FPT ACADEMY INTERNATIONAL

FPT – APTECH COMPUTER EDUCATION

Centre Name: ACE-THUDUC-1-FPT

Address: 62 Street 36, Van Phuc Residential Area, Hiep Binh Phuoc Ward, Thu Duc City

Pharmacy

Supervisor	**Mr. Pham Cong Danh**	
Batch	**T5.2308.M0**	
Group	**Group 02**	
No	**Student code**	**Full name**
1	Student1501057	Hoang Gia Huy
2	Student1501059	Nguyen Anh Quan
3	Student1501053	Nguyen Anh Minh
4	Student1501060	Tran Nhat Linh
5	Student1491746	Doan Duc Do

Month: June Year: 2024

This is to certify that

Mr. **Hoang Gia Huy**

Mr. **Nguyen Anh Quan**

Mr. **Nguyen Anh Minh**

Mr. **Tran Nhat Linh**

Mr. **Doan Duc Do**

Have successfully Designed & Developed

Pharmacy

Submitted by:

Mr. PHAM CONG DANH

Date Of Issue:

10/7/2024

Authorized Signature:

CONTENT

ACKNOWLEDGE

We extend our deepest gratitude and appreciation to all those who have contributed to the successful completion of our Semester 2 project as part of the Advanced Diploma of Software Engineering program. This collaborative effort involved the dedicated work of our team of five students, who have tirelessly strived to deliver a robust and innovative pharmacy management application.

First and foremost, we would like to express our heartfelt thanks to our esteemed teacher, Mr. Danh, whose guidance and mentorship have been instrumental throughout the development of our project. His wealth of knowledge, unwavering support, and constructive feedback have been invaluable, shaping not only our technical skills but also fostering a deep understanding of software engineering principles.

The foundation of our project lies in the integration of various technologies, and we want to acknowledge the pivotal role played by JavaFX and Scene Builder in NetBeans. These tools have been the building blocks of our project, enabling us to create a dynamic and feature-rich application. The hands-on experience gained in utilizing these technologies has not only enriched our technical prowess but also provided practical insight into real-world software development.

The collaborative nature of this project has allowed us to cultivate effective teamwork and communication skills. Working as a cohesive unit, we learned the importance of coordination, task delegation, and problem-solving. These invaluable skills extend beyond the realm of software engineering and are applicable to various aspects of our academic and professional lives.

Furthermore, we want to express our gratitude to our fellow team members for their commitment and diligence. Each member has brought a unique set of skills and perspectives, contributing to the overall success of the project.

In conclusion, this project has been a journey of growth, learning, and camaraderie. We are sincerely thankful to everyone who has been part of this endeavor, shaping it into a rewarding experience that will undoubtedly influence our future endeavors in the field of software engineering.

INTRODUCTION

Greetings! We are excited to present our project in the realm of software engineering. As part of our Advanced Diploma studies, our team of five diligent students has developed a comprehensive pharmacy management application. The primary objective is to provide pharmacies with a user-friendly platform to efficiently manage their business operations.

We aim to deliver an application that is not only visually appealing but also technologically robust. Our focus is on creating a seamless user experience, enabling pharmacies to easily handle tasks such as inventory management, prescription tracking, and customer care. With intuitive management features, our goal is to simplify the management process, providing users with a convenient and efficient means to run their pharmacies.

We invite you to join us on this technological journey as we unveil a pharmacy management application that not only showcases our technical expertise but also enhances the operational efficiency of every pharmacy.

PROBLEM DEFINITION

Our project focuses on addressing common struggles that pharmacies face when managing and operating their businesses. Currently, tasks like inventory management, order tracking, and customer interaction are not always straightforward and can be time-consuming. Additionally, existing solutions often do not leverage the latest technologies, resulting in a less smooth user experience. Our goal is to tackle these issues by creating a simple and effective pharmacy management application.

We utilize JavaFX and Scene Builder to build the user interface, along with SQL for data storage and management. In doing so, we aim to create an application that users can easily utilize to manage inventory, record orders, and interact with customers efficiently. We believe that this project will help improve the productivity and working experience of pharmacies, enabling them to save time and enhance their business management.

CUSTOMER REQUIREMENT SPECIFICATIONS

1.USER/INTERFACE

Input:

1. **Forms and Text Fields:** Users can input information through forms and text fields. This could include filling out personal details, selecting preferences, or entering search queries.
2. **Buttons and Links** Users interact with buttons and links to navigate through different sections of the pharmacy management app, submit forms, or trigger specific actions. Thanks to the user-friendly and intuitive interface, users can easily check inventory status, manage prescriptions, and track sales revenue. The app also provides real-time notifications, allowing users to quickly stay updated on important changes or the latest information.
3. **Dropdowns and Selections:** Users can make selections from dropdown menus or choose options from lists to customize their experience.
4. **Checkboxes and Radio Buttons:** Users can provide input by selecting checkboxes or radio buttons to indicate preferences or choices.
5. **Uploads:** If applicable, users may upload files, images, or documents using file input fields.

Process:

1. **Form Submission:** When users fill out forms and submit them, the application processes the input, which may involve data validation and verification.
2. **Dynamic Updates:** Based on user input or interactions, certain sections of the application may dynamically update without requiring a full app reload.
3. **Client-Side Scripting:** JavaFX can be used for client-side scripting to enhance user interaction, validate input, and dynamically modify the content without reloading the entire page.

Output:

1. **Displaying Results:** The application outputs information based on user input, such as displaying search results, updated content, or personalized recommendations.
2. **Error Messages:** If there are issues with the input, the application may output error messages to guide users in correcting their input.
3. **Confirmation Messages:** Users may receive confirmation messages for successful actions, such as submitting a form or completing a transaction.
4. **Visual Changes:** The application may undergo visual changes to reflect the processed information, such as updating images, text, or layout dynamically.

2. ADMIN

Input

1. **Login Credentials:** The system requires input in the form of username and password for authentication during the login process.
2. **Configuration Settings:** Administrators may input or modify configuration settings, such as system preferences, security parameters, or feature toggles.

Process:

1. **Authentication:** The system processes login credentials to authenticate administrators and grant access to the admin panel.
2. **Authorization:** Once authenticated, the system checks for appropriate authorization levels to determine the scope of actions an administrator can perform.
3. **Configuration Processing:** Changes made to configuration settings are processed by the system to implement the updated preferences or rules.

Output:

1. **Admin Dashboard:** After successful login, the system outputs an admin dashboard, providing an overview of system status, analytics, or key metrics.
2. **Error Messages:** In case of incorrect login credentials or invalid input during configuration, the system outputs error messages to guide administrators.
3. **Activity Logs:** The system generates logs to record administrator activities, helping to track changes and monitor system interactions.
4. **System Status Updates:** If configuration settings are modified, the system outputs updates to reflect changes in system behavior or functionality.
5. **Confirmation Messages:** After successful actions, such as data uploads or configuration updates, the system outputs confirmation messages to inform administrators.

Additional Aspects:

1. **Role Management:** The system allows administrators to manage user roles and permissions, defining who can access certain features or perform specific actions.
2. **Security Measures:** Input validation, secure authentication protocols, and encryption are crucial components to ensure the security of admin interactions.
3. **Notification System:** The system may include a notification system to alert administrators about critical events, updates, or potential issues.
4. **Backup and Recovery:** Administrators may have the capability to initiate backup processes and recovery procedures in case of data loss or system failures.

Hardware / Software requirement

Hardware:

For Setting Up the Application

- Processor Intel Core I3 or higher.
- Memory 6 GB RAM or greater.
- Monitor: Super VGA (1024x768) or higher resolution
- Internet Access: Modem/ADSL Internet access is required

Software:

Server:

- Operation System: Window 7 or later.
- Databases: SQL.
- Development Framework: JavaFX
- Browser: Google Chrome version 35.

Client:

- Operation System Window 7 or higher.
- Browser Google Chrome , MS-Edge, Firefox

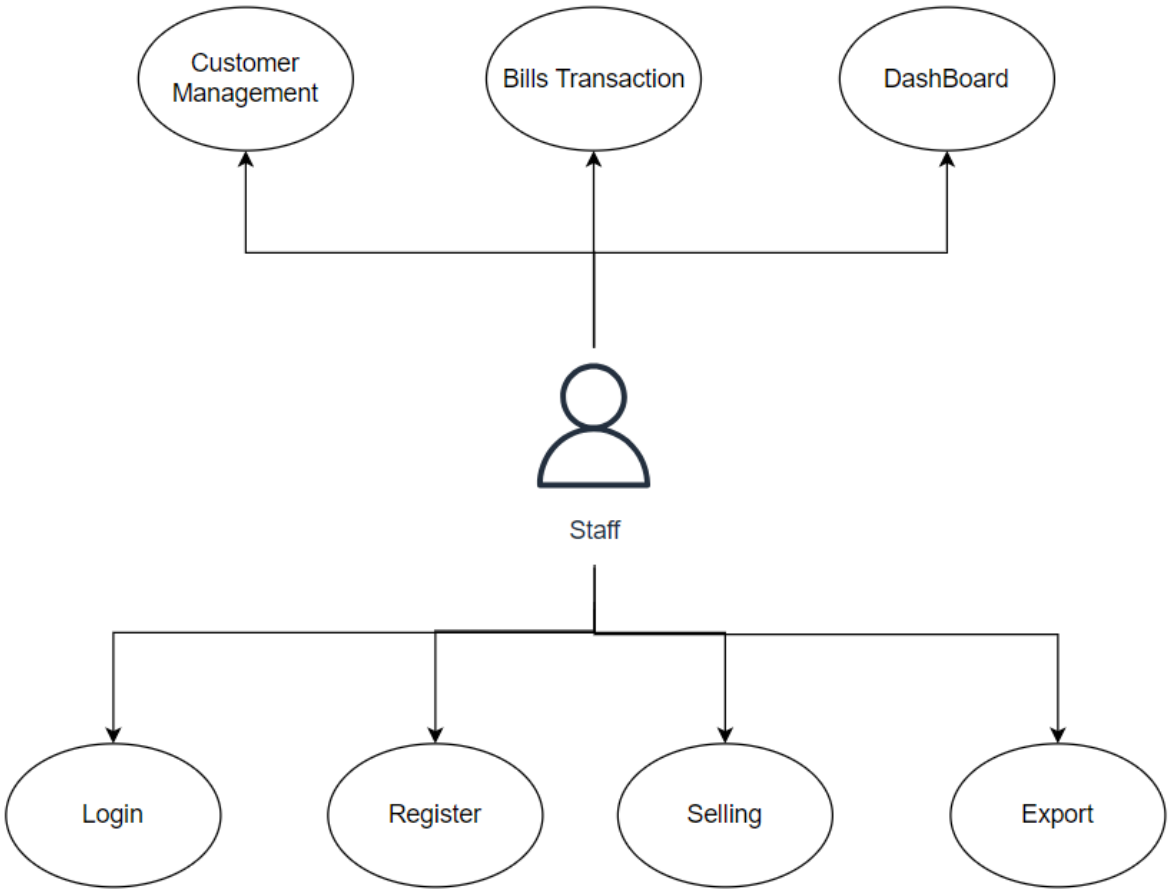
TASK SHEET REVIEW 1

Project: Pharmacy			**Date of preparation of activity plan:**			
#	**Task**	**Prepared by**	**Start date**	**Actual Date**	**Team member name**	**status**
1	Acknowledgement	Huy	27/05/2024	31/05/2024	Huy	completed
2	Introduction	Huy	27/05/2024	31/05/2024	Huy	completed
3	Problem definition	Huy	27/05/2024	31/05/2024	Huy	completed
4	Customer requirement specification	Huy	27/05/2024	31/05/2024	Huy	completed
5	Scope of work	Huy	27/05/2024	31/05/2024	Huy	completed
6	Hardware/software requirement	Huy	27/05/2024	31/05/2024	Huy	completed
7	Task Sheet	Huy	27/05/2024	31/05/2024	Huy	completed
Review			Signature of instructor			
			Mr. Pham Cong Danh			

REVIEW 2

1 .USECASE

1.1 Use case for Staff



Login

Use case name	Login	
Actors	User	
Description	Staff who has registered an account can login	
Requirements	Staff provides username and password	
Pre-conditions	N/A	
Post-conditions	Success: Staff is logged in to application	
	Fail: Alert and refill the information	
Basic flow	**Actor's actions:**	**System's responses:**
	1. Actors click 'Login' button on Home Page 3. Actors input Email and Password, then click the 'Login' button	2. System redirects to Login Page with the following controls: - "Email" text field - "Password" text field - "Login" button 4. System checks the information 5. System redirects to Home page
Exceptions	**Actor's actions:**	**System's responses:**
	1. Actors input invalid email or password.	2.System redirects to Login page with the following controls: - "Email" text field - "Password" text field - "Login" button 3.System shows message: "Invalid email and password"

Register

Use case name	Export	
Actors	User	
Description	Create new staff account	
Requirements	Have a phone email to receive OTP when forgot password	
Pre-conditions	N/A	
Post-conditions	Success: Create a new account and return to login page	
	Fail: Alert and refill information	
Basic flow	**Actor's actions:**	**System's responses:**
	1. Actors click 'Sign Up" Button under the "Login" button	2.The systems show the form which force actors fill in like ID, Full name, Password, Confirm Password, Phone, Emai, Address, Role (set Staff default)
	3. Actors fill in the form and click the "Sign Up"	4. Systems check whether ID, Phone, Email exist. 5. System inserts the status into database 6. System shows success message and return to Login page
Exceptions	**Actor's actions:**	**System's responses:**

Selling

Use case name	Selling	
Actors	Staff	
Description	Staff can sell product form storage	
Requirements	Input name of medicine	
Pre-conditions	N/A	
Post-conditions	Success: Show information that matches search phrase	
	Fail: No information is shown	
Basic flow	**Actor's actions:**	**System's responses:**
	1. Actors input search phrase 3.Actors click information in list or view everything that matches search phrase 5. Actors click the products and click sell	2.System shows list of information that match the search phrase 4. Systems show bill in right table 6. System update products to SQL
Exceptions	**Actor's actions:**	**System's responses:**
	1.Actors enter is not wrong information	2.The system does not show the search information

Export:

Use case name	Export	
Actors	Staff	
Description	Export the data file to be printed to PDF	
Requirements	Data is not null	
Pre-conditions	N/A	
Post-conditions	Success: Export PDF to your computer	
	Fail: N/A	
Basic flow	**Actor's actions:**	**System's responses:**
	1.Actors click the "Export" button	2.System export to your computer
Exceptions	**Actor's actions:**	**System's responses:**
	1. N/A	2. N/A

Customer Management

Use case name	Customer Management	
Actors	Employee	
Description	Employee can manage all customer	
Requirements	N/A	
Pre-conditions	N/A	
Post-conditions	Success: Show up all customer's information	
	Fail: Actor can't do anything in the page	
Basic flow	**Actor's actions:**	**System's responses:**
	1.Actors click Customer Management button sidebar	2.System display the management page
Exceptions	**Actor's actions:**	**System's responses:**

	N/A	N/A

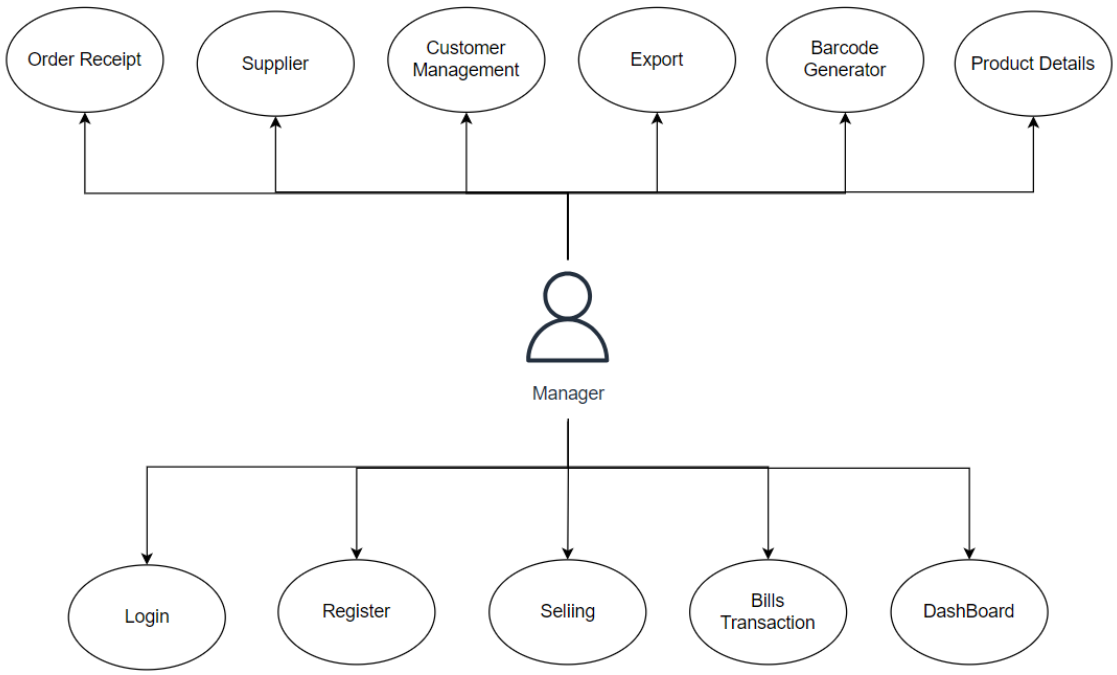
Bills Transaction

Use case name	Bills Transaction	
Actors	Staff	
Description	Manage bills	
Requirements	N/A	
Pre-conditions	N/A	
Post-conditions	Success: System shows the data	
	Fail: No information shown	
Basic flow	**Actor's actions:**	**System's responses:**
	1. Actors click on "Bills Transaction" button.	2.System shows Bills ID, Customer Name, Phone, Price, Method, Date
Exceptions	**Actor's actions:**	**System's responses:**
	1. Actors double click the bill	2. System shows clearly about that bill

Dashboard

Use case name	Dashboard	
Actors	Staff	
Description	See all information show in Home Page	
Requirements	N/A	
Pre-conditions		
Post-conditions	Success: N/A	
	Fail: N/A	
Basic flow	**Actor's actions:**	**System's responses:**
	1. Actors log in successfully	1.System redirects to Home page
Alternative flow	**Actor's actions:**	**System's responses:**
	N/A	N/A
Exceptions	**Actor's actions:**	**System's responses:**
	N/A	N/A

1.2 Use case for Managers



Order Receipt

Use case name	Order Receipt	
Actors	Admin	
Description	Admin order product	
Requirements	N/A	
Pre-conditions	N/A	
Post-conditions	Success: System show up bills(Receipt ID, Order Date, Receive Date, Supplier, Total, Status) and can delete	
	Fail: N/A	
Basic flow	**Actor's actions:**	**System's responses:**
	1. Actors click 'Order Receipt" button 3. Admin click "Add New" button and fill the form in right table and click "+ " button	2. System show up data 4. System will update to database
Exceptions	**Actor's actions:**	**System's responses:**

