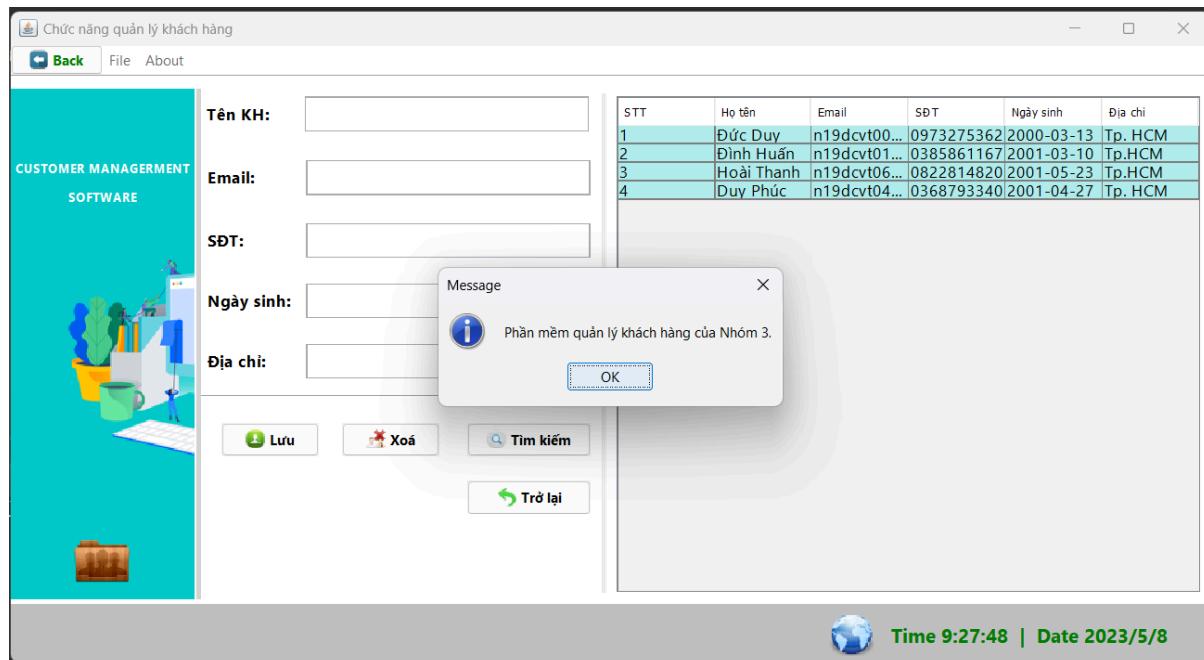
a. Menu About

- Bấm vào “About” sẽ có menu con là “About me”



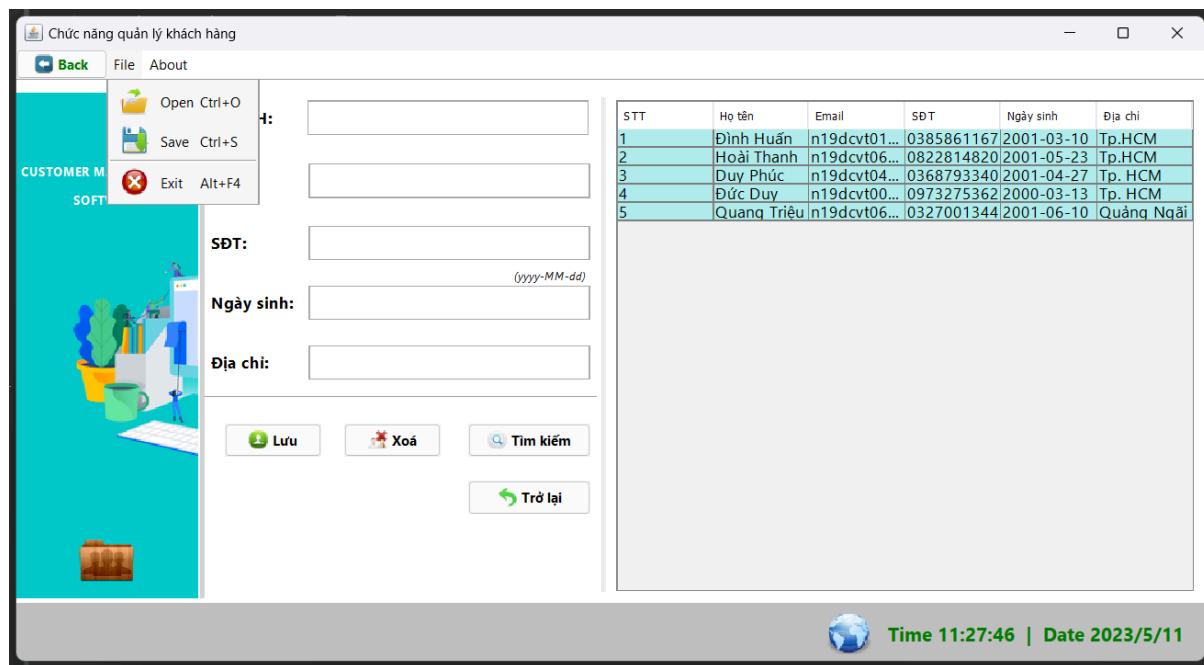
- Khi bấm vào “About me” sẽ hiển thị thông báo thông tin về phần mềm quản lý khách hàng

Nhóm 3 - Báo cáo môn học – Quản lý khách hàng



b. Menu file

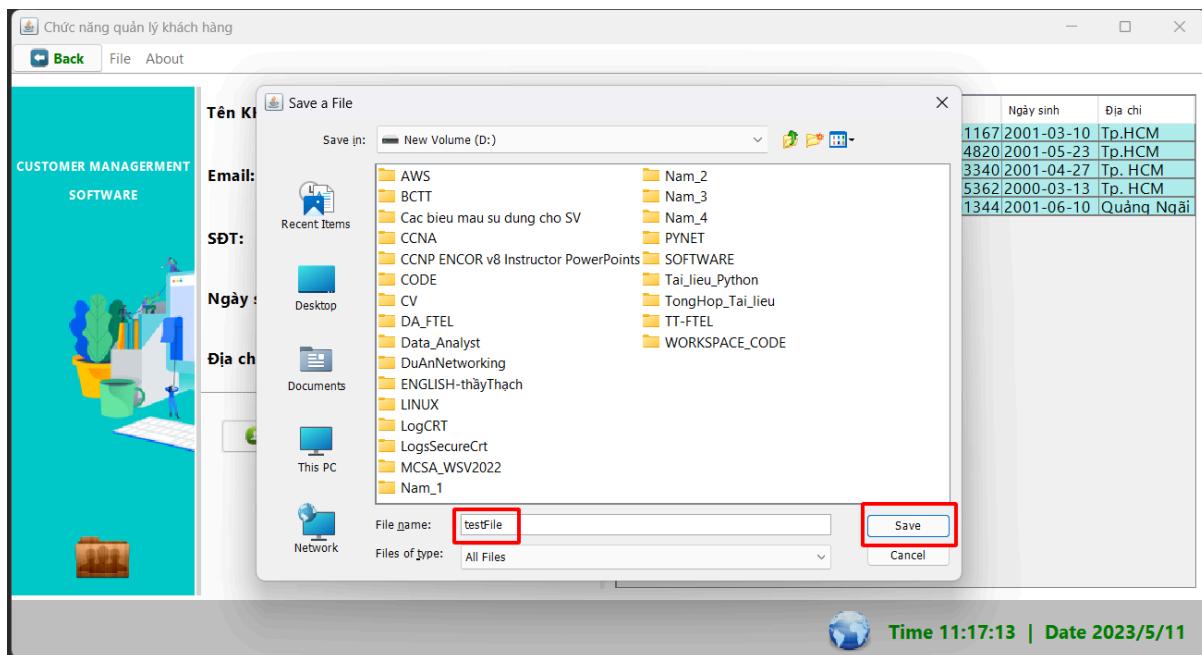
- Khi bấm vào menu file sẽ hiển thị menu con “Open”, “Save”, “Exit”