Supplier

Use case name	Supplier	
Actors	Admin	
Description	Show data and add new supplier	
Requirements	N/A	
Pre-conditions	N/A	
Post-conditions	Success: N/A	
	Fail: N/A	
Basic flow	**Actor's actions:**	**System's responses:**
	1. Actors choose "Supplier" button 3. Actors choose "Add New Supplier" 5. Actors click "Add" button	2. Systems show up the information products 4. System show the form with Name Address Phone Email 6. System checks the information and add to database
Exceptions	**Actor's actions:**	**System's responses:**
	N/A	N/A

Barcode Generator

Use case name	Barcode Generator	
Actors	Admin	
Description	Create barcode image and save to file	
Requirements	N/A	
Pre-conditions	N/A	
Post-conditions	Success: N/A	
	Fail: N/A	
Basic flow	**Actor's actions:**	**System's responses:**
	1. Actors click "Barcode Generator" button on Product Details 3. Actors click "Save"	2. System show the folder to save image 4. System will save to your computer
Exceptions	**Actor's actions:**	**System's responses:**
	N/A	N/A

Product Details

Use case name	Product Details	
Actors	Admin	
Description	See all full information of medicine	
Requirements	N/A	
Pre-conditions	N/A	
Post-conditions	Success: All information will show up	
	Fail: N/A	
Basic flow	**Actor's actions:**	**System's responses:**
	1. Actors click "Product Details" button	2. System show up
Exceptions	**Actor's actions:**	**System's responses:**
	N/A	N/A

Staff

Use case name	Staff	
Actors	Admin	
Description	See all information about employees	
Requirements	N/A	
Pre-conditions	N/A	
Post-conditions	Success: Display all Staff information	
	Fail: N/A	
Basic flow	**Actor's actions:**	**System's responses:**
	1. Actors click "Staff" button 3. Actors fill in form under the table information 5. Actors click Add button	2. System show up 3. System show the database add new like: ID, Name, Gender, Phone, Address, Gmail, Birthday, Role and Status 6. System save and update to database
Exceptions	**Actor's actions:**	**System's responses:**
	1. N/A	2. N/A

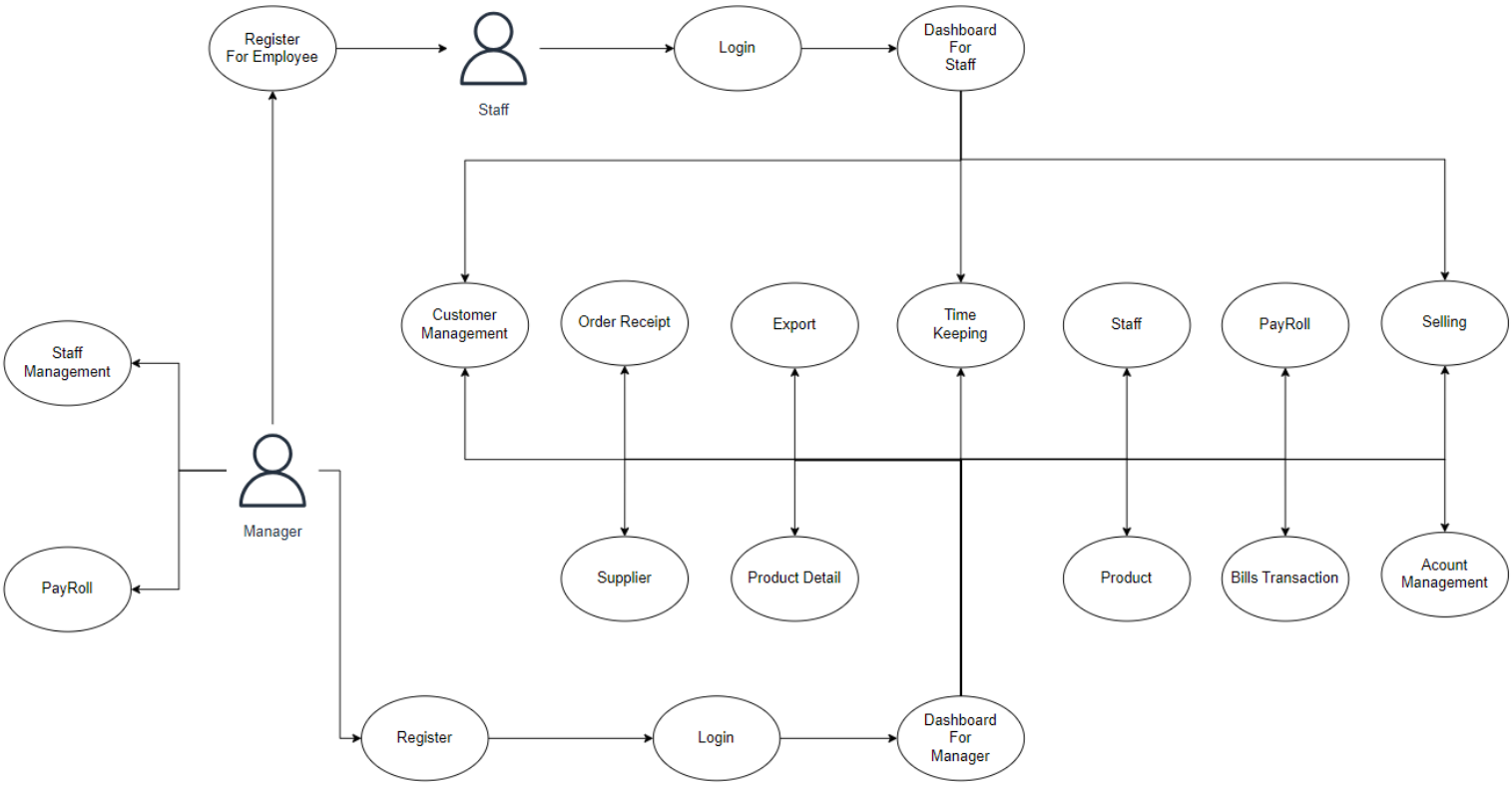
Payroll Management

Use case name	Payroll Management	
Actors	Admin	
Description	Payroll Management	
Requirements	admin	
Pre-conditions	Admin can see all information and can change it	
Post-conditions	Success: the systems show up the table view	
	Fail: Recheck	
Basic flow	**Actor's actions:**	**System's responses:**
	1. Actors click Fund Management button on tab 3. Actors select name of fund to see	2. System redirects to search fund section 4. System checks the information 5. System returns the result
Exceptions	**Actor's actions:**	**System's responses:**
	N/A	N/A

Time Keeping

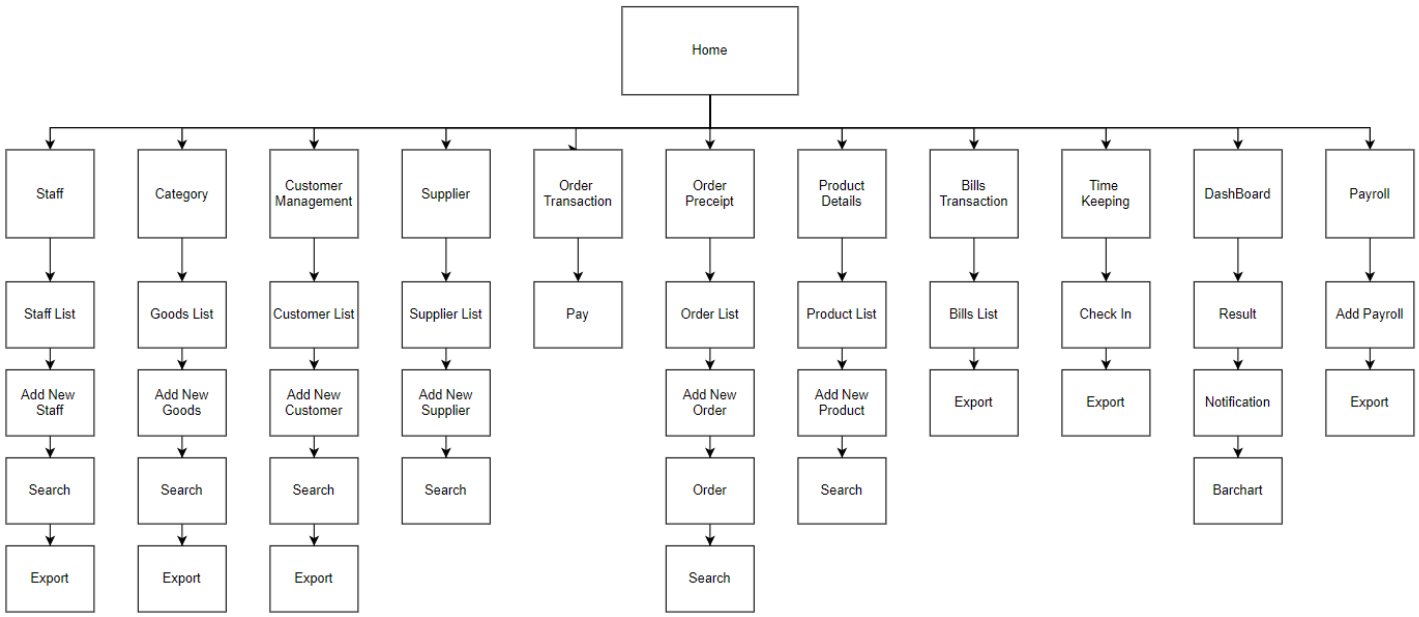
Use case name	Time Keeping	
Actors	Admin	
Description	See check in check out	
Requirements	Admin	
Pre-conditions	Admin can see all process checked of staff	
Post-conditions	Success: N/A	
	Fail: N/A	
Basic flow	**Actor's actions:**	**System's responses:**
	1. Actors click Time Keeping button 3. Actors click Check In	2. System show the table with ID, Staff ID, Name, Day, Check in, Check out 4. System get date time and update to database
Exceptions	**Actor's actions:**	**System's responses:**
	N/A	N/A

2.Data Flow Diagram (DFD)

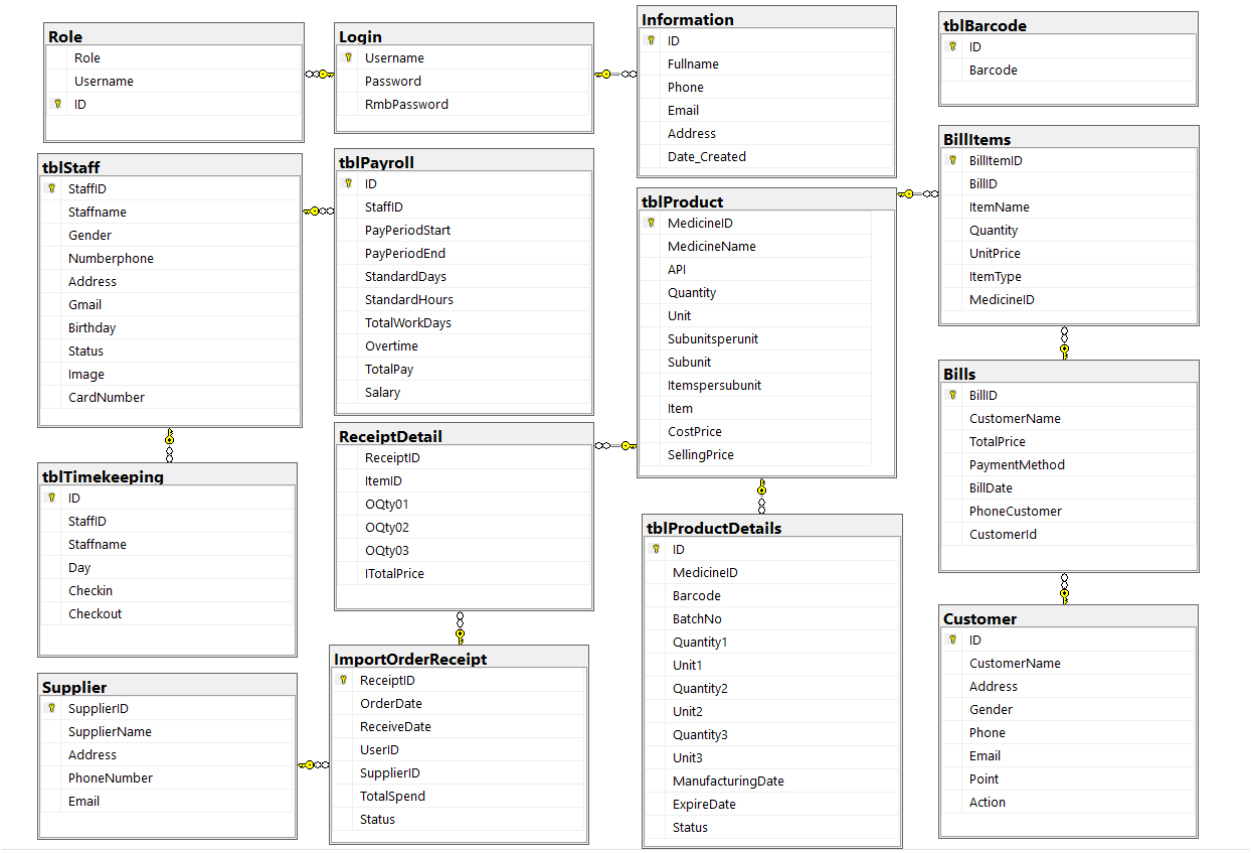


SITE MAP

1. User



ENTITY RELATIONSHIP DIAGRAM (ERD)



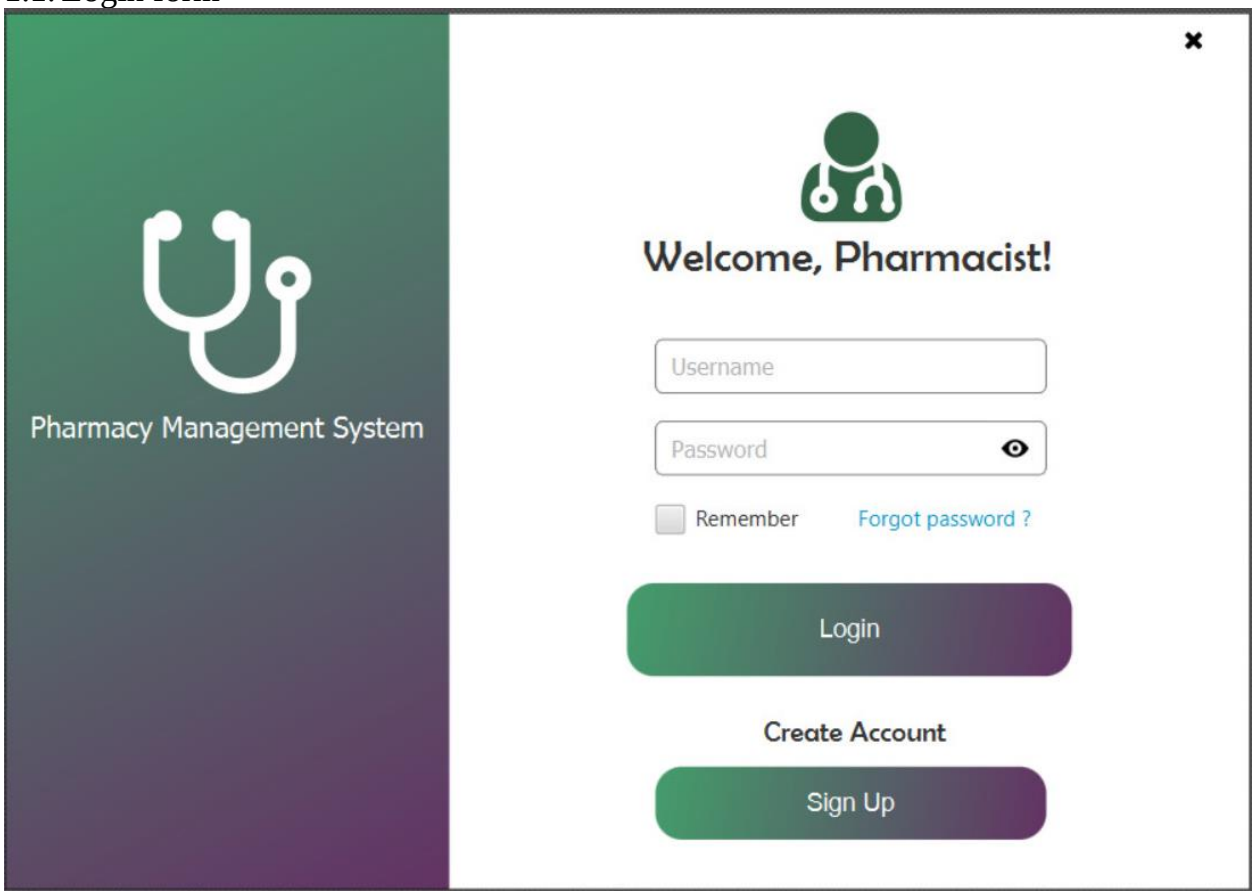
TASK SHEET REVIEW 2

<table>
<tr><th colspan="3">Project: Pharmacy</th><th colspan="4">Date of preparation of activity plan:</th></tr>
<tr><th>#</th><th>Task</th><th>Prepared by</th><th>Start date</th><th>Actuals Date</th><th>Team member name</th><th>status</th></tr>
<tr><td>1</td><td>Use case</td><td rowspan="4">Hoang Gia Huy
&
Nguyen Anh Minh
&
Nguyen Anh Quan
&
Tran Nhat Linh
&
Doan Duc Do</td><td>31/05/2024</td><td>07/06/2024</td><td rowspan="4">Hoang Gia Huy
&
Nguyen Anh Minh
&
Nguyen Anh Quan
&
Tran Nhat Linh
&
Doan Duc Do</td><td>completed</td></tr>
<tr><td>2</td><td>Data Flow Diagram (DFD)</td><td>31/05/2024</td><td>07/06/2024</td><td>completed</td></tr>
<tr><td>3</td><td>Site Map</td><td>31/05/2024</td><td>07/06/2024</td><td>completed</td></tr>
<tr><td>4</td><td>Entity Relationship Diagram (ERD)</td><td>31/05/2024</td><td>07/06/2024</td><td>completed</td></tr>
<tr><td colspan="3">Review</td><td colspan="4">Signature of instructor</td></tr>
<tr><td colspan="3"></td><td colspan="4"></td></tr>
<tr><td colspan="3"></td><td colspan="4">Mr. Pham Cong Danh</td></tr>
</table>