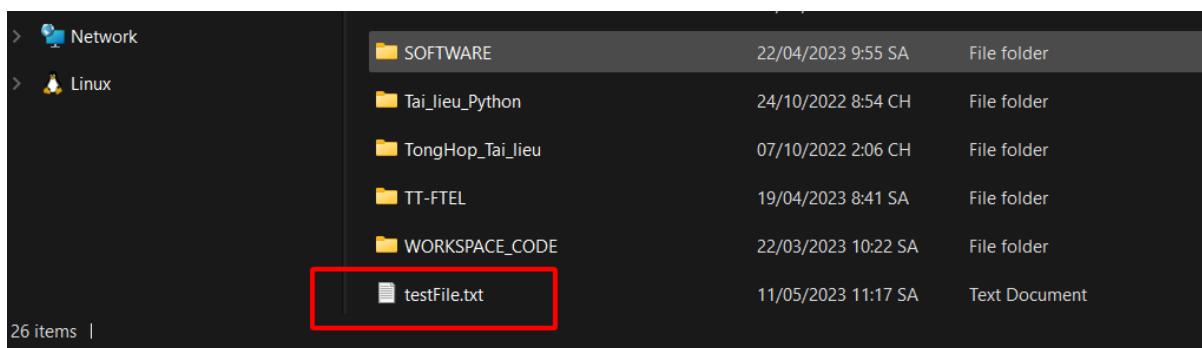
● Save File

- Khi bấm vào “Save” (hoặc dùng tổ hợp phím Ctrl + S) sẽ hiển thị hộp thoại để lưu file danh sách khách hàng, ta đặt tên file và bấm nút “Save”

Nhóm 3 - Báo cáo môn học – Quản lý khách hàng

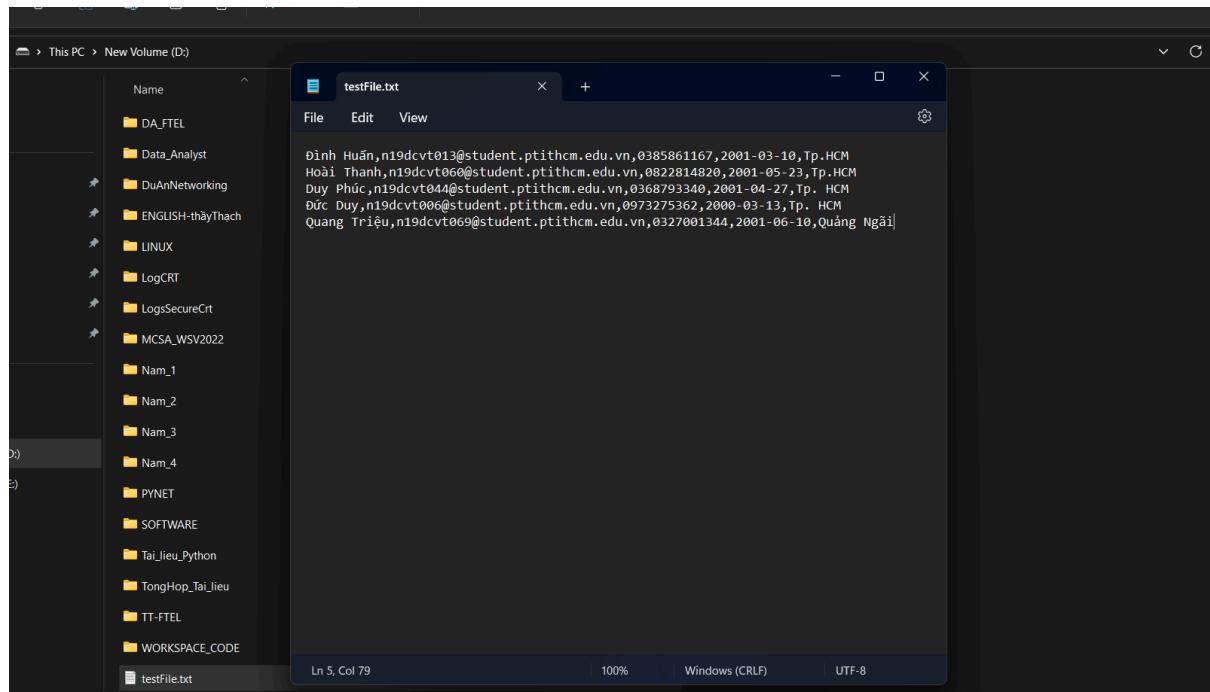


- File “testFile” đã được lưu về máy dưới dạng “.txt”



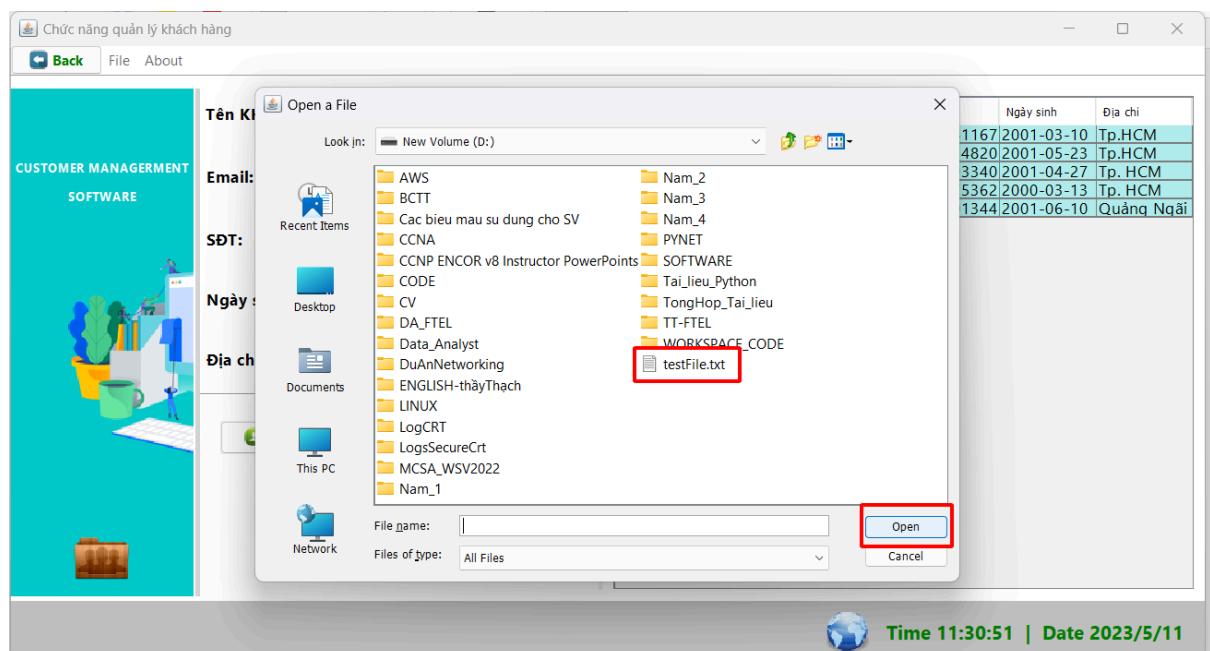
- Mở file “testFile.txt”, sẽ hiển thị danh sách khách hàng mà ta đã save