REVIEW 3
GUI DESIGN

1. User side

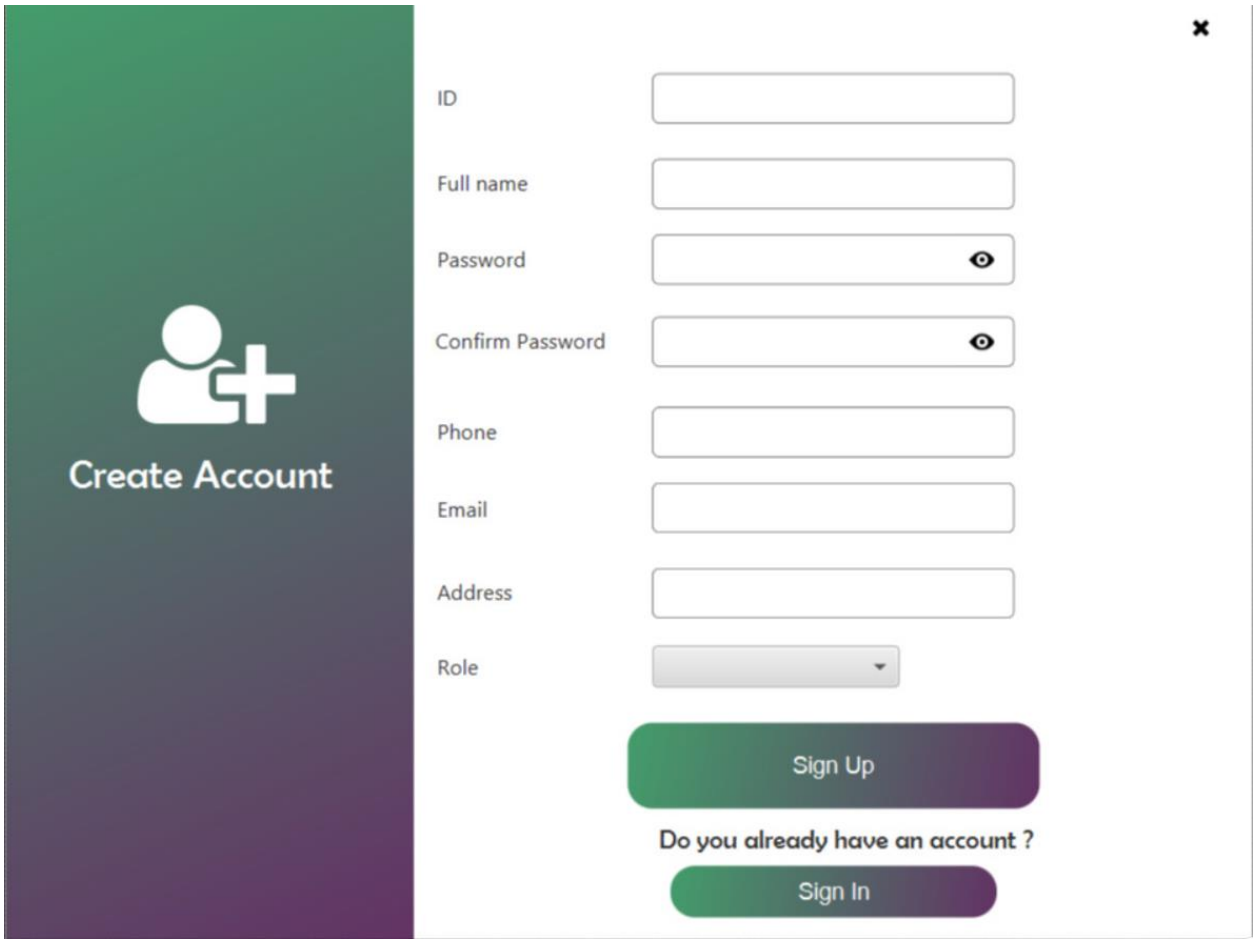
1.1. Login form



Users will fill in information log in:

+ Text field Username
  Validate: "Name is required!", "User does not exist!"
+ Text field Password
  Validate: "Password is required"
  "Wrong password"
+ Icon Eye/Eye Close: Show/Hide Password
+ Checkbox Remember: Remember password for the next sign in
+ Link "Forgot password": Click to go Reset password page
+ Login Button: Complete fill in and start validation to sign in
+ Sign Up Button: Click to go Register Staff Account page

1.2. Register page

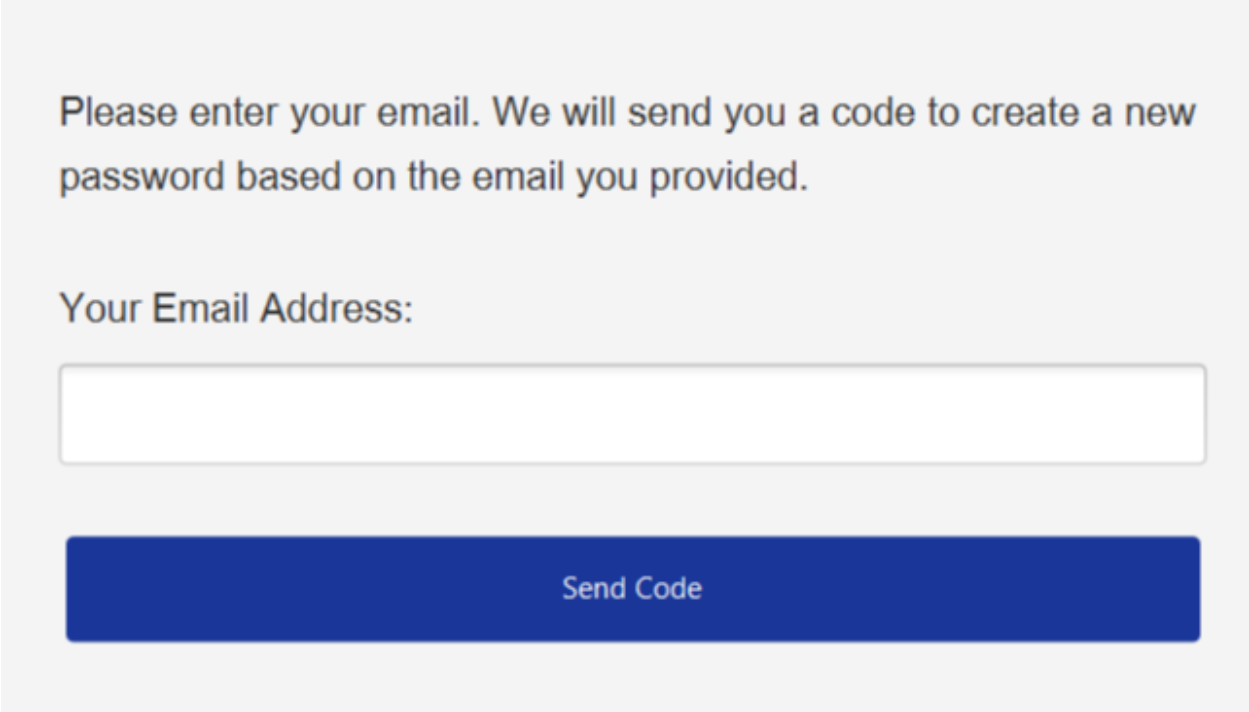


Users fill full information:

+ Text field String Name
    Validate: “Name is required”, “Name must not be longer than 30 characters”
+ Text field String Email (Gmail is registered with the company)
    Validate: “Email is required”, validate with letter “@”
+ Text field String Address
+ Text field Integer Phone
    Validate: “Phone is required”, “Format error, enter number”
+ Text field String Password
    Validate: “Password is required”, “Wrong password!”
+ Role: Admin-Staff (Default Admin).
+ Then click the “Sign Up" button to complete the login process.

If you have an account, you click the “Sign In" button to comeback sign up page.

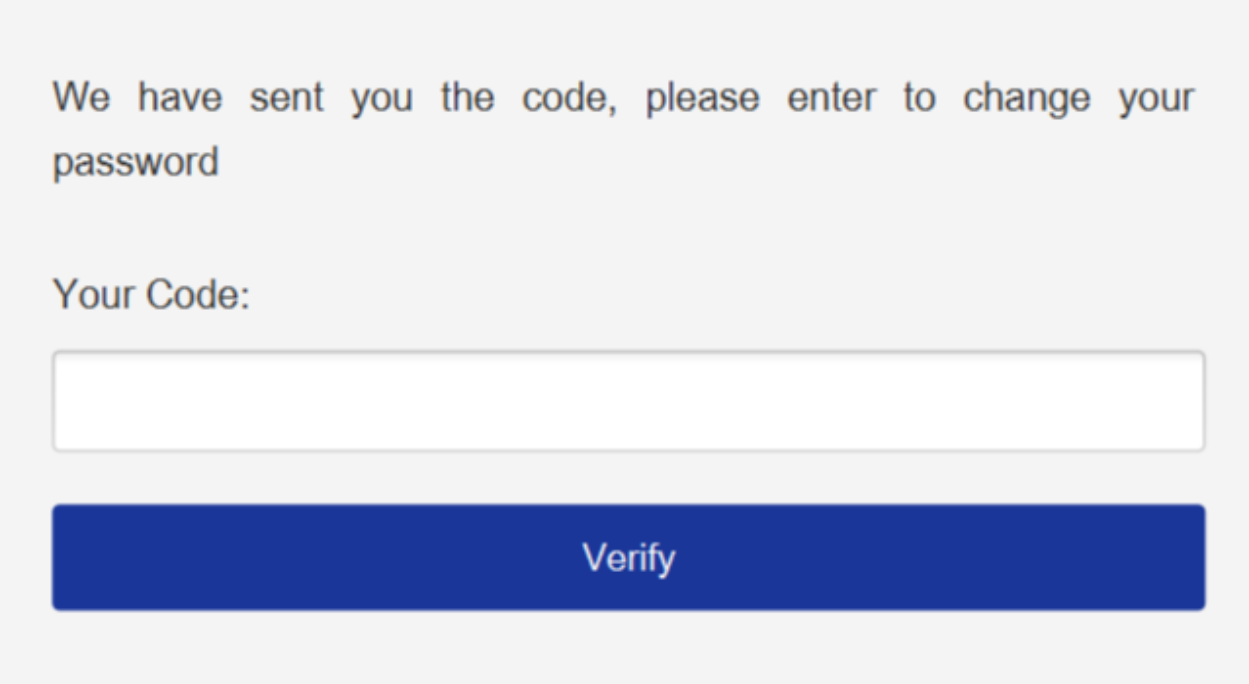
1.3. Forgot Password



+ User input email (which register before) to text field and click “Send Code” button to validate and receive OTP by email

Validate: “Email does not exist”, “Wrong email”, “Wrong format”

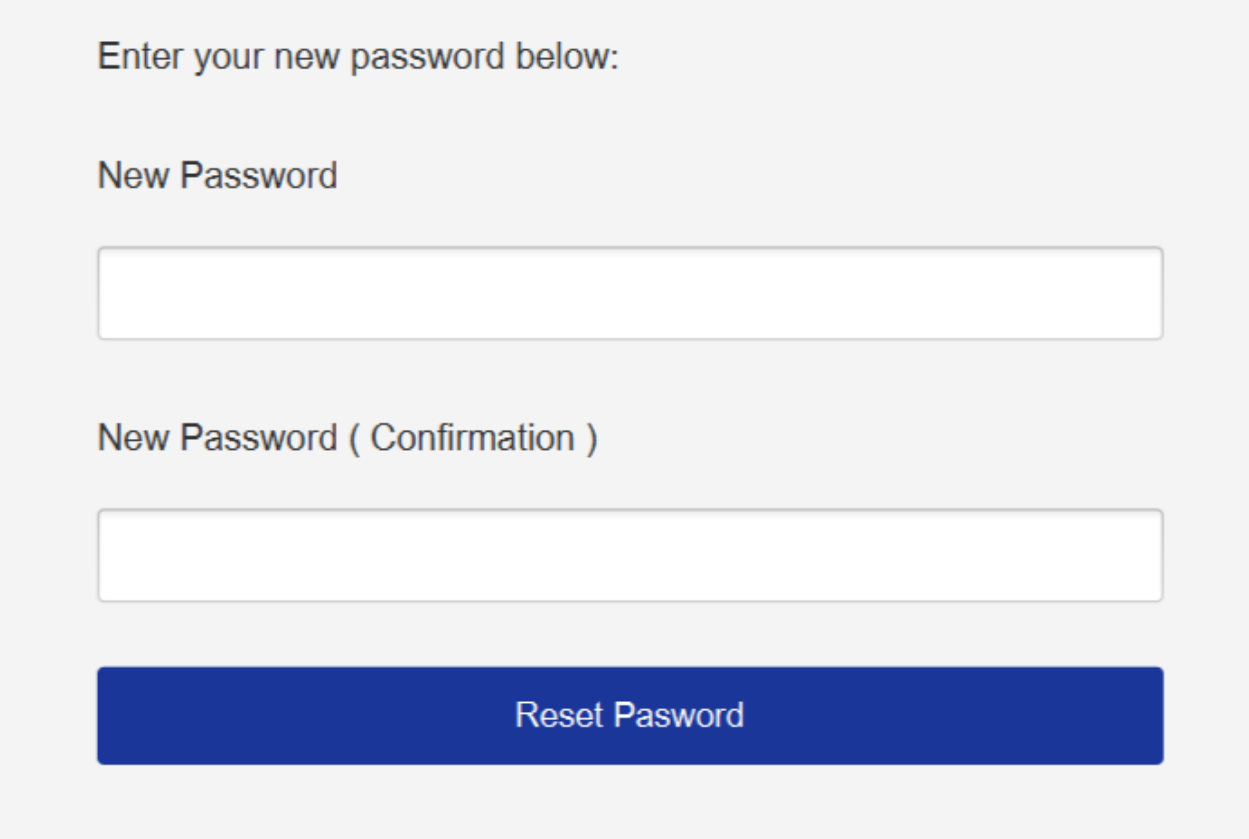
1.4. Verify OTP



+ After receive OTP email, user enter it to text field and click “Verify” button to validate

Validate: “Wrong OTP”

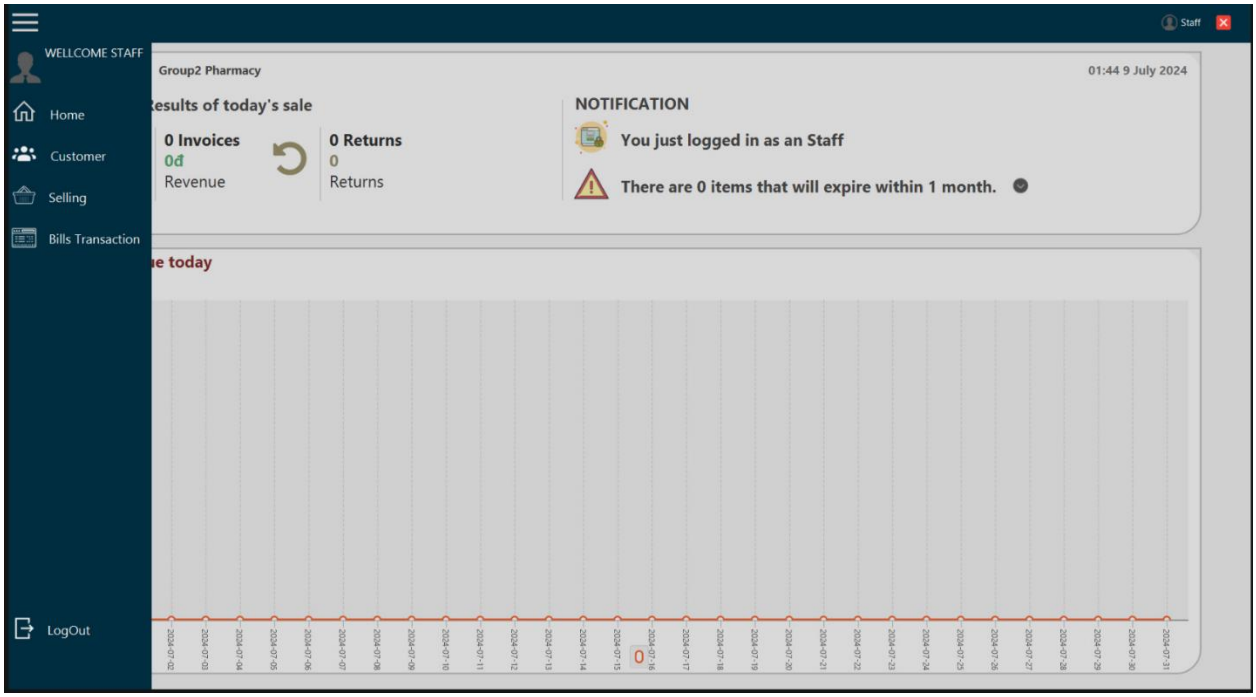
1.5. Reset Password



+ User input new password and confirm password and click “Reset Password” button to update new password into database and return to login page
Validate: “Password is not blank”, “Confirm Password is not blank”

1.6. Home

1.6.1. Dashboard for Staff:



The dashboard page displays:

-A tab bar in left side:

+Profile User with the title “Welcome Staff *Name of user*”

+ Button: Home to go Home page, Customer to go Customer Management Page, Selling to go Sell Page

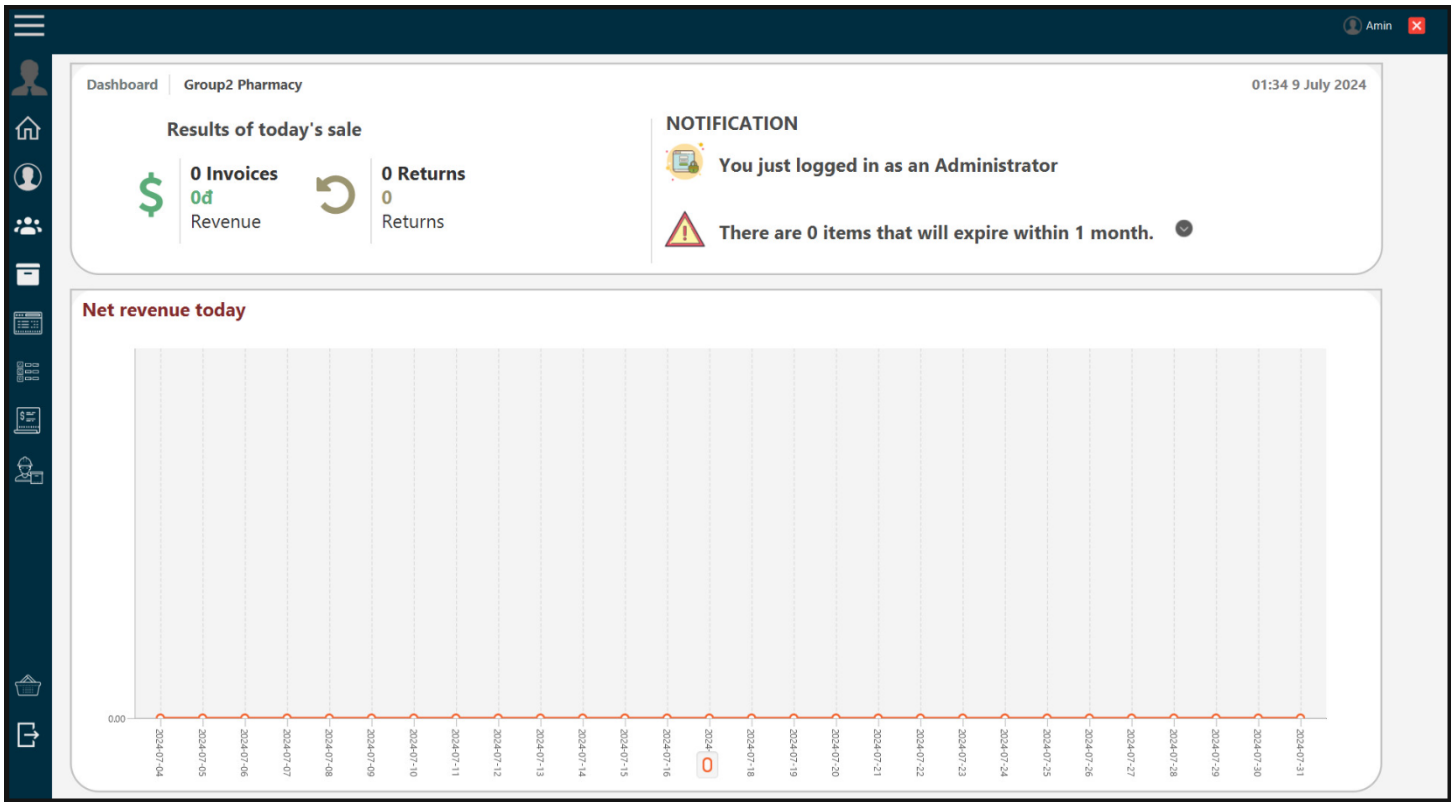
+ Log Out to sign out the application

- Dashboard in right side:

+ Above: show the report of the day, warning if sign up in unknow device, warning goods out of date…

+Below: displays a statistical data table

1.6.2. Dashboard for Admin:



The dashboard page displays:

-A tab bar in left side:

+Profile User with the title “Welcome Staff *Name of user*”

+ Button: Home to go Home page,
Customer to go Customer Management Page
Product go to Product Page
Transaction to go Transaction Page

+ Setting to custom your panel, color theme, account…

+ Log Out to sign out the application

+ Transaction Button: Click go to Transaction page

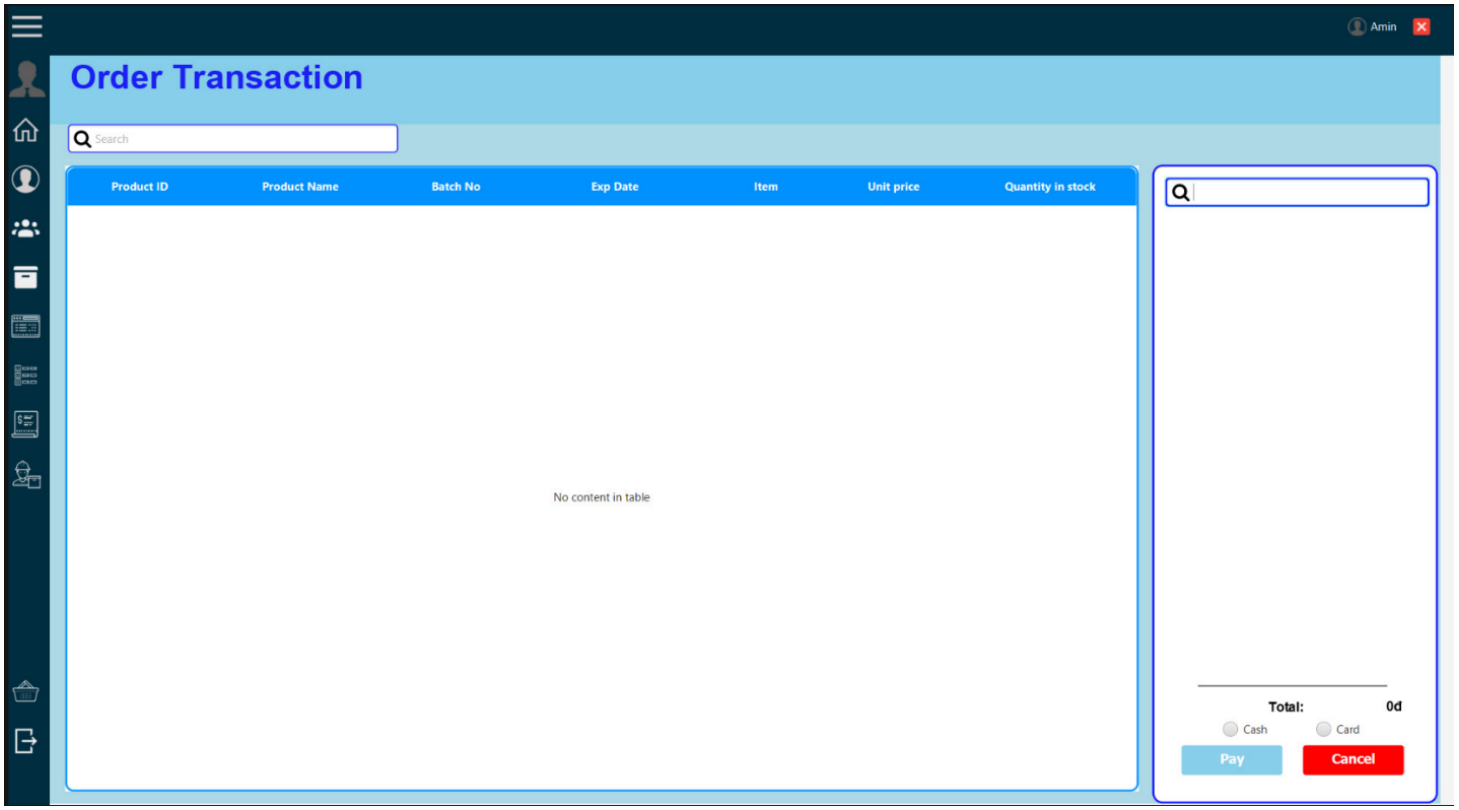
+ Account Management Button: Click to go to Account Management page

- Dashboard in right side:

+ Above: show the report of the day, warning if sign up in unknow device, warning goods out of date…

+Below: displays a statistical data table

1.6.3. Order Transaction



Search: + Search order by Name shown in table below in left side
+ Search customer by name or number in table below in right side

Users fill full information:

Integer Quantity

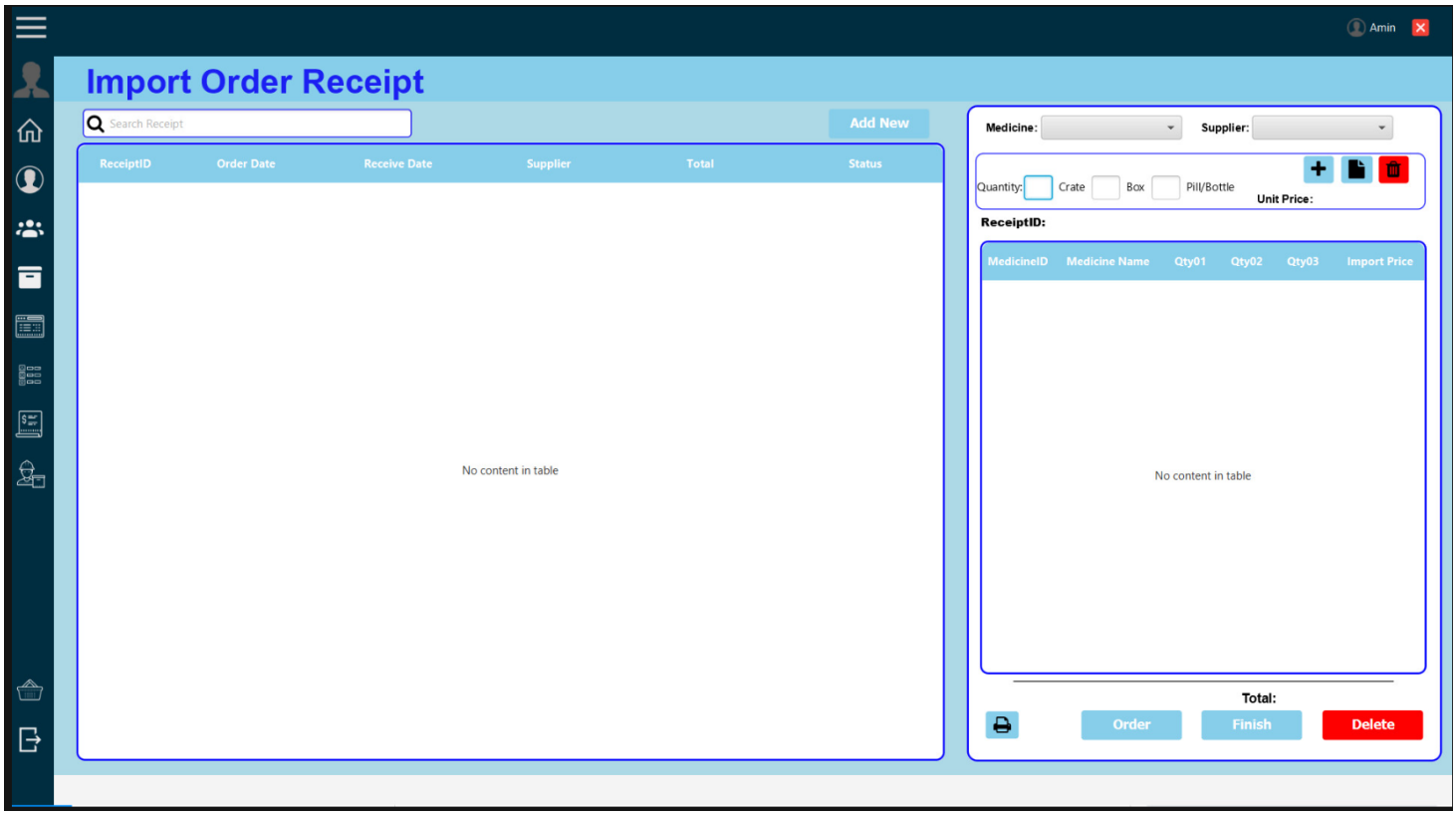
Integer Pill

Radio Choice Cash/Card to choose type of payment

Pay Button: to pay the order

Cancel Button: cancel process

1.7. Receipt Management



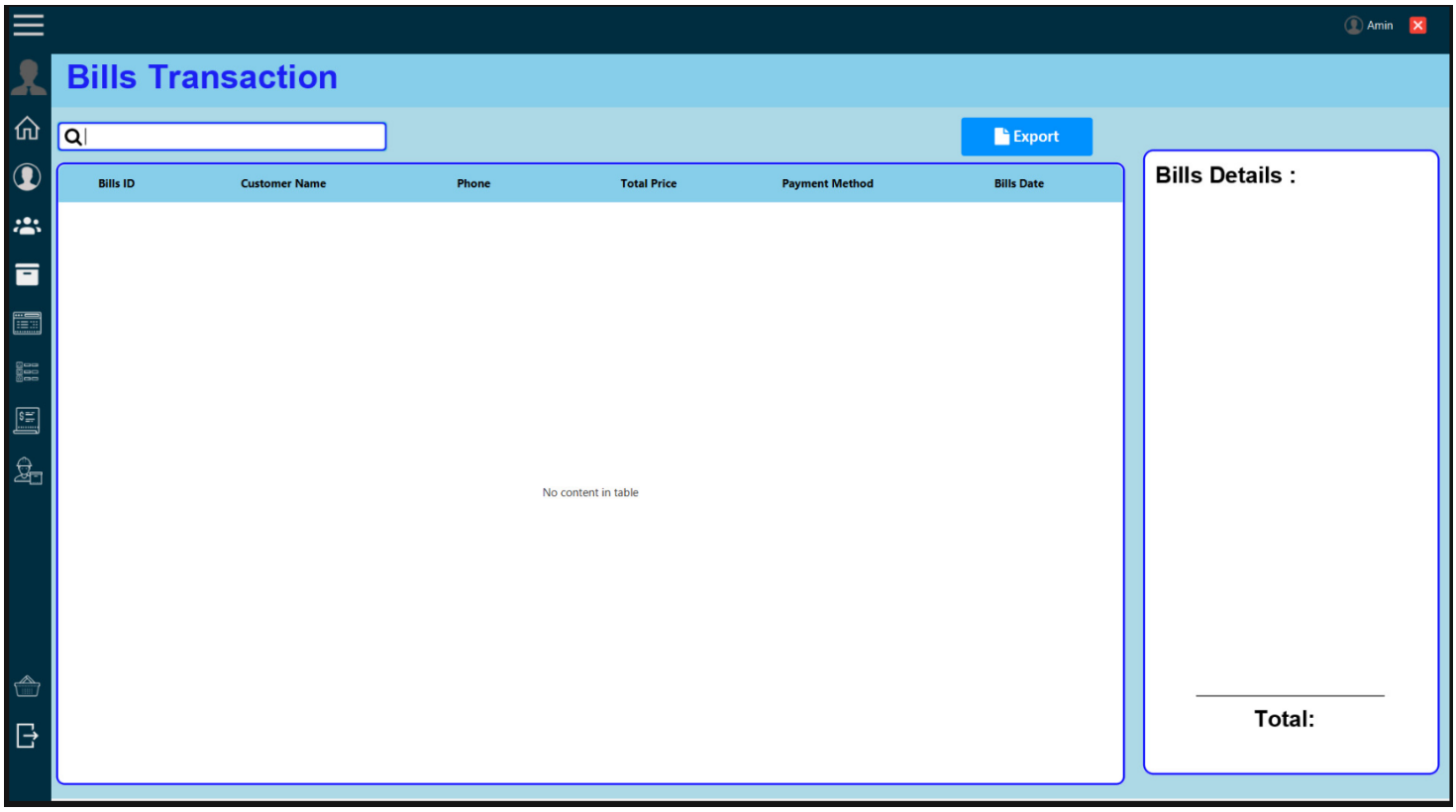
Search: + Search receipt by Name shown in table below in left side
+ Search bill by name or number in table below in right side

Users fill full information:

Integer Quantity
Integer Crate
Integer Box
Integer Pill/Bottle

Delete Button: Delete all product show in table
Print Button: Print a doc, excel… file to report
Cancel Button: Cancel process
Trash Icon: Delete the product
Order Button: Order receipt
Finish Button: Finish order process
Medicine Selection: Choose name of medicine
Supplier Selection: Choose name of supplier

2.1. Bills Transaction



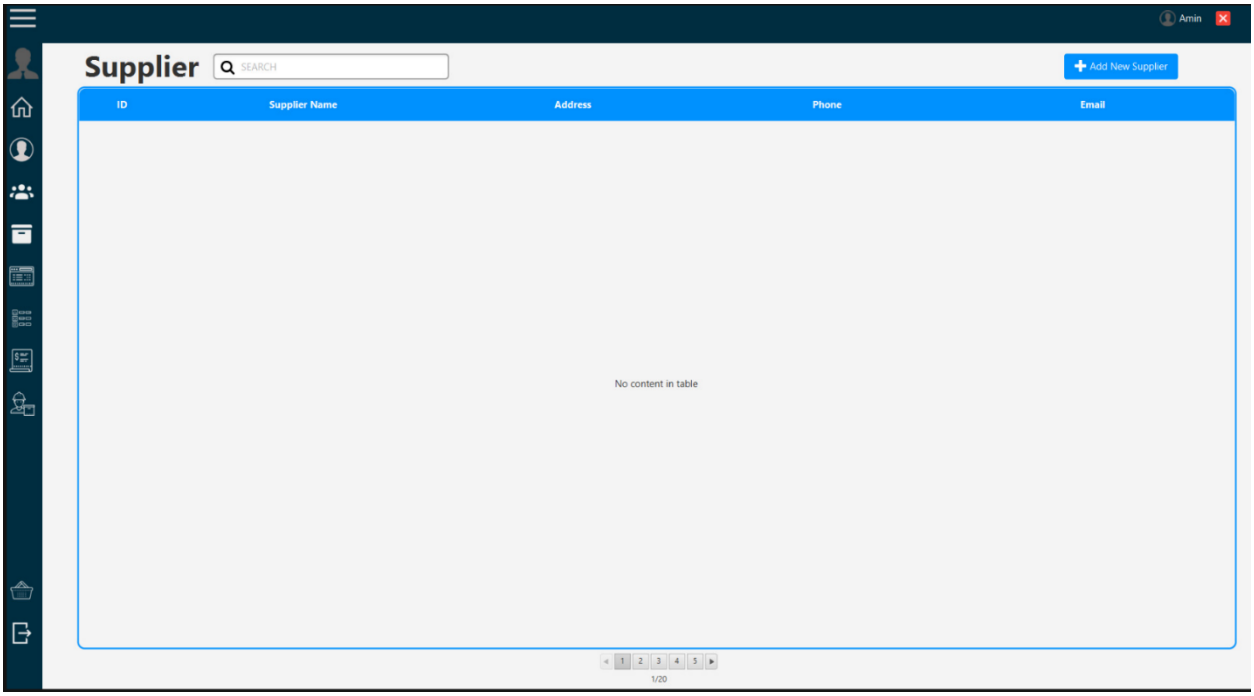
Search: + Search bills by Name shown in table below in left side
+ Search bill by Name or number in table belowin right side

Users fill full information:

Integer Quantity
Integer Crate
Integer Box
Integer Pill
Integer Batch No with can have a device support to scan