Nhóm 3 - Báo cáo môn học – Quản lý khách hàng



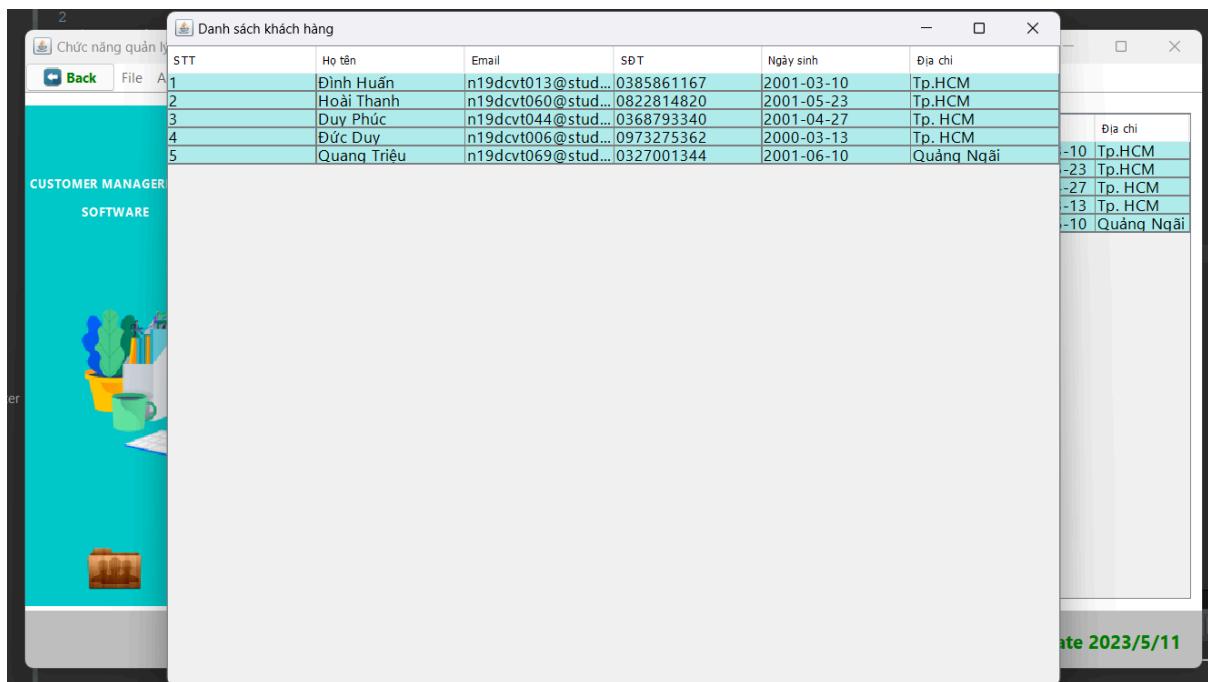
● Open File

- Khi bấm vào “Open” (hoặc dùng tổ hợp phím Ctrl + O) sẽ hiển thị hộp thoại để mở file danh sách khách hàng, sau đó ta chọn vào file muốn mở và bấm open



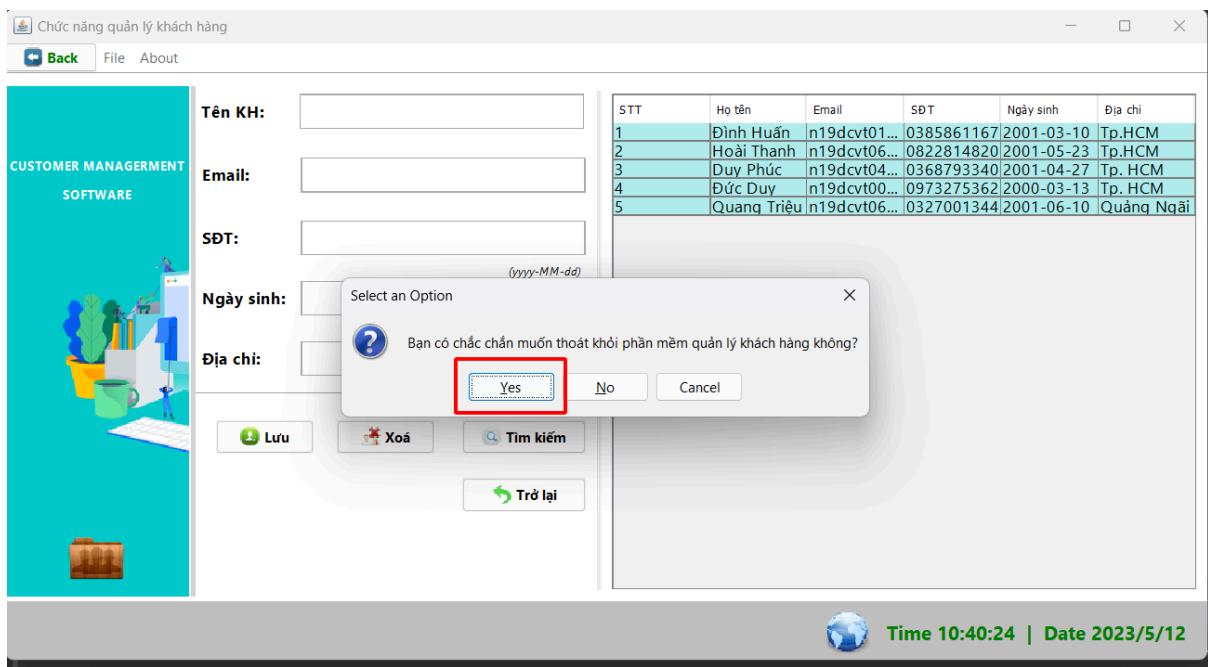
- Một giao diện “Danh sách khách hàng” sẽ hiện ra, hiển thị ra danh sách khách hàng có trong file ta đã open

Nhóm 3 - Báo cáo môn học – Quản lý khách hàng



● Exit

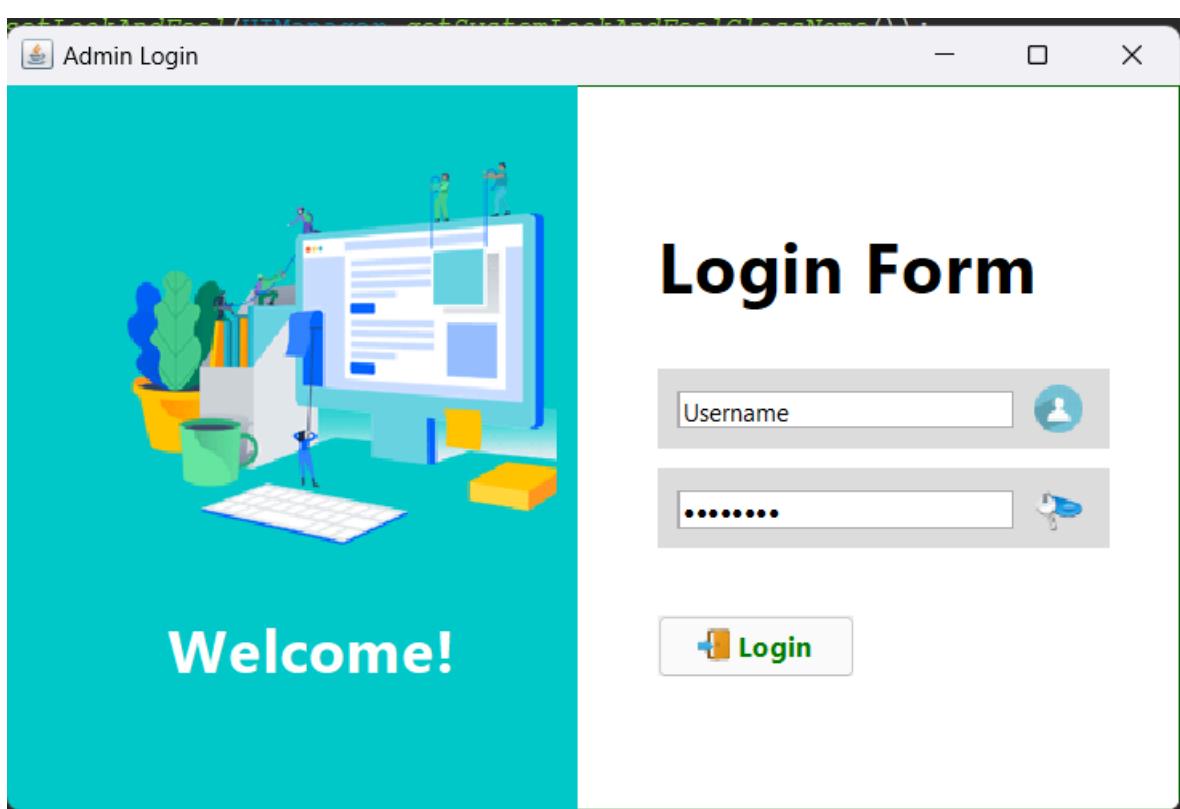
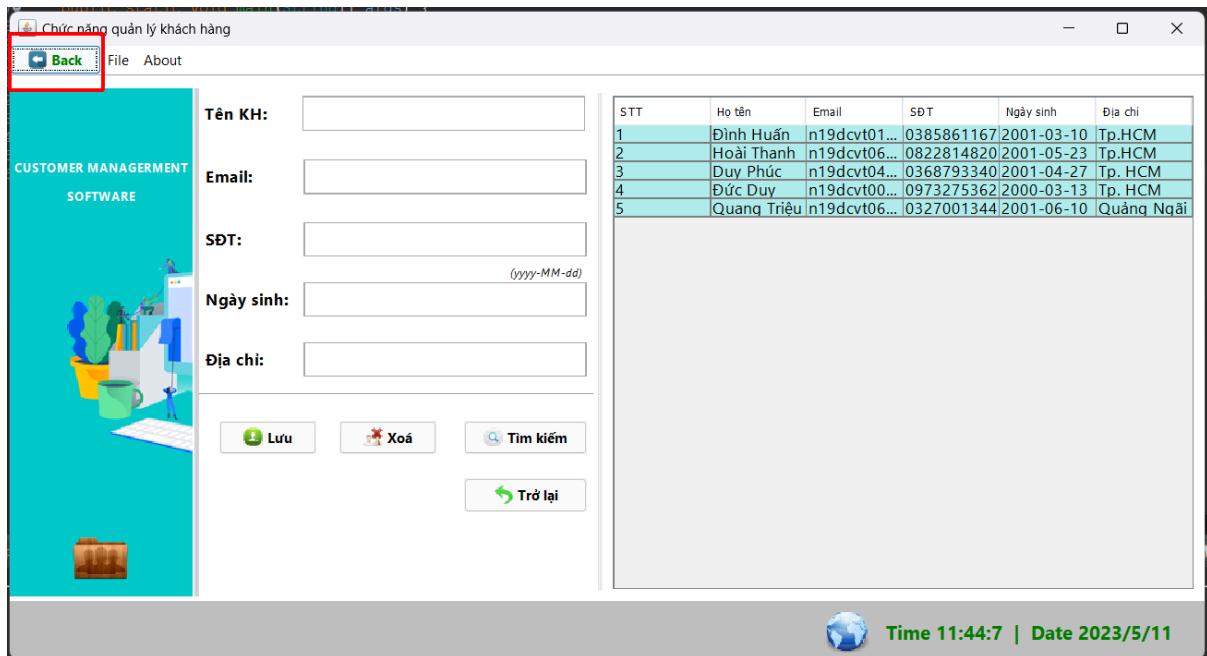
Khi bấm vào “Exit” trên thanh menu (hoặc bấm tổ hợp phím ALT + F4), sẽ hiển thị thông báo nhắc nhở, nếu chọn “Yes” thì phần mềm sẽ đóng lại.