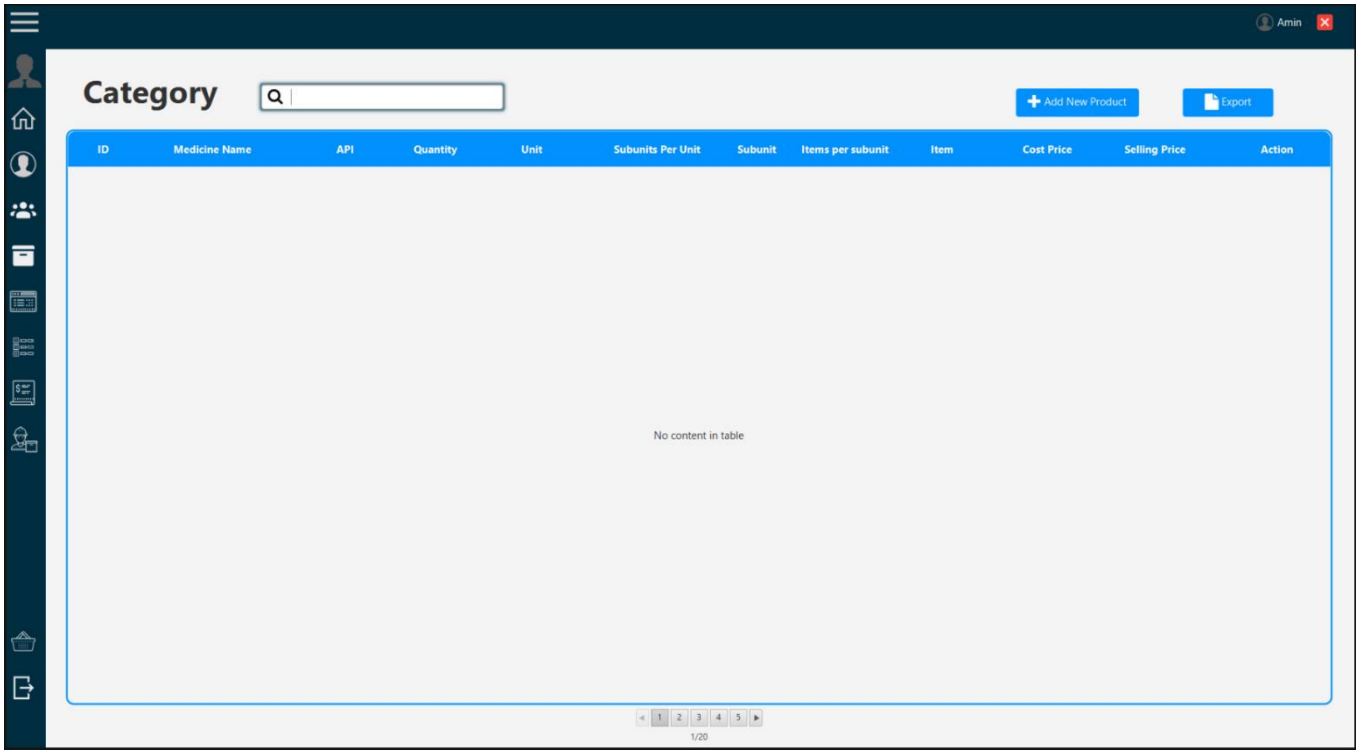
Print Button: Print a doc, excel… file to report

3.1. Supplier

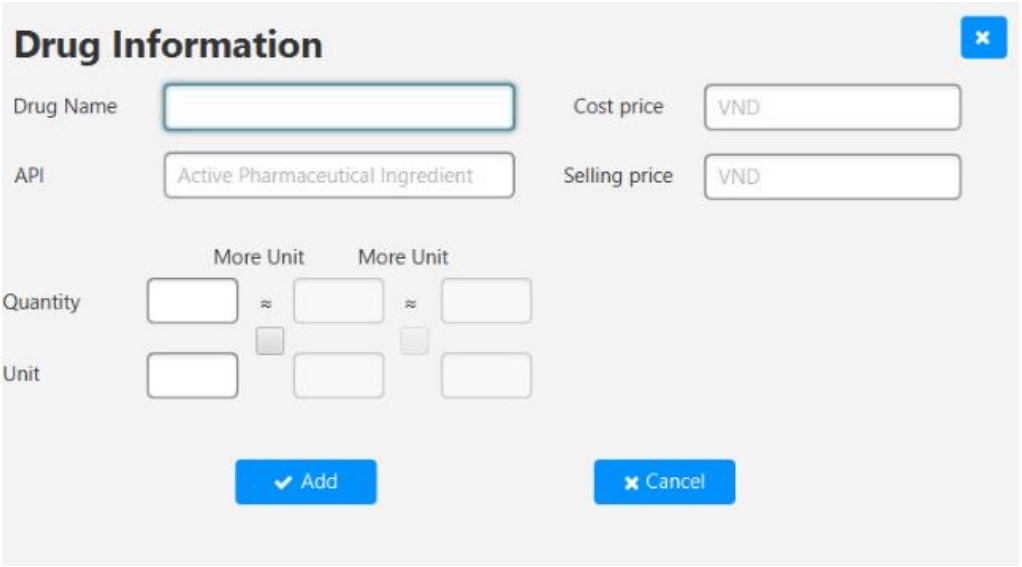


Search:	Search product by Name shown in table below
Table shows up full information:	ID Supplier Name Address Phone Email

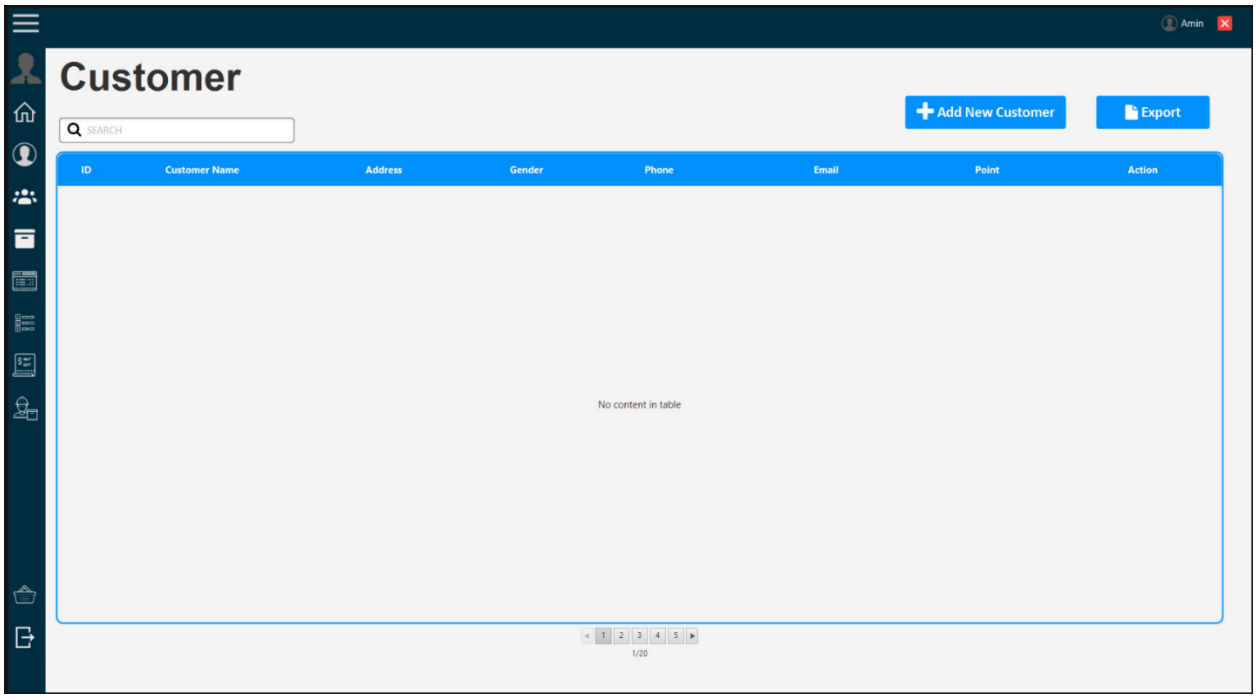
4.1. Category



Search:	Search product by Name shown in table below
Export Button:	Click to go to Export page
Add New Button:	Click to go to Add new form

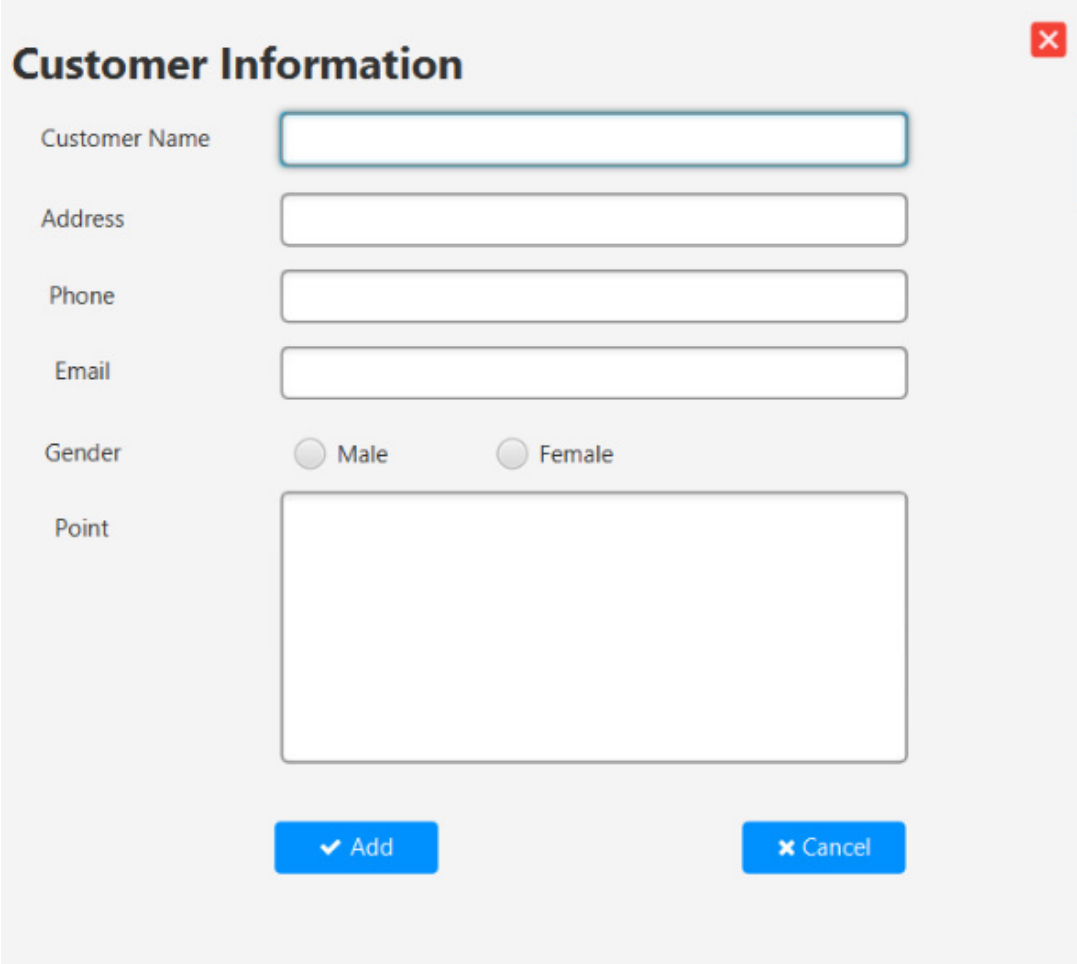


4.2. Customer Management



Search:	Search product by Name shown in table below
Add New Customer Button:	Click to go to Add new form

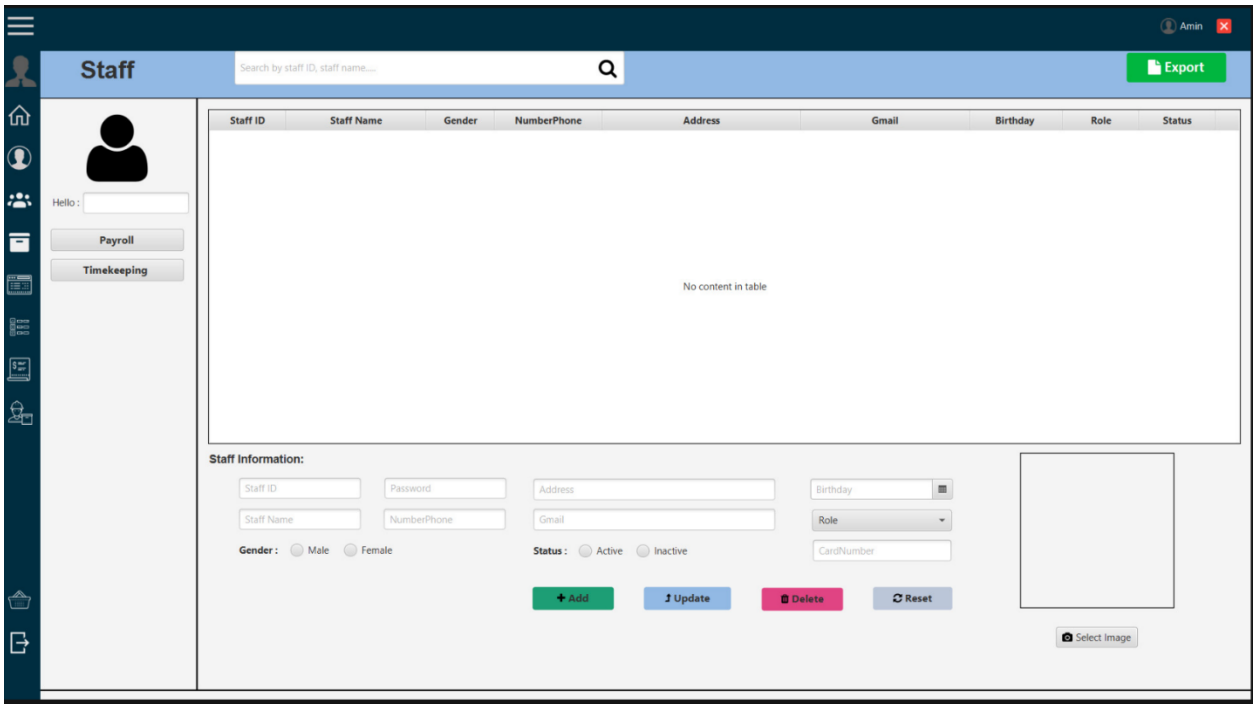
2.7.1.And new customer



Users fill full information below:

+ Text field Customer Name
  Validate: “Name must not be longer than 30 characters”
+ Text field Address
+ Text field integer Phone
  Validate: “Phone number must be entered in numbers”
+ Text field string Email (validate with @)
+ Choice Gender
  Validate: “Gender is not blank”
+ Text field string Note

4.3. Staff Management

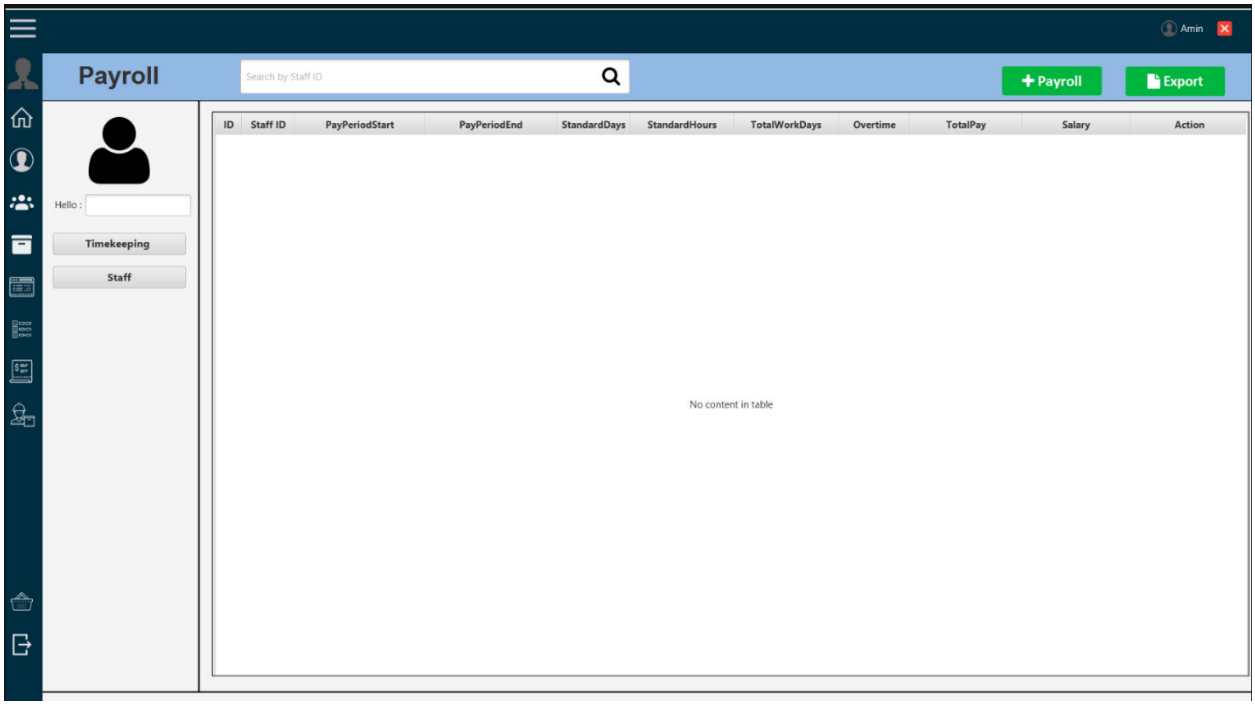


Search: + Search product by Name shown in table below
Payroll Button: Click to go Payroll page
Timekeeping Button: Click to go Timekeeping page
Export Button: Click to go Export page
User will fill full information:

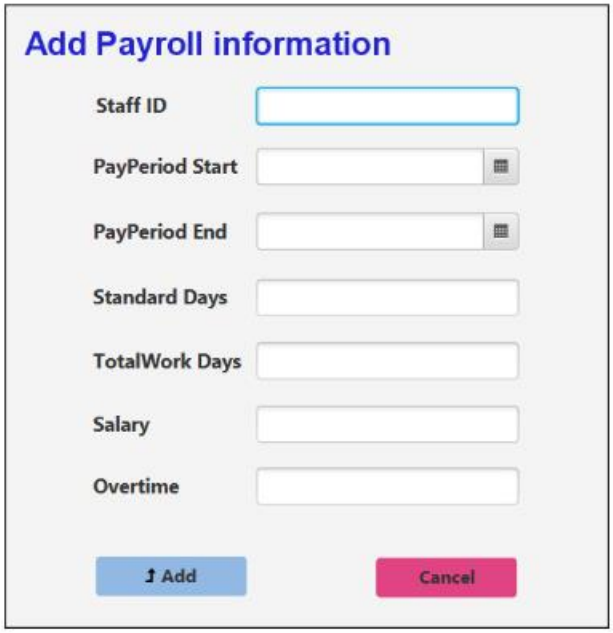
- Integer Staff ID: Validate: “NV*number*”
- String Password Validate: “Password is not blank”
- String Address
- Date picker Birthday
- String Staff Name Validate: “Name shorter than 30 characters”
- Integer Phone Validate: “Enter by number”
- String Gmail Validate with @
- Choice Role: Manager-Staff
- Choice Gender: Male-Female
- Choice Status: Active- Inactive
- Import Picture

Add Button: Complete addition
Update Button: Choose the staff’s information and change it
Delete Button: Delete info is chosen
Reset Button: Reset all text field