c. Back

Khi bấm vào nút “Back” trên thanh menu, sẽ quay lại giao diện đăng nhập “Admin Login”

Nhóm 3 - Báo cáo môn học – Quản lý khách hàng



PHẦN IV: SOURCE CODE

1. ConnectDatabase.java

```
package database;  
  
public interface ConnectDatabase {
```

```
String DB_URL = "jdbc:mysql://localhost:3306/customermanager";  
String USERNAME = "root";  
String PASSWORD = "10062001";  
}
```

2. Customer.java

```
package model;  
  
import java.io.Serializable;  
  
public class Customer implements Serializable {  
  
    private int id;  
    private String fullname, email, phoneNumber, birthday, address,  
    tenFile;  
  
    public Customer(int id, String fullname, String email, String  
    phoneNumber, String birthday, String address) {  
        this.id = id;  
        this.fullname = fullname;  
        this.email = email;  
        this.phoneNumber = phoneNumber;  
        this.birthday = birthday;  
        this.address = address;  
        this.tenFile = "";  
    }  
  
    public String getTenFile() {  
        return tenFile;  
    }  
    public void setTenFile(String tenFile) {  
        this.tenFile = tenFile;  
    }  
}
```

```
public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getFullscreen() {
    return fullname;
}

public void setFullscreen(String fullname) {
    this.fullname = fullname;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getPhoneNumber() {
    return phoneNumber;
}

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

public String getBirthday() {
    return birthday;
}

public void setBirthday(String birthday) {
```

```
this.birthday = birthday;
}

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

@Override
public String toString() {
    return "Customer [id=" + id + ", fullname=" + fullname + ", email="
+ email + ", phoneNumber=" + phoneNumber
        + ", birthday=" + birthday + ", address=" + address + "]";
}
}
```

3. CustomerModify.java

```
package controller;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import database.ConnectDatabase;

import model.Customer;
```

```
public class CustomerModify {

    public static List<Customer> getCustomerList(String s) {
        List<Customer> dataList = new ArrayList<>();

        Connection conn = null;
        PreparedStatement statement = null;

        try {
            conn = DriverManager.getConnection(ConnectDatabase.DB_URL,
                ConnectDatabase.USERNAME, ConnectDatabase.PASSWORD);

            String sql = "select * from customer where 1 = 1";
            if(s != null && !s.isEmpty()) {
                sql += " and (fullname like ? or phone_number like ?)";
            }
            statement = conn.prepareStatement(sql);
            if(s != null && !s.isEmpty()) {
                statement.setString(1, "%" + s + "%");
                statement.setString(2, "%" + s + "%");
            }
            ResultSet resultSet = statement.executeQuery();

            while(resultSet.next()) {
                Customer customer = new Customer(
                    resultSet.getInt("id"),
                    resultSet.getString("fullname"),
                    resultSet.getString("email"),
                    resultSet.getString("phone_number"),
                    resultSet.getString("birthday"),
                    resultSet.getString("address")
                );
                dataList.add(customer);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

```
}

} catch (SQLException ex) {

Logger.getLogger(CustomerModify.class.getName()).log(Level.SEVERE, null, ex);

} finally {

if(statement != null) {

try {

statement.close();

} catch (SQLException ex) {

Logger.getLogger(CustomerModify.class.getName()).log(Level.SEVERE, null, ex);

}

}

if(conn != null) {

try {

conn.close();

} catch (SQLException ex) {

Logger.getLogger(CustomerModify.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

return dataList;
}

public static void insert(Customer customer) {

Connection conn = null;

PreparedStatement statement = null;

try {

conn = DriverManager.getConnection(ConnectDatabase.DB_URL,
ConnectDatabase.USERNAME, ConnectDatabase.PASSWORD);
```

```
        String sql = "insert into customer(fullname, email,
phone_number, birthday, address) "
                + "values (?, ?, ?, ?, ?);"
        statement = conn.prepareStatement(sql);
        statement.setString(1, customer.getFullscreen());
        statement.setString(2, customer.getEmail());
        statement.setString(3, customer.getPhoneNumber());
        statement.setString(4, customer.getBirthday());
        statement.setString(5, customer.getAddress());
        statement.execute();

    } catch (SQLException ex) {

Logger.getLogger(CustomerModify.class.getName()).log(Level.SEVERE, null, ex);
    } finally {
        if(statement != null) {
            try {
                statement.close();
            } catch (SQLException ex) {

Logger.getLogger(CustomerModify.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
        if(conn != null) {
            try {
                conn.close();
            } catch (SQLException ex) {

Logger.getLogger(CustomerModify.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }
}
```

```
public static void update(Customer customer) {  
    Connection conn = null;  
    PreparedStatement statement = null;  
  
    try {  
        conn = DriverManager.getConnection(ConnectDatabase.DB_URL,  
ConnectDatabase.USERNAME, ConnectDatabase.PASSWORD);  
  
        String sql = "update customer set fullname = ?, email = ?,  
phone_number = ?, birthday = ?, address = ? where id = ?";  
        statement = conn.prepareStatement(sql);  
        statement.setString(1, customer.getFullscreen());  
        statement.setString(2, customer.getEmail());  
        statement.setString(3, customer.getPhoneNumber());  
        statement.setString(4, customer.getBirthday());  
        statement.setString(5, customer.getAddress());  
        statement.setInt(6, customer.getId());  
  
        statement.execute();  
  
    } catch (SQLException ex) {  
        Logger.getLogger(CustomerModify.class.getName()).log(Level.SEVERE, null, ex);  
    } finally {  
        if(statement != null) {  
            try {  
                statement.close();  
            } catch (SQLException ex) {  
                Logger.getLogger(CustomerModify.class.getName()).log(Level.SEVERE, null, ex);  
            }  
        }  
        if(conn != null) {  
    }
```

```
try {
    conn.close();
} catch (SQLException ex) {

Logger.getLogger(CustomerModify.class.getName()).log(Level.SEVERE, null, ex);
}

}

}

}

public static void delete(int id) {
    Connection conn = null;
    PreparedStatement statement = null;

    try {
        conn = DriverManager.getConnection(ConnectDatabase.DB_URL,
ConnectDatabase.USERNAME, ConnectDatabase.PASSWORD);

        String sql = "delete from customer where id = ?";
        statement = conn.prepareStatement(sql);
        statement.setInt(1, id);

        statement.execute();

    } catch (SQLException ex) {

Logger.getLogger(CustomerModify.class.getName()).log(Level.SEVERE, null, ex);
    } finally {
        if(statement != null) {
            try {
                statement.close();
            } catch (SQLException ex) {

Logger.getLogger(CustomerModify.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
}
```

```
        }
    }

    if(conn != null) {
        try {
            conn.close();
        } catch (SQLException ex) {

Logger.getLogger(CustomerModify.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}

}
```

4. Login_Admin.java

```
package view;

import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;

import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.border.LineBorder;
import java.awt.Color;
import javax.swing.JTextField;
import javax.swing.JPasswordField;
import javax.swing.ImageIcon;
import javax.swing.SwingConstantsConstants;
import javax.swing.SwingUtilities;

import java.awt.Font;
```

```
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.UIManager;

public class Login_Admin extends JFrame {

    private JPanel contentPane;
    private JTextField txtUsername;
    private JPasswordField txtPassword;

    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    Login_Admin frame = new Login_Admin();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    public Login_Admin() {
       setFont(new Font("Segoe UI", Font.PLAIN, 14));
       setTitle("Admin Login");

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 600, 400);
        contentPane = new JPanel();
        contentPane.setBackground(new Color(255, 255, 255));
    }
}
```

```
contentPane.setBorder(new LineBorder(new Color(0, 100, 0)));

setContentPane(contentPane);
contentPane.setLayout(null);
contentPane.setLayout(null);

JPanel panel = new JPanel();
panel.setBackground(new Color(0, 204, 204));
panel.setBounds(0, 0, 285, 363);
contentPane.add(panel);
panel.setLayout(null);

JLabel lblIconLogo = new JLabel("");
lblIconLogo.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\login_logo.png"));
lblIconLogo.setBounds(10, 38, 265, 192);
panel.add(lblIconLogo);

JLabel lblNewLabel_3 = new JLabel("Welcome!");
lblNewLabel_3.setForeground(Color.WHITE);
lblNewLabel_3.setBackground(Color.WHITE);
lblNewLabel_3.setFont(new Font("Segoe UI", Font.BOLD, 30));
lblNewLabel_3.setBounds(80, 250, 146, 62);
panel.add(lblNewLabel_3);

JPanel panel_1 = new JPanel();
panel_1.setBackground(new Color(220, 220, 220));
panel_1.setBounds(325, 142, 226, 40);
contentPane.add(panel_1);
panel_1.setLayout(null);

txtUsername = new JTextField();
txtUsername.setFont(new Font("Segoe UI", Font.PLAIN, 12));
txtUsername.setText("Username");
```

```
txtUsername.setBounds(10, 11, 168, 19);
panel_1.add(txtUsername);
txtUsername.setColumns(10);

JLabel lblNewLabel = new JLabel("");
lblNewLabel.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\\\LTrinhHDT\\my_icons\\login_user.png"));
lblNewLabel.setBounds(188, 0, 34, 40);
panel_1.add(lblNewLabel);

JPanel panel_1_1 = new JPanel();
panel_1_1.setBackground(new Color(220, 220, 220));
panel_1_1.setBounds(325, 192, 226, 40);
contentPane.add(panel_1_1);
panel_1_1.setLayout(null);

txtPassword = new JPasswordField();
txtPassword.setFont(new Font("Segoe UI", Font.ITALIC, 10));
txtPassword.setText("Password");
txtPassword.setBounds(10, 11, 168, 19);
panel_1_1.add(txtPassword);

JLabel lblNewLabel_1 = new JLabel("");
lblNewLabel_1.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\\\LTrinhHDT\\my_icons\\login_pass.png"));
lblNewLabel_1.setBounds(188, 0, 34, 40);
panel_1_1.add(lblNewLabel_1);

JLabel lblNewLabel_2 = new JLabel("Login Form");
lblNewLabel_2.setFont(new Font("Segoe UI", Font.BOLD, 35));
lblNewLabel_2.setBounds(325, 63, 199, 53);
contentPane.add(lblNewLabel_2);

JButton btnLogin = new JButton("Login");
```

```
btnLogin.setIcon(new  
ImageIcon("D:\\\\Nam_4\\\\HK2\\\\LTrinhHDT\\\\my_icons\\\\login.png"));  
btnLogin.setBackground(UIManager.getColor("Button.background"));  
btnLogin.setForeground(new Color(0, 128, 0));  
btnLogin.setFont(new Font("Segoe UI", Font.BOLD, 14));  
btnLogin.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        String username = txtUsername.getText();  
        String password = txtPassword.getPassword().toString();  
  
        //login  
        if (username.contains("Quang Trieu") ||  
password.contains("1234567890")) {  
            txtUsername.setText(null);  
            txtPassword.setText(null);  
  
            //link  
            CustomerManagement.main(null);  
        }  
        else if (username.contains("Admin") ||  
password.contains("1234567890")){  
            txtUsername.setText(null);  
            txtPassword.setText(null);  
  
            // dong form hien tai  
            JFrame currentFrame = (JFrame)  
SwingUtilities.getWindowAncestor(btnLogin);  
            currentFrame.dispose();  
  
            //link  
            CustomerManagement.main(null);  
        }  
        else {  
    }  
    }  
});
```

```
        JOptionPane.showMessageDialog(null, "Invalid Login Details", "Login Error", JOptionPane.ERROR_MESSAGE);

        txtUsername.setText(null);
        txtPassword.setText(null);
    }

}

});

btnLogin.setBounds(325, 265, 98, 31);
contentPane.add(btnLogin);

setLocationRelativeTo(null);

this.setVisible(true);
}

}
```

5. Ds_Admin.java

```
package view;

import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.SwingUtilities;
import javax.swing.border.EmptyBorder;
import java.awt.Color;
import javax.swing.JLabel;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import java.awt.Font;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
```

```
public class Ds_Admin extends JFrame {  
  
    private JPanel contentPane;  
    public static void main(String[] args) {  
        EventQueue.invokeLater(new Runnable() {  
            public void run() {  
                try {  
                    Ds_Admin frame = new Ds_Admin();  
                    frame.setVisible(true);  
                } catch (Exception e) {  
                    e.printStackTrace();  
                }  
            }  
        });  
    }  
  
    /**  
     * Create the frame.  
     */  
    public Ds_Admin() {  
        setTitle("Danh sách Admin");  
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        setBounds(100, 100, 665, 478);  
        contentPane = new JPanel();  
        contentPane.setBackground(new Color(0, 204, 204));  
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));  
  
        setContentPane(contentPane);  
        contentPane.setLayout(null);  
  
        JLabel lblNewLabel = new JLabel("");  
        lblNewLabel.setBounds(274, 10, 101, 108);  
        contentPane.add(lblNewLabel);  
    }  
}
```

```
lblNewLabel.setIcon(new  
ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\ds_admin.png"));  
  
JPanel panel = new JPanel();  
panel.setBackground(new Color(255, 255, 255));  
panel.setBounds(145, 70, 353, 341);  
contentPane.add(panel);  
panel.setLayout(null);  
  
JLabel lblNewLabel_1 = new JLabel("Huỳnh Quang Triệu -  
N19DCVT069");  
lblNewLabel_1.setFont(new Font("Segoe UI", Font.BOLD, 14));  
lblNewLabel_1.setBounds(63, 112, 246, 29);  
panel.add(lblNewLabel_1);  
  