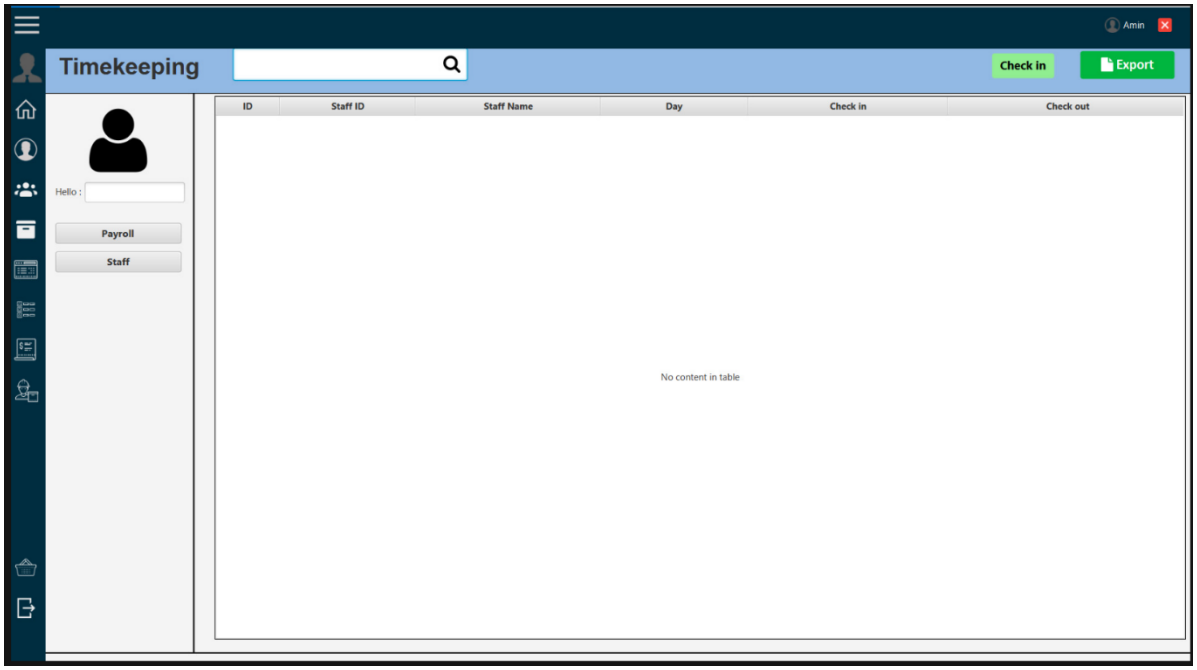
4.4. Payroll page



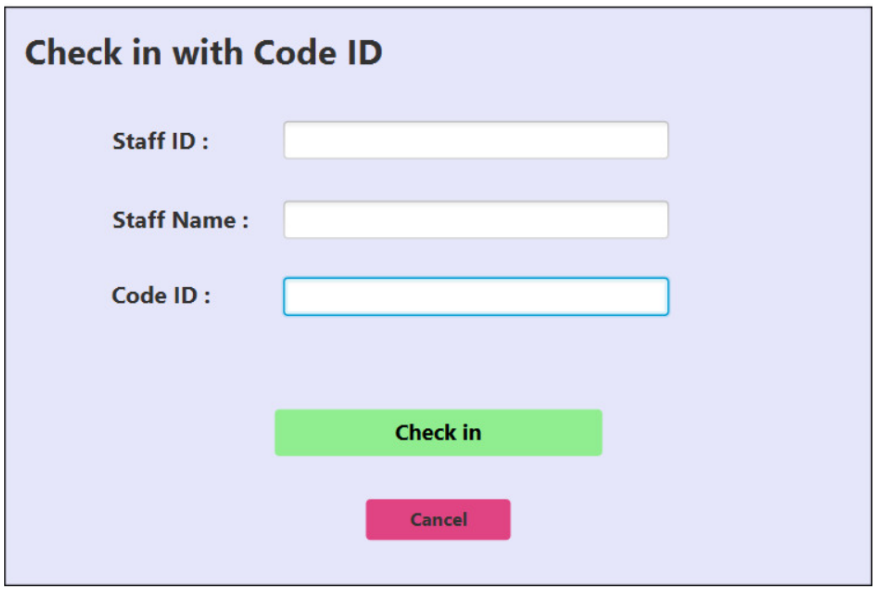
Search: Search product by Name shown in table below
Payroll Button:
Export Button: Click to go Export page
Timekeeping Button: Click to go Timekeeping page
Staff Button: Click to go Staff page



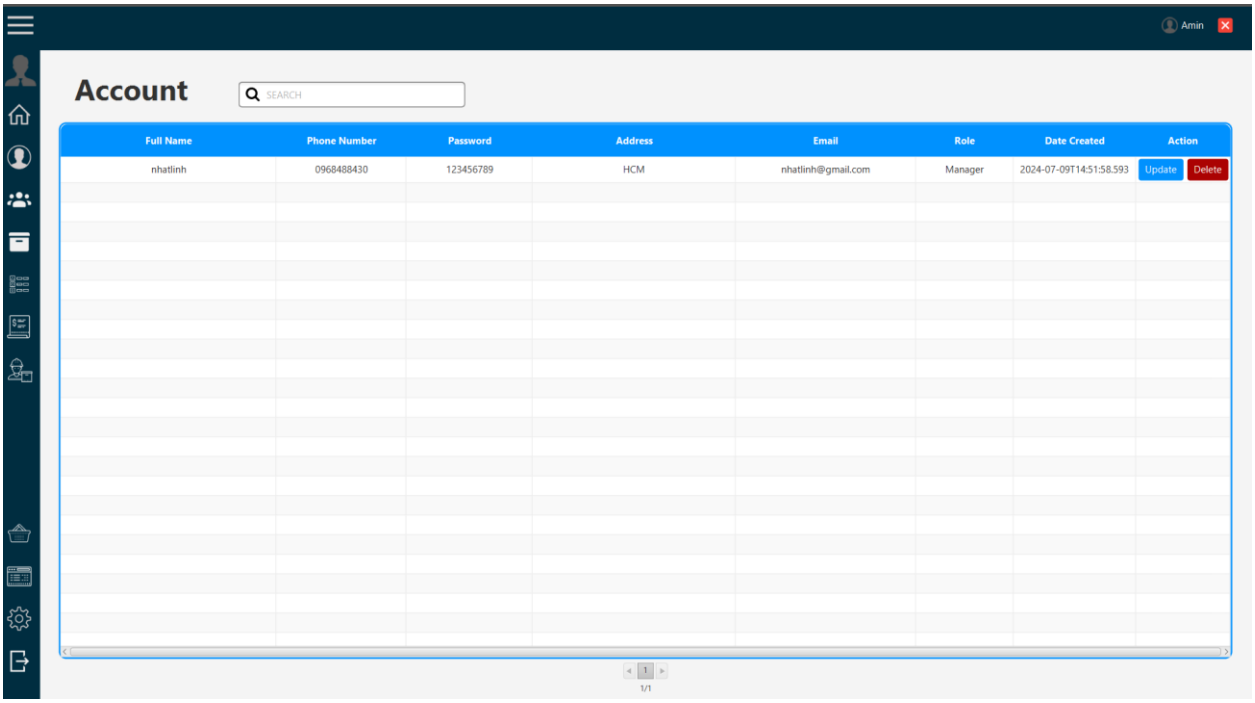
4.5. Time Keeping Page



Search:	Search product by Name shown in table below
Date picker Icon:	Choose the day to show
Export Button:	Click to go Export page
Staff Button:	Click to go Staff page



4.6. Account Management



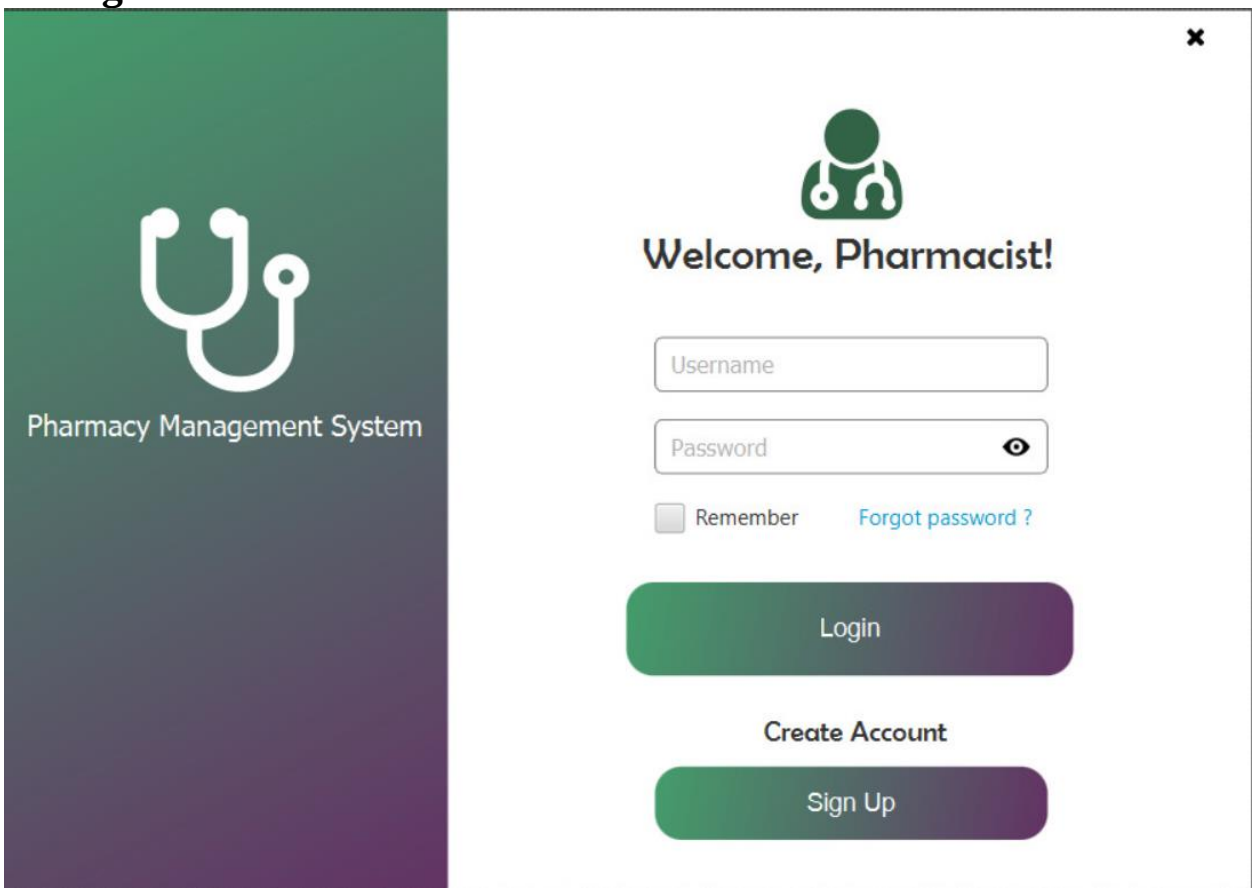
Search:	Search product by Name/Phone/Email shown in table below
Update Button:	Click to Update data selected
Delete Button:	Click to delete data selected

TASK SHEET REVIEW 3

Project: Pharmacy			**Date of preparation of activity plan:**			
#	**Task**	**Prepared by**	**Start date**	**Actuals Date**	**Team member name**	**status**
1	GUI DESIGN	Hoang Gia Huy & Nguyen Anh Minh & Nguyen Anh Quan & Tran Nhat Linh & Doan Duc Do	07/06/2024	17/6/2024	Hoang Gia Huy & Nguyen Anh Minh & Nguyen Anh Quan & Tran Nhat Linh & Doan Duc Do	completed
Review			Signature of instructor			
			Mr. Pham Cong Danh			

Coding Review 1

1. Login Form



1.1. Code

1.1.1. Login Button

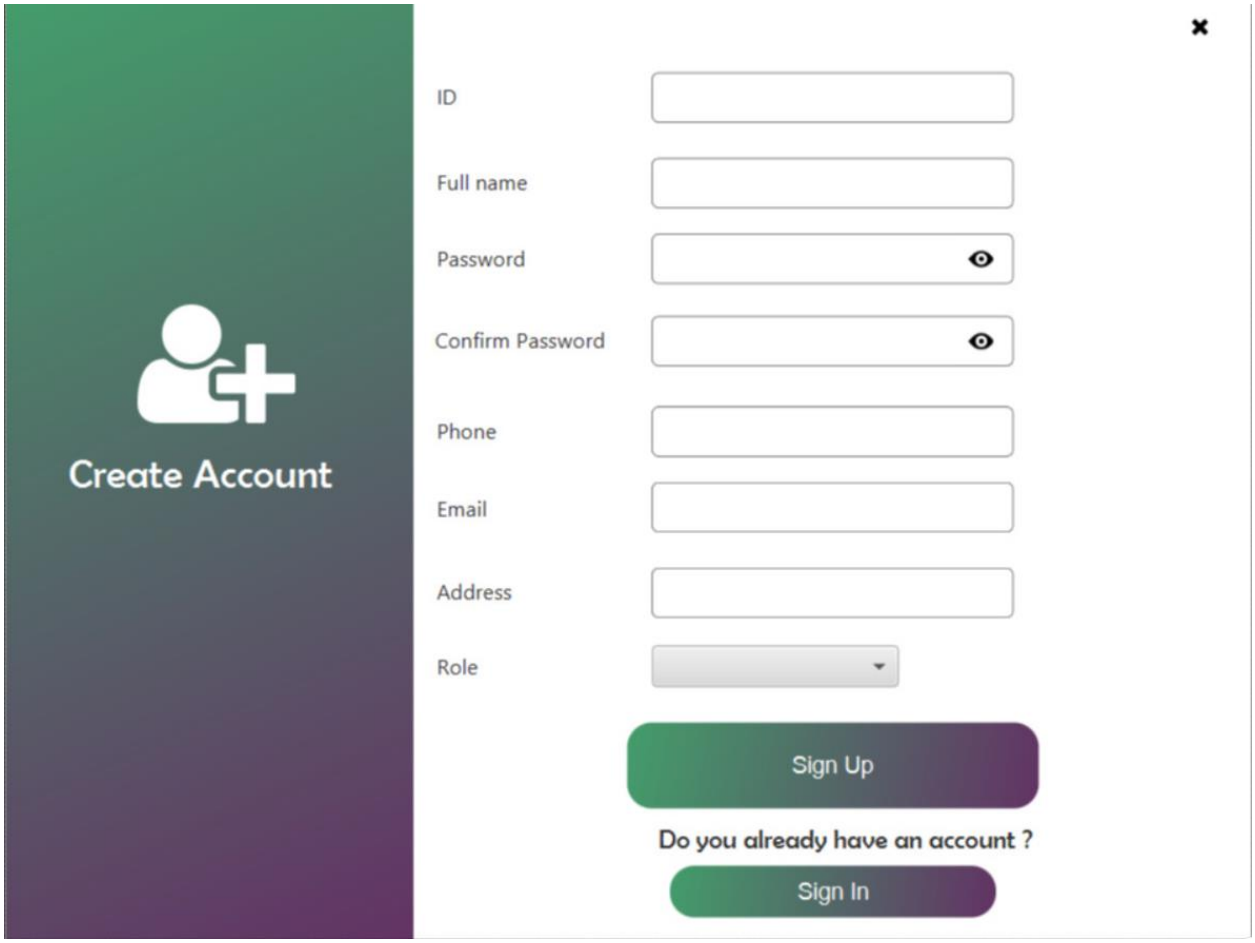
```java
public void login() throws IOException, SQLException {
        try {
            if (username.getText().isEmpty()) {
                AlertHelper.errorMessage("Username cannot be empty");
                return;
            }

            if (passwordField.getText().isEmpty()) {
                AlertHelper.errorMessage("Password cannot be empty");
                return;
            }

            String selectLogin = "SELECT Username FROM Login WHERE Username = ? AND Password = ?";
            conn = DB.ConnectDB.getConnectDB();
            prepare = conn.prepareStatement(selectLogin);
            prepare.setString(1, username.getText());
            prepare.setString(2, passwordField.getText());
            rs = prepare.executeQuery();

            if (rs.next()) {

                String selectRole = "SELECT Role FROM Role WHERE Username = ?";
                prepare = conn.prepareStatement(selectRole);
                prepare.setString(1, username.getText());
                rs = prepare.executeQuery();

                if (rs.next()) {
                    String role = rs.getString("Role");
                    String fxmlFile = "";

                    if (!"Manager".equals(role)) {
                        if ("Staff".equals(role)) {
                            fxmlFile = "/sam/FXMLStaff.fxml";
                        } else {
                            AlertHelper.errorMessage("Unknown role: " + role);
                            return;
                        }
                    } else {
                        fxmlFile = "/sam/FXMLMain.fxml";
                    }

                    String selectUserInfo = "SELECT Fullname FROM Information WHERE Phone = ?";
                    prepare = conn.prepareStatement(selectUserInfo);
                    prepare.setString(1, username.getText());
                    rs = prepare.executeQuery();

                    if (rs.next()) {
                        fullName = rs.getString("Fullname");
                    }

```

```
if (ckremember.isSelected()) {
                    savePassword(username.getText(), passwordField.getText());
                } else {
                    clearRMPassword(username.getText());
                }
                FXMLLoader loader = new FXMLLoader(getClass().getResource(fxmlFile));
                Parent root = loader.load();
                Scene scene = new Scene(root);
                String cssPath = getClass().getResource("/sam/button-style.css").toExternalForm();
                scene.getStylesheets().add(cssPath);

                Stage stage = (Stage) username.getScene().getWindow(); // Get the current stage

                stage.setResizable(true);

                if ("Staff".equals(role)) {
                    FXMLStaffController staffController = loader.getController();
                    staffController.setFullName(fullName);
                } else {
                    FXMLMainController mainController = loader.getController();
                    mainController.setFullName(fullName);
                }

                Sam appInstance = new Sam();
                appInstance.makeWindowDraggable(stage, root);

                stage.setScene(scene);
                stage.centerOnScreen();
                stage.show();
            } else {
                AlertHelper.errorMessage("Role not found for the username");
            }
        } else {
            AlertHelper.errorMessage("Incorrect username or password");
        }
    } catch (SQLException e) {
        AlertHelper.errorMessage("Database error: " + e.getMessage());
    } finally {
        closeResources(rs, prepare, conn);
    }
}
```

1.2. Validate

```java
public class Validate {

    public boolean checkAccountExists(String username) {
        String sql = "SELECT COUNT(*) FROM users WHERE username = ?";
        try (
            Connection conn = DB.ConnectDB.getConnectDB();
            PreparedStatement stmt = conn.prepareStatement(sql)
        ) {
            stmt.setString(1, username);
            ResultSet rs = stmt.executeQuery();
            if (rs.next()) {
                int count = rs.getInt(1);
                return count > 0;
            }
        } catch (SQLException e) {
        }
        return false;
    }


    public boolean checkPassword(String username, String password) {
        String sql = "SELECT password FROM users WHERE username = ?";
        try (
            Connection conn = DB.ConnectDB.getConnectDB();
            PreparedStatement stmt = conn.prepareStatement(sql)
        ) {
            stmt.setString(1, username);
            ResultSet rs = stmt.executeQuery();
            if (rs.next()) {
                String storedPassword = rs.getString("password");
                return storedPassword.equals(password);
            }
        } catch (SQLException e) {
         }
        return false;
    }
```