JLabel lblNewLabel_1_1 = new JLabel("Võ Hoài Thành - N19DCVT060");  
lblNewLabel_1_1.setFont(new Font("Segoe UI", Font.BOLD, 14));  
lblNewLabel_1_1.setBounds(63, 147, 246, 29);  
panel.add(lblNewLabel_1_1);  
  
JLabel lblNewLabel_1_2 = new JLabel("Lê Nguyễn Đức Duy -  
N19DCVT006");  
lblNewLabel_1_2.setFont(new Font("Segoe UI", Font.BOLD, 14));  
lblNewLabel_1_2.setBounds(63, 182, 246, 29);  
panel.add(lblNewLabel_1_2);  
  
JLabel lblNewLabel_1_3 = new JLabel("Chu Đình Huân - N19DCVT013");  
lblNewLabel_1_3.setFont(new Font("Segoe UI", Font.BOLD, 14));  
lblNewLabel_1_3.setBounds(63, 215, 246, 29);  
panel.add(lblNewLabel_1_3);  
  
JLabel lblNewLabel_1_4 = new JLabel("Nguyễn Duy Phúc -  
N19DCVT044");  
lblNewLabel_1_4.setFont(new Font("Segoe UI", Font.BOLD, 14));
```

```
lblNewLabel_1_4.setBounds(63, 249, 246, 29);
panel.add(lblNewLabel_1_4);

JLabel lblNewLabel_2 = new JLabel("Nhóm 3:");
lblNewLabel_2.setFont(new Font("Segoe UI", Font.BOLD, 18));
lblNewLabel_2.setBounds(142, 58, 88, 30);
panel.add(lblNewLabel_2);

JButton btnBack = new JButton("Back");
btnBack.setForeground(new Color(0, 128, 0));
btnBack.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        CustomerManagement.main(null);
        JFrame currentFrame = (JFrame)
SwingUtilities.getWindowAncestor(btnBack);
        currentFrame.dispose();
    }
});
btnBack.setIcon(new
ImageIcon("D:\\\\Nam_4\\\\HK2\\\\LTrinhHDT\\\\my_icons\\\\back.png"));
btnBack.setFont(new Font("Segoe UI", Font.BOLD, 12));
btnBack.setBounds(0, 0, 87, 25);
contentPane.add(btnBack);

setLocationRelativeTo(null);
}
}
```

6. CustomerManagement.java

```
package view;

import java.awt.EventQueue;
```

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JPopupMenu;
import javax.swing.SwingUtilities;
import javax.swing.border.EmptyBorder;

import java.awt.BorderLayout;
import java.awt.Color;
import javax.swing.JLabel;
import javax.swing.JOptionPane;

import java.awt.Font;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.Iterator;
import java.util.List;
import java.util.Scanner;

import javax.security.auth.Refreshable;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFileChooser;

import java.awt.event.ActionListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.ActionEvent;
import javax.swing.JToolBar;
import javax.swing.JScrollBar;
import javax.swing.JSeparator;
import javax.swing.JTextField;
import javax.swing.JSpinner;
```

```
import javax.swing.JSlider;
import javax.swing.JTabbedPane;
import javax.swing.JTable;
import java.awt.ScrollPane;
import javax.swing.JScrollPane;
import javax.swing.table.DefaultTableModel;

import model.Customer;
import controller.CustomerModify;
import javax.swing.JMenuBar;
import javax.swing.JMenu;
import javax.swing.JMenuItem;
import javax.swing.UIManager;
import javax.swing.KeyStroke;
import java.awt.event.KeyEvent;
import java.awt.event.InputEvent;
import java.io.*;
import java.text.SimpleDateFormat;
import java.awt.event.MouseAdapter;

public class CustomerManagement extends JFrame {
    private Customer model;
    private JPanel contentPane;
    private JLabel lblClock;
    private JTextField emailTxt;
    private JTextField fullnameTxt;
    private JTextField phoneTxt;
    private JTextField birthdayTxt;
    private JTextField addressTxt;
    private JTable customerTable;

    //SQL
    DefaultTableModel tableModel;
```

```
List<Customer> dataList;
int currentPos = -1;

public static void main(String[] args) {

    EventQueue.invokeLater(new Runnable() {
        public void run() {
            try {
                CustomerManagement frame = new CustomerManagement();
                frame.setVisible(true);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    });
}

// set clock
public void clock() {
    Thread clock = new Thread() {
        public void run() {
            try {
                for(;;) {
                    Calendar cal = new GregorianCalendar();
                    int day = cal.get(Calendar.DAY_OF_MONTH);
                    int month = cal.get(Calendar.MONTH) + 1;
                    int year = cal.get(Calendar.YEAR);

                    int second = cal.get(Calendar.SECOND);
                    int minute = cal.get(Calendar.MINUTE);
                    int hour = cal.get(Calendar.HOUR);
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    };
}
```

```
        lblClock.setText("Time " + hour + ":" + minute + ":" +  
second + " | " + " Date " + year + "/" + month + "/" + day);  
  
        sleep(1000);  
    }  
}  
} catch (InterruptedException e) {  
// TODO Auto-generated catch block  
e.printStackTrace();  
}  
}  
};  
  
clock.start();  
}  
  
public CustomerManagement() {  
    this.model = new Customer(currentPos, getName(), getName(),  
getTitle(), getWarningString(), getName());  
    initComponents();  
    clock();  
  
    //SQL  
    tableModel = (DefaultTableModel) customerTable.getModel();  
    dataList = CustomerModify.getCustomerList(null);  
  
    showData();  
  
    customerTable.addMouseListener(new MouseListener() {  
        public void mouseReleased(MouseEvent e) {  
  
        }  
        public void mousePressed(MouseEvent e) {  
  
        }  
    });  
}
```

```
public void mouseExited(MouseEvent e) {  
}  
public void mouseEntered(MouseEvent e) {  
}  
  
}  
  
public void mouseClicked(MouseEvent e) {  
    // TODO Auto-generated method stub  
    currentPos = customerTable.getSelectedRow();  
  
fullnameTxt.setText(dataList.get(currentPos).getFullname());  
emailTxt.setText(dataList.get(currentPos).getEmail());  
  
phoneTxt.setText(dataList.get(currentPos).getPhoneNumber());  
  
birthdayTxt.setText(dataList.get(currentPos).getBirthday());  
addressTxt.setText(dataList.get(currentPos).getAddress());  
  
}  
});  
}  
  
//ham hien thi du lieu len bang  
public void showData() {  
    tableModel.setRowCount(0);  
    for (Customer customer : dataList) {  
        tableModel.addRow(new Object[] {  
            tableModel.getRowCount() + 1,  
            customer.getFullname(),  
            customer.getEmail(),  
            customer.getPhoneNumber(),  
            customer.getBirthday(),  
            customer.getAddress(),  
            customer.getPhoneType()  
        });  
    }  
}
```

```
        customer.getAddress()

    });
}

}

public void initComponents( ) {
    setTitle("Chức năng quản lý khách hàng");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 1050, 569);

    JMenuBar menuBar = new JMenuBar();
    setJMenuBar(menuBar);

    JButton BackBtn = new JButton("Back");
    BackBtn.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\back.png"));
    menuBar.add(BackBtn);
    BackBtn.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            // Đóng form hiện tại
            JFrame currentFrame = (JFrame)
            SwingUtilities.getWindowAncestor(BackBtn);
            currentFrame.dispose();

            // Mở form mới
            Login_Admin.main(null);
        }
    });
    BackBtn.setFont(new Font("Segoe UI", Font.BOLD, 12));
    BackBtn.setForeground(new Color(0, 128, 0));
    BackBtn.setBackground(UIManager.getColor("Button.background"));

    JMenu menuFile = new JMenu("File");
```

```
menuBar.add(menuFile);

JMenuItem menuOpen = new JMenuItem("Open");
menuOpen.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_O,
InputEvent.CTRL_DOWN_MASK));
menuOpen.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\Open.png"));
menuFile.add(menuOpen);
menuOpen.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        thucHienOpenFile();
    }
});

JMenuItem menuSave = new JMenuItem("Save");
menuSave.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_S,
InputEvent.CTRL_DOWN_MASK));
menuSave.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\Save.png"));
menuFile.add(menuSave);
menuSave.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        thucHienSaveFile();
    }
});

JSeparator separator = new JSeparator();
menuFile.add(separator);

JMenuItem menuExit = new JMenuItem("Exit");
menuExit.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_F4,
InputEvent.ALT_DOWN_MASK));
menuExit.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\Exit.png"));
menuFile.add(menuExit);
```

```
menuExit.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        thoatKhoiPhanMem();
    }
});

JMenu menuAbout = new JMenu("About");
menuBar.add(menuAbout);

JMenuItem menuAboutMe = new JMenuItem("About me");
menuAboutMe.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\About.png"));
menuAbout.add(menuAboutMe);
menuAboutMe.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        hienThiAbout();
    }
});
contentPane = new JPanel();
contentPane.setBackground(new Color(255, 255, 255));
contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);
contentPane.setLayout(null);

JPanel panel = new JPanel();
panel.setBackground(new Color(192, 192, 192));
panel.setBounds(0, 454, 1036, 55);
contentPane.add(panel);
panel.setLayout(null);

lblClock = new JLabel("Clock");
lblClock.setForeground(new Color(0, 128, 0));
```

```
lblClock.setBackground(Color.WHITE);
lblClock.setFont(new Font("Segoe UI", Font.BOLD, 17));
lblClock.setBounds(759, 10, 267, 35);
panel.add(lblClock);

JLabel lblTraiDat = new JLabel("");
lblTraiDat.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\clock.png"));
lblTraiDat.setBounds(701, 10, 48, 35);
panel.add(lblTraiDat);

JPanel panel_1 = new JPanel();
panel_1.setBackground(new Color(0, 204, 204));
panel_1.setBounds(0, 10, 159, 440);
contentPane.add(panel_1);
panel_1.setLayout(null);

JLabel lblNewLabel_1 = new JLabel("");
lblNewLabel_1.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\login_logo.png"));
lblNewLabel_1.setBounds(0, 10, 159, 420);
panel_1.add(lblNewLabel_1);

JLabel lblNewLabel_2 = new JLabel("CUSTOMER MANAGEMENT");
lblNewLabel_2.setForeground(new Color(255, 255, 255));
lblNewLabel_2.setFont(new Font("Segoe UI", Font.BOLD, 11));
lblNewLabel_2.setBounds(4, 50, 152, 36);
panel_1.add(lblNewLabel_2);

JLabel lblNewLabel_2_1 = new JLabel("SOFTWARE");
lblNewLabel_2_1.setForeground(new Color(255, 255, 255));
lblNewLabel_2_1.setFont(new Font("Segoe UI", Font.BOLD, 11));
lblNewLabel_2_1.setBounds(50, 75, 67, 36);
panel_1.add(lblNewLabel_2_1);
```

```
JLabel lblGroupAdmin = new JLabel("");
lblGroupAdmin.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        Ds_Admin.main(null);
        JFrame frame = (JFrame)
SwingUtilities.getWindowAncestor(lblGroupAdmin);
        frame.setVisible(false);
    }
});
lblGroupAdmin.setIcon(new
ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\group_admin.png"));
lblGroupAdmin.setBounds(55, 380, 48, 54);
panel_1.add(lblGroupAdmin);

JLabel lblName = new JLabel("Tên KH:");
lblName.setFont(new Font("Segoe UI", Font.BOLD, 14));
lblName.setBounds(169, 17, 75, 29);
contentPane.add(lblName);

JLabel lblEmail = new JLabel("Email:");
lblEmail.setFont(new Font("Segoe UI", Font.BOLD, 14));
lblEmail.setBounds(170, 72, 75, 29);
contentPane.add(lblEmail);

JLabel lblPhone = new JLabel("SĐT:");
lblPhone.setFont(new Font("Segoe UI", Font.BOLD, 14));
lblPhone.setBounds(170, 126, 75, 29);
contentPane.add(lblPhone);

JLabel lblBirth = new JLabel("Ngày sinh:");
lblBirth.setFont(new Font("Segoe UI", Font.BOLD, 14));
```

```
lblBirth.setBounds(170, 178, 75, 29);
contentPane.add(lblBirth);

JLabel lblAddress = new JLabel("Địa chỉ:");
lblAddress.setFont(new Font("Segoe UI", Font.BOLD, 14));
lblAddress.setBounds(170, 230, 75, 29);
contentPane.add(lblAddress);

emailTxt = new JTextField();
emailTxt.setFont(new Font("Arial", Font.PLAIN, 12));
emailTxt.setColumns(10);
emailTxt.setBounds(255, 72, 245, 30);
contentPane.add(emailTxt);

fullnameTxt = new JTextField();
fullnameTxt.setFont(new Font("Arial", Font.PLAIN, 12));
fullnameTxt.setColumns(10);
fullnameTxt.setBounds(254, 17, 245, 30);
contentPane.add(fullnameTxt);