2. Sign-up Form



2.1. Code

2.1.1. Import

```java
@FXML
    private ImageView EYE;

    @FXML
    private ImageView EYE_SLASH;

    @FXML
    private Button btnEYE;

    @FXML
    private Button btnEYE_SLASH;

    @FXML
    private ImageView EYECF;

    @FXML
    private ImageView EYE_SLASHCF;

    @FXML
    private Button btnEYECF;

    @FXML
    private Button btnEYE_SLASHCF;

    @FXML
    private Button btnSignin;

    @FXML
    private Button btnSignup;

    @FXML
    private ComboBox<String> cbbRole;

    @FXML
    private Button close;
```

```java
@FXML
    private TextField txtFullname;

    @FXML
    private TextField txtAddress;

    @FXML
    private TextField txtEmail;

    @FXML
    private PasswordField txtPassword;

    @FXML
    private PasswordField txtCFPassword;

    @FXML
    private TextField txtShowPassword;

    @FXML
    private TextField txtshowCFPassword;

    @FXML
    private TextField txtPhone;

    @FXML
    private TextField txtID;

    @FXML

    private boolean isPasswordVisible = false;
    private boolean isCFPasswordVisible = false;

    private Connection conn;
    private PreparedStatement prepare;
```

2.1.2. Get and Save Data

```java
public void saveUserData() {
        String password = txtPassword.getText();

        String email = txtEmail.getText();
        String phone = txtPhone.getText();
        String address = txtAddress.getText();
        String fullname = txtFullname.getText();
        String role = cbbRole.getValue();
        String id;
        if (isBarcodeScanned()) {
            id = getScannedBarcode();

        } else {
            id = txtID.getText();

        }

        Timestamp dateCreated = Timestamp.from(Instant.now());

        try {
            conn = DB.ConnectDB.getConnectDB();

            String sqlInformation = "INSERT INTO Information (ID, Fullname, Phone, Email, Address, Date_Created) VALUES (?, ?, ?, ?, ?, ?)";
            prepare = conn.prepareStatement(sqlInformation);
            prepare.setString(1, id);
            prepare.setString(2, fullname);
            prepare.setString(3, phone);
            prepare.setString(4, email);
            prepare.setString(5, address);
            prepare.setTimestamp(6, dateCreated);
            prepare.executeUpdate();


            String sqlRole = "INSERT INTO Role (Role, Username) VALUES (?, ?)";
            prepare = conn.prepareStatement(sqlRole);
            prepare.setString(1, role);
            prepare.setString(2, phone);
            prepare.executeUpdate();


            String sqlLogin = "INSERT INTO Login (Username, Password) VALUES (?, ?)";
            prepare = conn.prepareStatement(sqlLogin);
            prepare.setString(1, phone);
            prepare.setString(2, password);
            prepare.executeUpdate();

            System.out.println("Data saved successfully!");


            showAlert("SuccessFully", "Congratulation!!");
            ReturntoLogin();
        } catch (SQLException e) {

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

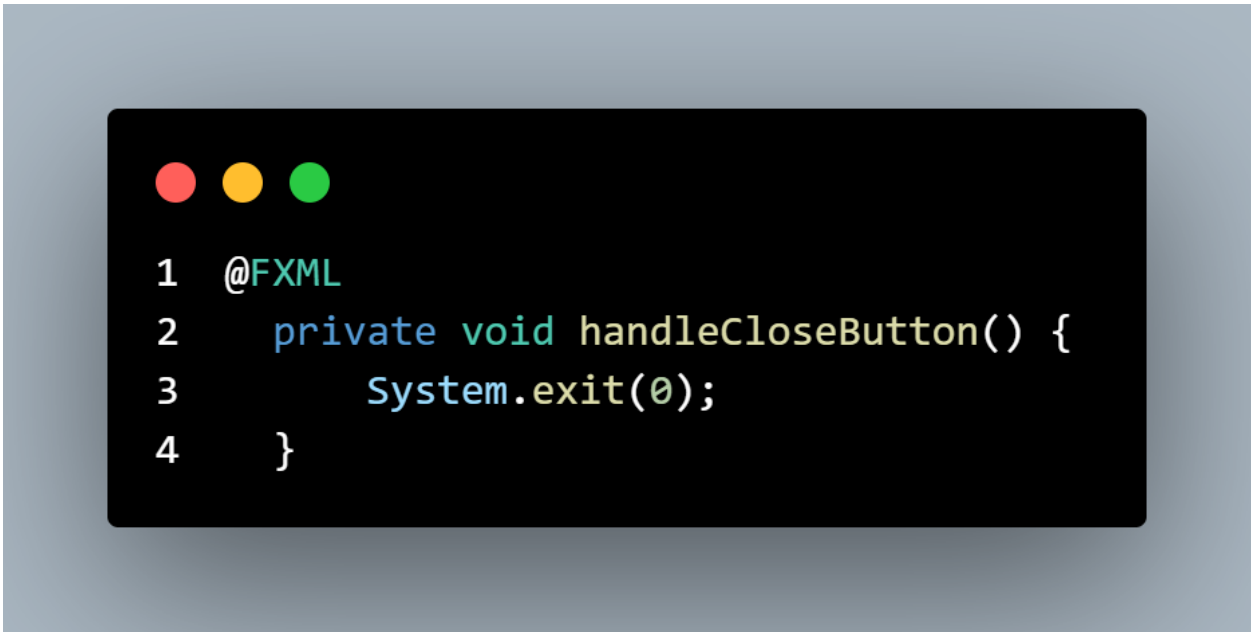
2.1.3. Select Role

```
@FXML
    private ComboBox<String> cbbRole;
@Override
    public void initialize(URL url, ResourceBundle rb) {
    conn = DB.ConnectDB.getConnectDB();
        cbbRole.getItems().addAll("Staff", "Manager");
        cbbRole.setValue("Staff");
    }
    public void saveUserData() {
        String role = cbbRole.getValue();
    }
```

2.2. Validate

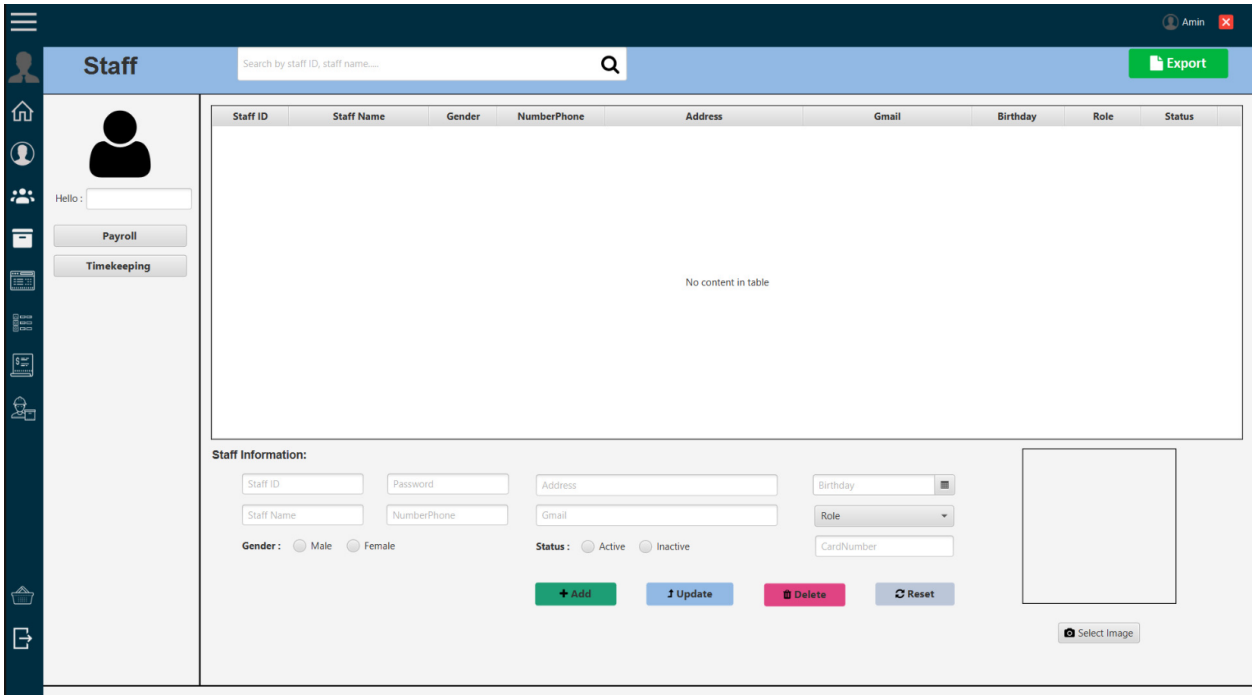
```
if (!ValidateRegister.isNotEmpty(txtID.getText())) {
            showAlert("Validation Error", "ID cannot be empty");
            return;
        }
        if (isIdAlreadyExists(txtID.getText())) {
            showAlert("Validation Error", "ID already exists in the database");
            return;
        }

        if (!ValidateRegister.isNotEmpty(txtFullname.getText())) {
            showAlert("Validation Error", "Fullname cannot be empty");
            return;
        }

        if (!ValidateRegister.isNotEmpty(txtPassword.getText())) {
            showAlert("Validation Error", "Password cannot be empty");
            return;
        }

        if (!ValidateRegister.isPasswordConfirmed(txtPassword.getText(), txtCFPassword.getText())) {
            showAlert("Validation Error", "Confirm Password must match Password and cannot be empty");
            return;
        }

        if (!ValidateRegister.isNotEmpty(txtPhone.getText())) {
            showAlert("Validation Error", "Phone cannot be empty");
            return;
        }

        if (!ValidateRegister.isNumeric(txtPhone.getText())) {
            showAlert("Validation Error", "Phone must be a number");
            return;
        }
        if (isPhoneAlreadyExists(txtPhone.getText())) {
            showAlert("Validation Error", "Phone already exists in the database");
            return;
        }



        if (!ValidateRegister.isValidEmail(txtEmail.getText())) {
            showAlert("Validation Error", "Email is not valid");
            return;
        }

        if (isEmailAlreadyExists(txtEmail.getText())) {
            showAlert("Validation Error", "Email already exists in the database");
            return;
        }
        if (!ValidateRegister.isNotEmpty(txtAddress.getText())) {
            showAlert("Validation Error", "Address cannot be empty");
            return;
        }
        public class ValidateRegister {

        public static boolean isNotEmpty(String input) {
            return input != null && !input.trim().isEmpty();
        }

        public static boolean isPasswordConfirmed(String password, String confirmPassword) {
            return isNotEmpty(password) && password.equals(confirmPassword);
        }

        public static boolean isNumeric(String input) {
            return input != null && input.matches("\\d+");
        }

        public static boolean isValidEmail(String email) {
            String emailRegex = "^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,}$";
            Pattern pat = Pattern.compile(emailRegex);
            return email != null && pat.matcher(email).matches();
        }
    }
```

3. Close Button



```
@FXML
  private void handleCloseButton() {
      System.exit(0);
  }
```

4. Staff Form



4.1. Code

4.1.1. Import

4.1.2. Add Button

```
public void ThemStaff() {
        Thuvien tv = new Thuvien();
        if (txtmanv.getText().isEmpty() || txtCardNumber.getText().isEmpty() || txttennv.getText().isEmpty() || txtsdt.getText().isEmpty()
                || txtdiachi.getText().isEmpty() || txtemail.getText().isEmpty() || datengaysinh.getvalue() == null
                || comboboxRole.getSelectionModel().getSelectedItem() == null) {
            tv.showAlert("Please enter complete information");
        } else if (!txtmanv.getText().matches("^NV\\d+$")) {
            tv.showAlert("StaffID must start with 'NV' followed by numbers!");
        } else if (!txtpass.getText().matches("^(?=.*[A-Z])(?=.*[a-zA-Z])(?=.*\\d)[A-Za-z\\d]{8,}$")) {
            tv.showAlert("Password must be at least 8 characters long, start with an uppercase letter, and include both letters and numbers!");
        } else if (!txtsdt.getText().matches("^\\d{1,11}$")) {
            tv.showAlert("Phone number must contain only digits and be up to 11 digits long!");
        } else if (!(rbNam.isSelected() ^ rbNu.isSelected())) {
            tv.showAlert("Please select either 'Male' or 'Female'!");
        } else if (!(rbActive.isSelected() ^ rbInactive.isSelected())) {
            tv.showAlert("Please select either 'Active' or 'InActive'!");
        } else if (!txtemail.getText().matches("^[\\w.-]+@gmail\\.com$")) {
            tv.showAlert("Email must be in the format 'example@gmail.com'!");
        } else {
            String checkStaffIDSQL = "SELECT COUNT(*) AS count FROM tblStaff WHERE StaffID = ?";
            String checkGmailSQL = "SELECT COUNT(*) AS count FROM tblStaff WHERE Gmail = ?";
            String checkPhoneSQL = "SELECT COUNT(*) AS count FROM tblStaff WHERE Numberphone = ?";
            String checkCardNumberSQL = "SELECT COUNT(*) AS count FROM tblStaff WHERE CardNumber = ?";
            try {
                conn = connect.getConnectDB();
                pst = conn.prepareStatement(checkStaffIDSQL);
                pst.setString(1, txtmanv.getText());
                ResultSet rs = pst.executeQuery();
                if (rs.next() && rs.getInt("count") > 0) {
                tv.showAlert("StaffID already exists. Please enter a unique StaffID.");
                    return;
                }
                pst = conn.prepareStatement(checkGmailSQL);
                pst.setString(1, txtemail.getText());
                rs = pst.executeQuery();
                if (rs.next() && rs.getInt("count") > 0) {
                    tv.showAlert("Gmail already exists. Please enter a unique Gmail.");
                    return;
                }
                pst = conn.prepareStatement(checkPhoneSQL);
                pst.setString(1, txtsdt.getText());
                rs = pst.executeQuery();
                if (rs.next() && rs.getInt("count") > 0) {
                    tv.showAlert("Phone number already exists. Please enter a unique phone number.");
                    return;
                }
                pst = conn.prepareStatement(checkCardNumberSQL);
                pst.setString(1, txtCardNumber.getText());
                rs = pst.executeQuery();
                if (rs.next() && rs.getInt("count") > 0) {
                    tv.showAlert("CardNumber already exists. Please enter a unique CardNumber.");
                    return;
                }
            } catch (Exception e) {
                System.out.println("Error while checking StaffID: " + e.getMessage());
                return;
            }

            String sql = "INSERT INTO tblStaff (StaffID, Staffname, Gender, Numberphone, Address, Gmail, Birthday, Role, Status, Image, Password,CardNumber) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,?)";
            try {
                pst = conn.prepareStatement(sql);

                pst.setString(1, txtmanv.getText());
                pst.setString(2, txttennv.getText());

                String gender = rbNam.isSelected() ? "Male" : "Female";
                pst.setString(3, gender);

                pst.setString(4, txtsdt.getText());
                pst.setString(5, txtdiachi.getText());
                pst.setString(6, txtemail.getText());


                pst.setDate(7, java.sql.Date.valueOf(datengaysinh.getValue()));

                pst.setString(8, (String) comboboxRole.getSelectionModel().getSelectedItem());

                String status = rbActive.isSelected() ? "Active" : "InActive";
                pst.setString(9, status);

                pst.setString(10, file_path.getText());
                pst.setString(11, txtpass.getText());
                pst.setString(12, txtCardNumber.getText());
                pst.executeUpdate();
                tv.showAlert("Staff added successfully!");
                showData();
                resetData();
            } catch (Exception e) {
                System.out.println(e.getMessage());
            }
        }
    }
```

4.1.3. Update Button

```
public void updateData() {
        TableNhanvien selectedData = tablenhanvien.getSelectionModel().getSelectedItem();

        if (selectedData != null) {
            Thuvien tv = new Thuvien();

            if (txtmanv.getText().isEmpty() || txtCardNumber.getText().isEmpty() || txttennv.getText().isEmpty() || txtsdt.getText().isEmpty()
                    || txtdiachi.getText().isEmpty() || txtemail.getText().isEmpty() || datengaysinh.getValue() == null
                    || comboboxRole.getSelectionModel().getSelectedItem() == null) {

                tv.showAlert("Please enter complete information");
            } else if (!txtmanv.getText().equals(selectedData.getStaffID())) {
                tv.showAlert("Cannot update StaffID.");
            } else if (!txtpass.getText().equals(selectedData.getPassword())) {
                tv.showAlert("Cannot update Password.");
            } else if (!txtmanv.getText().matches("^NV\\d+$")) {
                tv.showAlert("StaffID must start with 'NV' followed by numbers!");
            } else if (!txtpass.getText().matches("^(?=.*[A-Z])(?=.*[a-zA-Z])(?=.*\\d)[A-Za-z\\d]{8,}$")) {
                tv.showAlert("Password must be at least 8 characters long, start with an uppercase letter, and include both letters and numbers!");
            } else if (!txtsdt.getText().matches("^\\d{1,11}$")) {
                tv.showAlert("Phone number must contain only digits and be up to 11 digits long!");
            } else if (!(rbNam.isSelected() ^ rbNu.isSelected())) {
                tv.showAlert("Please select either 'Male' or 'Female'!");
            } else if (!(rbActive.isSelected() ^ rbInactive.isSelected())) {
                tv.showAlert("Please select either 'Active' or 'InActive'!");
            } else if (!txtemail.getText().matches("^[\\w.-]+@gmail\\.com$")) {
                tv.showAlert("Email must be in the format 'example@gmail.com'!");
            } else {
                String checkGmailSQL = "SELECT COUNT(*) AS count FROM tblStaff WHERE Gmail = ? AND StaffID != ?";
                String checkPhoneSQL = "SELECT COUNT(*) AS count FROM tblStaff WHERE Numberphone = ? AND StaffID != ?";
                String checkCardNumberSQL = "SELECT COUNT(*) AS count FROM tblStaff WHERE CardNumber = ? AND StaffID != ?";

                try {
                    conn = connect.getConnectDB();


                    pst = conn.prepareStatement(checkGmailSQL);
                    pst.setString(1, txtemail.getText());
                    pst.setString(2, selectedData.getStaffID());
                    rs = pst.executeQuery();
                    if (rs.next() && rs.getInt("count") > 0) {
                        tv.showAlert("Gmail already exists. Please enter a unique Gmail.");
                        return;
                    }


                    pst = conn.prepareStatement(checkPhoneSQL);
                    pst.setString(1, txtsdt.getText());
                    pst.setString(2, selectedData.getStaffID());
                    rs = pst.executeQuery();
                    if (rs.next() && rs.getInt("count") > 0) {
                        tv.showAlert("Phone number already exists. Please enter a unique phone number.");
                        return;
                    }
                    pst = conn.prepareStatement(checkCardNumberSQL);
                    pst.setString(1, txtCardNumber.getText());
                    pst.setString(2, selectedData.getStaffID());
                    rs = pst.executeQuery();
                    if (rs.next() && rs.getInt("count") > 0) {
                        tv.showAlert("CardNumber already exists. Please enter a unique CardNumber.");
                        return;
                    }