phoneTxt = new JTextField();
phoneTxt.setFont(new Font("Arial", Font.PLAIN, 12));
phoneTxt.setColumns(10);
phoneTxt.setBounds(255, 126, 245, 30);
contentPane.add(phoneTxt);

birthdayTxt = new JTextField();
birthdayTxt.setFont(new Font("Arial", Font.PLAIN, 12));
birthdayTxt.setColumns(10);
birthdayTxt.setBounds(255, 178, 245, 30);
contentPane.add(birthdayTxt);

addressTxt = new JTextField();
```

```
addressTxt.setFont(new Font("Arial", Font.PLAIN, 12));
addressTxt.setColumns(10);
addressTxt.setBounds(255, 230, 245, 30);
contentPane.add(addressTxt);

JSeparator separator_1 = new JSeparator();
separator_1.setBounds(164, 274, 342, 2);
contentPane.add(separator_1);

JTabbedPane tabbedPane = new JTabbedPane(JTabbedPane.TOP);
tabbedPane.setForeground(new Color(0, 128, 0));
tabbedPane.setBackground(new Color(192, 192, 192));
tabbedPane.setBounds(510, 14, 4, 431);
contentPane.add(tabbedPane);

 JScrollPane scrollPane = new JScrollPane();
scrollPane.setBounds(523, 17, 503, 427);
contentPane.add(scrollPane);

customerTable = new JTable();
customerTable.setBackground(new Color(175, 238, 238));
customerTable.setFont(new Font("Segoe UI", Font.PLAIN, 14));
customerTable.setModel(new DefaultTableModel(
    new Object[][] {
        },
    new String[] {
        "STT", "Họ tên", "Email", "SĐT", "Ngày sinh", "Địa chỉ"
    }
));
scrollPane.setViewportView(customerTable);

JButton saveBtn = new JButton("Lưu");
```

```
saveBtn.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\Add.png"));
saveBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        saveBtnActionPerformed(e);
    }
});
saveBtn.setFont(new Font("Segoe UI", Font.BOLD, 12));
saveBtn.setBounds(182, 298, 84, 29);
contentPane.add(saveBtn);

JButton delBtn = new JButton("Xoá");
delBtn.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\delete.png"));
delBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        delBtnActionPerformed(e);
    }
});
delBtn.setFont(new Font("Segoe UI", Font.BOLD, 12));
delBtn.setBounds(286, 298, 84, 29);
contentPane.add(delBtn);

JButton searchBtn = new JButton("Tìm kiếm");
searchBtn.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\search.png"));
searchBtn.setFont(new Font("Segoe UI", Font.BOLD, 12));
searchBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        searchBtnActionPerformed(e);
    }
});
searchBtn.setBounds(394, 298, 106, 29);
contentPane.add(searchBtn);
```

```
JButton cannelBtn = new JButton("Trở lại");
cannelBtn.setIcon(new ImageIcon("D:\\Nam_4\\HK2\\LTrinhHDT\\my_icons\\go_back.png"));
cannelBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        thucHienTaiLaiDuLieu(e);
    }
});
cannelBtn.setFont(new Font("Segoe UI", Font.BOLD, 12));
cannelBtn.setBounds(394, 348, 106, 29);
contentPane.add(cannelBtn);

JLabel lblNewLabel = new JLabel("(yyyy-MM-dd)");
lblNewLabel.setFont(new Font("Segoe UI", Font.ITALIC, 10));
lblNewLabel.setBounds(434, 163, 65, 13);
contentPane.add(lblNewLabel);

JPanel panel_2 = new JPanel();
panel_2.setBackground(new Color(211, 211, 211));
panel_2.setBounds(160, 10, 4, 440);
contentPane.add(panel_2);
setLocationRelativeTo(null);
}

public void saveBtnActionPerformed(java.awt.event.ActionEvent evt) {
    if (currentPos >= 0) {
        dataList.get(currentPos).setFullscreen(fullnameTxt.getText());
        dataList.get(currentPos).setEmail(emailTxt.getText());
        dataList.get(currentPos).setPhoneNumber(phoneTxt.getText());
        dataList.get(currentPos).setBirthday(birthdayTxt.getText());
        dataList.get(currentPos).setAddress(addressTxt.getText());
    }

    CustomerModify.update(dataList.get(currentPos));
}
```

```
        currentPos = -1;

    } else {
        Customer customer = new Customer(
            0,
            fullnameTxt.getText(),
            emailTxt.getText(),
            phoneTxt.getText(),
            birthdayTxt.getText(),
            addressTxt.getText()

        );
        CustomerModify.insert(customer);
        dataList = CustomerModify.getCustomerList(null);
    }
    showData();

    fullnameTxt.setText("");
    emailTxt.setText("");
    phoneTxt.setText("");
    birthdayTxt.setText("");
    addressTxt.setText("");
}

public void delBtnActionPerformed(java.awt.event.ActionEvent evt) {
    if(currentPos == -1) {
        JOptionPane.showMessageDialog(rootPane, "Chưa chọn khách hàng cần xoá, vui lòng kiểm tra lại.");
        return;
    }
    int option = JOptionPane.showConfirmDialog(rootPane, "Bạn có chắc muốn xoá khách hàng này không?");
    if(option == 0) {
        CustomerModify.delete(dataList.get(currentPos).getId());
        dataList.remove(currentPos);
    }
}
```

```
        currentPos = -1;
        showData();
    }

    fullnameTxt.setText("");
    emailTxt.setText("");
    phoneTxt.setText("");
    birthdayTxt.setText("");
    addressTxt.setText("");
}

public void searchBtnActionPerformed(java.awt.event.ActionEvent evt) {
    String s = JOptionPane.showInputDialog("Nhập tên khách hàng cần tìm kiém.");
    if(!s.isEmpty()) {
        s = "%" + s + "%";
    }
    dataList = CustomerModify.getCustomerList(s);
    showData();
}

public void thucHienTaiLaiDuLieu(java.awt.event.ActionEvent evt) {
    dataList = CustomerModify.getCustomerList("");
    showData();
    JOptionPane.showMessageDialog(this, "Đã Loading!");
}

public void hienThiAbout() {
    JOptionPane.showMessageDialog(this, "Phần mềm quản lý khách hàng
của Nhóm 3.");
    showData();
}
```

```
public void thoatKhoiPhanMem() {  
    int option = JOptionPane.showConfirmDialog(rootPane, "Bạn có chắc  
chắn muốn thoát khỏi phần mềm quản lý khách hàng không?");  
    if(option == 0) {  
        System.exit(0);  
        showData();  
    }  
}  
  
public void saveFile(String path) {  
    try {  
        this.model.setTenFile(path);  
        FileWriter fileWriter = new FileWriter(path);  
        BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);  
        for (Customer customer : dataList) {  
            StringBuilder builder = new StringBuilder();  
            builder.append(customer.getFullname()).append(",");  
            builder.append(customer.getEmail()).append(",");  
            builder.append(customer.getPhoneNumber()).append(",");  
            builder.append(customer.getBirthday()).append(",");  
            builder.append(customer.getAddress());  
            bufferedWriter.write(builder.toString());  
            bufferedWriter.newLine();  
        }  
        bufferedWriter.close();  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}  
  
public void loadFile(String path) {  
    try {  
        dataList.clear();  
    }
```

```
Scanner scanner = new Scanner(new File(path));
while (scanner.hasNextLine()) {
    String line = scanner.nextLine();
    String[] fields = line.split(",");
    Customer customer = new Customer(currentPos, fields[0],
fields[1], fields[2], fields[3], fields[4]);
    dataList.add(customer);
    currentPos++;
}
scanner.close();
} catch (Exception e) {
    e.printStackTrace();
}

}

public void thucHienSaveFile() {
    if (this.model.getTenFile().length() > 0) {
        String path = this.model.getTenFile() + ".txt";
        saveFile(path);
    } else {
        JFileChooser fs = new JFileChooser(new File("d:\\"));
        fs.setDialogTitle("Save a File");
        int result = fs.showSaveDialog(this);
        if (result == JFileChooser.APPROVE_OPTION) {
            File fi = fs.getSelectedFile();
            String path = fi.getAbsolutePath() + ".txt";
            saveFile(path);
        }
    }
}

public void openFile(String path) {
    try {
```

```
FileInputStream fis = new FileInputStream(path);
ObjectInputStream ois = new ObjectInputStream(fis);
Object obj;
List<Customer> customers = new ArrayList<Customer>();
while ((obj = ois.readObject()) != null) {
    if (obj instanceof Customer) {
        Customer customer = (Customer) obj;
        customers.add(customer);
    }
}
ois.close();
showCustomerTable(customers);
} catch (EOFException e) {
    // End of file reached
} catch (StreamCorruptedException e) {
    JOptionPane.showMessageDialog(this, "File không hợp lệ hoặc bị hỏng", "Lỗi", JOptionPane.ERROR_MESSAGE);
} catch (Exception e) {
    e.printStackTrace();
}
}

public void showCustomerTable(List<Customer> customers) {
    JTable customerTable = new JTable();
    customerTable.setBackground(new Color(175, 238, 238));
    customerTable.setFont(new Font("Segoe UI", Font.PLAIN, 14));
    customerTable.setModel(new DefaultTableModel(
        new Object[][] {},
        new String[] {"STT", "Họ tên", "Email", "SĐT", "Ngày sinh", "Địa chỉ"})
    );
    DefaultTableModel model = (DefaultTableModel)
customerTable.getModel();
    int i = 1;
```

```
for (Customer customer : customers) {
    model.addRow(new Object[] {i, customer.getFullname(),
customer.getEmail(),
customer.getPhoneNumber(), customer.getBirthday(),
customer.getAddress()});
    i++;
}
JScrollPane scrollPane = new JScrollPane(customerTable);
JFrame frame = new JFrame("Danh sách khách hàng");
frame.getContentPane().add(scrollPane, BorderLayout.CENTER);
frame.setSize(800, 600);
frame.setLocationRelativeTo(null);
frame.setVisible(true);
}

public void thucHienOpenFile() {
    JFileChooser fs = new JFileChooser(new File("d:\\"));
    fs.setDialogTitle("Open a File");
    int result = fs.showOpenDialog(this);
    if (result == JFileChooser.APPROVE_OPTION) {
        File fi = fs.getSelectedFile();
        loadFile(fi.getAbsolutePath());
        showCustomerTable(dataList);
    }
}
```

7. Test.java

```
package test;

import javax.swing.UIManager;
import view.Login_Admin;
```

```
public class Test {  
    public static void main(String[] args) {  
        try {  
  
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());  
            new Login_Admin();  
        } catch (Exception e) {  
            e.printStackTrace();  
        }  
    }  
}
```

KẾT LUẬN CHUNG

Phần mềm quản lý khách hàng là một công cụ hữu ích để quản lý và tương tác với khách hàng một cách hiệu quả. Nhờ phần mềm này, ta có thể thu thập, lưu trữ và quản lý thông tin khách hàng một cách tự động, đồng thời có thể thực hiện các bước thêm, sửa, xóa, hiển thị và tìm kiếm khách hàng một cách nhanh chóng và thuận tiện.

HẾT!