                    String sql = "UPDATE tblStaff SET Staffname = ?, Gender = ?, Numberphone = ?, Address = ?, Gmail = ?, Birthday = ?, Role = ?, Status = ?, Image = ? , CardNumber = ? WHERE StaffID = ?";
                    pst = conn.prepareStatement(sql);

                    pst.setString(1, txttennv.getText());

                    String gender = rbNam.isSelected() ? "Male" : "Female";
                    pst.setString(2, gender);

                    pst.setString(3, txtsdt.getText());
                    pst.setString(4, txtdiachi.getText());
                    pst.setString(5, txtemail.getText());
                    pst.setDate(6, java.sql.Date.valueOf(datengaysinh.getValue()));
                    pst.setString(7, (String) comboboxRole.getSelectionModel().getSelectedItem());
                    String status = rbActive.isSelected() ? "Active" : "Inactive";
                    pst.setString(8, status);
                    pst.setString(9, file_path.getText());
                    pst.setString(10, txtCardNumber.getText());
                    pst.setString(11, selectedData.getStaffID());
                    int rowsAffected = pst.executeUpdate();
                    if (rowsAffected > 0) {
                        tv.showAlert("Data updated successfully!");

                        showData();
                        resetData();
                    } else {
                        tv.showAlert("No data found to update!");
                    }
                } catch (Exception e) {
                    System.out.println("Error while updating data: " + e.getMessage());
                } finally {
                    try {
                        if (rs != null) {
                            rs.close();
                        }
                        if (pst != null) {
                            pst.close();
                        }
                        if (conn != null) {
                            conn.close();
                        }
                    } catch (Exception e) {
                        System.out.println("Error while closing connection: " + e.getMessage());
                    }
}
            }
        } else {
            Thuvien tv = new Thuvien();
            tv.showAlert("Please select a row to update.");
        }
    }
```

4.1.4. Delete Button

```
public void deleteData() {
        Thuvien tv = new Thuvien();
        TableNhanvien selectedData = tablenhanvien.getSelectionModel().getSelectedItem();

        if (selectedData != null) {
            String staffID = selectedData.getStaffID();

            Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
            alert.setTitle("Confirmation Dialog");
            alert.setHeaderText(null);
            alert.setContentText("Are you sure you want to delete this employee's information?");

            ButtonType buttonTypeOK = new ButtonType("OK");
            ButtonType buttonTypeCancel = new ButtonType("Cancel", ButtonBar.ButtonData.CANCEL_CLOSE);
            alert.getButtonTypes().setAll(buttonTypeOK, buttonTypeCancel);

            Optional<ButtonType> result = alert.showAndWait();
            if (result.isPresent() && result.get() == buttonTypeOK) {
                String sqlCheck = "SELECT * FROM tblPayroll WHERE StaffID = ?";

                try {
                    conn = connect.getConnectDB();
                    pst = conn.prepareStatement(sqlCheck);
                    pst.setString(1, staffID);
                    rs = pst.executeQuery();
                    if (rs.next()) {
                    tv.showAlert("Cannot delete staff. Please delete payroll data first.");
                    } else {
                        String sqlDeleteStaff = "DELETE FROM tblStaff WHERE StaffID = ?";
                        pst = conn.prepareStatement(sqlDeleteStaff);
                        pst.setString(1, staffID);

                        int rowsAffected = pst.executeUpdate();

                        if (rowsAffected > 0) {
                            tv.showAlert("Data deleted successfully!");

                            showData();
                            resetData();
                        } else {
                            tv.showAlert("No data found to delete!");
                        }
                    }
                } catch (Exception e) {
                    System.out.println("Error while deleting data: " + e.getMessage());
                } finally {
                    try {
                        if (rs != null) {
                            rs.close();
                        }
                        if (pst != null) {
                            pst.close();
                        }
                        if (conn != null) {
                            conn.close();
                        }
                    } catch (Exception ex) {
                        System.out.println("Error while closing resources: " + ex.getMessage());
                    }
                }
            } else {
                tv.showAlert("Delete operation cancelled.");
            }
        } else {
            tv.showAlert("Please select a row to delete.");
        }
    }
```

4.1.5. Import Image

```java
public void insertImage() {
        FileChooser open = new FileChooser();
        Stage stage = (Stage) top_main.getScene().getWindow();
        File file = open.showOpenDialog(stage);
        if (file != null) {
            String path = file.getAbsolutePath();
            path = path.replace("\\", "\\\\");
            file_path.setText(path);
            Image image = new Image(file.toURI().toString(), 110, 110, false, true);
            imageview.setImage(image);

        } else {

            System.out.println("NO DATA EXIST!");
        }
    }

```

4.2. Show date to table view

```
public void showData() {

        ObservableList<TableNhanvien> showList = dataList();

        cotstaffid.setCellValueFactory(new PropertyValueFactory<>("StaffID"));

        cotstaffname.setCellValueFactory(new PropertyValueFactory<>("Staffname"));

        cotgender.setCellValueFactory(new PropertyValueFactory<>("Gender"));
        cotsdt.setCellValueFactory(new PropertyValueFactory<>("Numberphone"));

        cotaddress.setCellValueFactory(new PropertyValueFactory<>("Address"));

        cotemail.setCellValueFactory(new PropertyValueFactory<>("Gmail"));

        cotbirthday.setCellValueFactory(new PropertyValueFactory<>("Birthday"));

        cotrole.setCellValueFactory(new PropertyValueFactory<>("Role"));

        cotstatus.setCellValueFactory(new PropertyValueFactory<>("Status"));

        tablenhanvien.setItems(showList);

    }
```

5. **Pagination**


```java
@FXML
    private Pagination pgtDL;

private static final int itemsperpage = 23;
    private void updateTable(int pageIndex) {
        int fromIndex = pageIndex * itemsperpage;
        int toIndex = Math.min(fromIndex + itemsperpage, DataList().size());
        tabledruglist.setItems(FXCollections.observableArrayList(DataList().subList(fromIndex, toIndex)));
    }
    private int getPageCount() {
        return (int) Math.ceil((double) DataList().size() / itemsperpage);
    }
    private Node createPage(int pageIndex) {
        updateTable(pageIndex);
        return tabledruglist;
    }

    public void Pagination() {
        pgtDL.setPageCount(getPageCount());
        pgtDL.setPageFactory(this::createPage);
    }

    @Override
    public void initialize(URL url, ResourceBundle rb) {
        Pagination();
    }

```

Coding Review 2

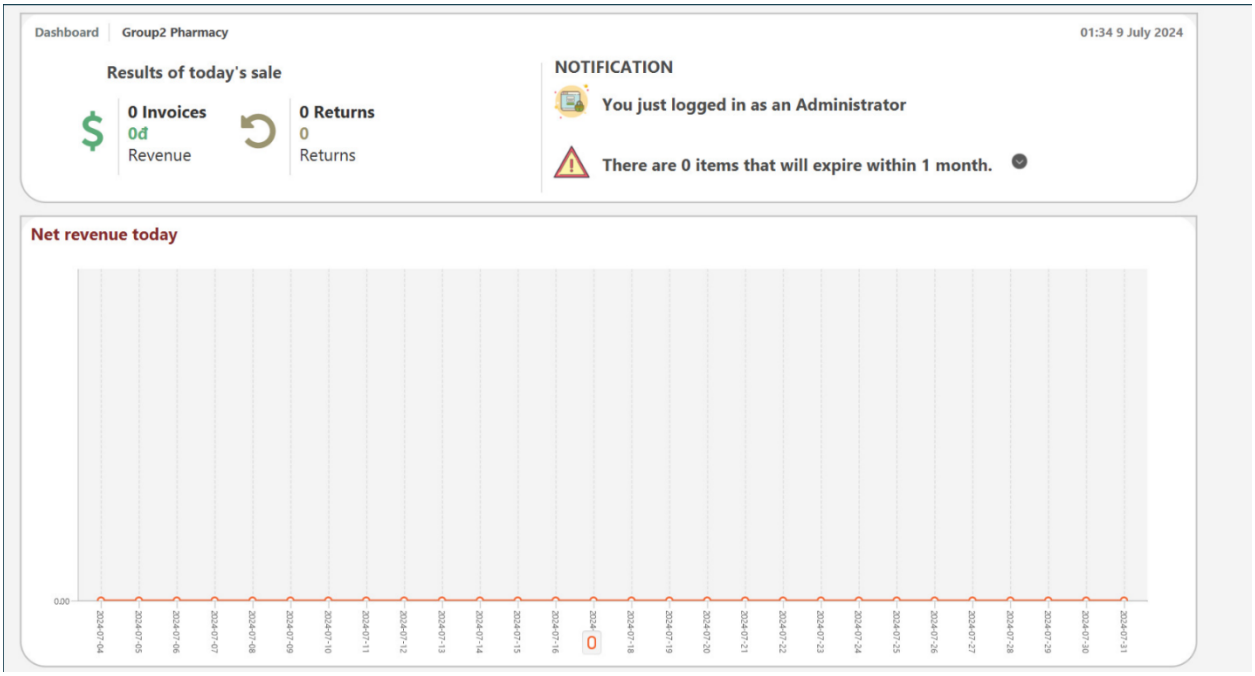
1. Remember Password Button

1.1. Code


```java
private void resetRemember() {
        if (isPasswordRemembered()) {
            ckremember.setSelected(true);
        } else {
            ckremember.setSelected(false);
        }
    }

    private boolean isPasswordRemembered() {
        try (Connection conn = DB.ConnectDB.getConnectDB()) {
            if (conn == null) {
                System.out.println("Connection is null. Cannot check password.");
                return false;
            }

            String usernameText = username.getText();
            if (usernameText == null || usernameText.isEmpty()) {
                System.out.println("Username field is empty.");
                return false;
            }

            try (PreparedStatement selectStmt = conn.prepareStatement(SELECT_RMBPASSWORD_SQL)) {
                selectStmt.setString(1, usernameText);
                try (ResultSet rs = selectStmt.executeQuery()) {
                    if (rs.next()) {
                        String rmbPasswordText = rs.getString("RmbPassword");
                        return rmbPasswordText != null && !rmbPasswordText.isEmpty();
                    }
                }
            }
        } catch (SQLException e) {
            System.err.println("SQL error: " + e.getMessage());
        }
        return false;
    }

    private void savePassword(String username, String password) {
        try (Connection conn = DB.ConnectDB.getConnectDB()) {
            if (conn == null) {
                System.out.println("Connection is null. Cannot save password.");
                return;
            }

            String query = UPDATE_RMBPASSWORD_SQL;
            try (PreparedStatement stmt = conn.prepareStatement(query)) {
                stmt.setString(1, password);
                stmt.setString(2, username);
                int rowsUpdated = stmt.executeUpdate();
                if (rowsUpdated > 0) {
                    System.out.println("Password remembered successfully.");
                } else {
                    System.out.println("No user found with the specified username.");
                }
            }
        } catch (SQLException e) {
            System.out.println("SQL Exception: " + e.getMessage());
        }
    }
```

```java
private void clearRMPassword(String username) {
        try (Connection conn =DB.ConnectDB.getConnectDB()) {
            if (conn == null) {
                System.out.println("Connection is null. Cannot clear password.");
                return;
            }

            String query = "UPDATE Login SET RmbPassword = NULL WHERE Username = ?";
            try (PreparedStatement stmt = conn.prepareStatement(query)) {
                stmt.setString(1, username);
                int rowsUpdated = stmt.executeUpdate();
                if (rowsUpdated > 0) {
                    System.out.println("Password cleared successfully.");
                } else {
                    System.out.println("No user found with the specified username.");
                }
            }
        } catch (SQLException e) {
            System.out.println("SQL Exception: " + e.getMessage());
        }
    }

    private void loadPassword() {
        try (Connection conn = DB.ConnectDB.getConnectDB()) {
            if (conn == null) {
                System.out.println("Connection is null. Cannot load password.");
                return;
            }

            String usernameText = username.getText();
            if (usernameText == null || usernameText.isEmpty()) {
                System.out.println("Username field is empty.");
                return;
            }

            try (PreparedStatement selectStmt = conn.prepareStatement(SELECT_RMBPASSWORD_SQL)) {
                selectStmt.setString(1, usernameText);
                try (ResultSet rs = selectStmt.executeQuery()) {
                    if (rs.next()) {
                        String rmbPasswordText = rs.getString("rmbpassword");
                        passwordField.setText(rmbPasswordText);
                        showpasswordField.setText(rmbPasswordText);
                        ckremember.setSelected(true);
                    }
                }
            }
        } catch (SQLException e) {
            System.err.println("SQL error: " + e.getMessage());
        }
    }
```

2. Dashboard without Tabbar (Both Manager/Staff)



2.1. Import

```
private CustomerController customerController;
  private OrderTransactionController orderTransactionController;
  private boolean drawerImageOpen = false;
  @FXML
  private Label Time;
  private Timeline timeline;
  @FXML
  private Label expiryCountLabel;
  @FXML
  private Button btnLogOut;
  @FXML
  private Button admin;
  @FXML
  private Label totalBillsLabel;
  @FXML
  private Label UserNameLB;
  @FXML
  private Label totalRevenueLabel;
  @FXML
  private Button drawerImage;
  @FXML
  private Button Selling;
  @FXML
  private AnchorPane drawerPane;
  @FXML
  private AnchorPane main;
  private LoginController loginController;
  private BillsController billsController;
  @FXML
  private AnchorPane opacityPane;

  @FXML
  private ImageView exit;
```

2.2. Update Invoices

```java
public void updateLabels() {
        billsController.updateTotalBillsAndRevenue();
        int totalBills = billsController.getTotalBills();
        int totalRevenue = billsController.getTotalRevenue();
        totalBillsLabel.setText(String.format("%d Invoices", totalBills));
        totalRevenueLabel.setText(String.format("%dđ", totalRevenue));
    }

```

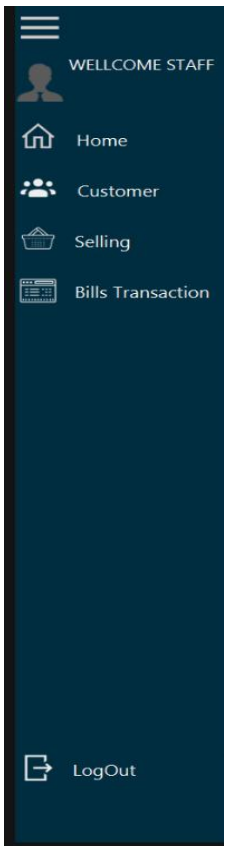
2.3. Bar chart

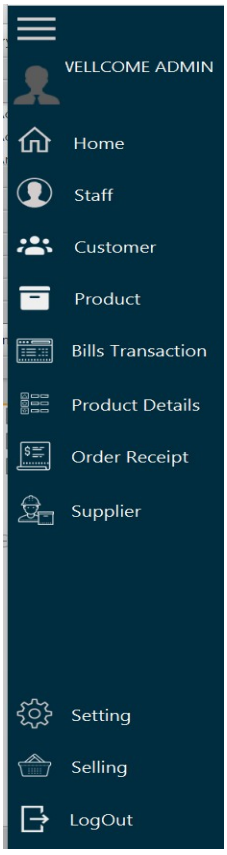
```java
public void updateRevenueChart() {
        revenueChart.getData().clear();
        XYChart.Series<String, Number> series = new XYChart.Series<>();
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        Calendar calendar = Calendar.getInstance();
        calendar.set(Calendar.DAY_OF_MONTH, 1);
        try {

            Connection conn = DB.ConnectDB.getConnectDB();
            while (calendar.get(Calendar.YEAR) == Calendar.getInstance().get(Calendar.YEAR)
                    && calendar.get(Calendar.MONTH) <= Calendar.getInstance().get(Calendar.MONTH)) {
                String currentDate = sdf.format(calendar.getTime());
                String sql = "SELECT SUM(TotalPrice) AS TotalRevenue FROM Bills WHERE BillDate = ?";
                PreparedStatement pst = conn.prepareStatement(sql);
                pst.setString(1, currentDate);
                ResultSet rs = pst.executeQuery();
                if (rs.next()) {
                    int totalRevenue = rs.getInt("TotalRevenue");
                    int dayOfMonth = calendar.get(Calendar.DAY_OF_MONTH);
                    series.getData().add(new XYChart.Data<>(currentDate, totalRevenue));
                }
                pst.close();
                calendar.add(Calendar.DAY_OF_MONTH, 1);
            }

            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }

        revenueChart.getData().add(series);
    }
```

2.4. Timeline

```java
@Override
  public void initialize(URL url, ResourceBundle rb) {

      timeline = new Timeline(
              new KeyFrame(Duration.millis(500), event -> {
                  updateTime();
              })
      );
      timeline.setCycleCount(Animation.INDEFINITE);
      timeline.play();
  }
```

3. Tabbar

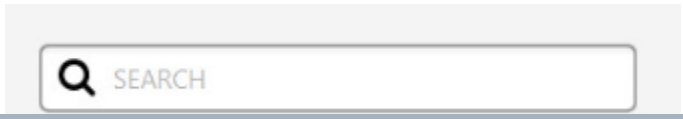




3.1. Code

```

    @FXML
    private void handleSelling() {
        try {
            // Load the OrderTransaction.fxml file
            FXMLLoader loader = new FXMLLoader(getClass().getResource("OrderTransaction.fxml"));
            Parent orderTransactionPane = loader.load();

            // Access the controller instance from the loader
            OrderTransactionController orderTransactionController = loader.getController();

            // Set the CustomerController instance in OrderTransactionController
            orderTransactionController.setCustomerController(new CustomerController());

            // Clear the main pane and add OrderTransaction.fxml to it
            main.getChildren().clear();
            main.getChildren().add(orderTransactionPane);

            // Show the opacity pane
            opacityPane.setVisible(true);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
     @FXML
    private void handleBills() {
        try {
            FXMLLoader loader = new FXMLLoader(getClass().getResource("Bills.fxml"));
            Parent customerPane = loader.load();

            main.getChildren().clear();
            main.getChildren().add(customerPane);
            opacityPane.setVisible(true);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
     @FXML
    private void handleCustomer() {
        try {
            FXMLLoader loader = new FXMLLoader(getClass().getResource("Customer.fxml"));
            Parent customerPane = loader.load();

            main.getChildren().clear();
            main.getChildren().add(customerPane);
            opacityPane.setVisible(true);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
     public void handleDashboardButtonClick() {
        try {

            FXMLLoader loader = new FXMLLoader(getClass().getResource("/sam/FXMLStaff.fxml"));
            Parent mainPane = loader.load();

            main.getChildren().clear();
            main.getChildren().add(mainPane);

            opacityPane.setVisible(true);

        } catch (IOException e) {
            e.printStackTrace();

        }
     }
```

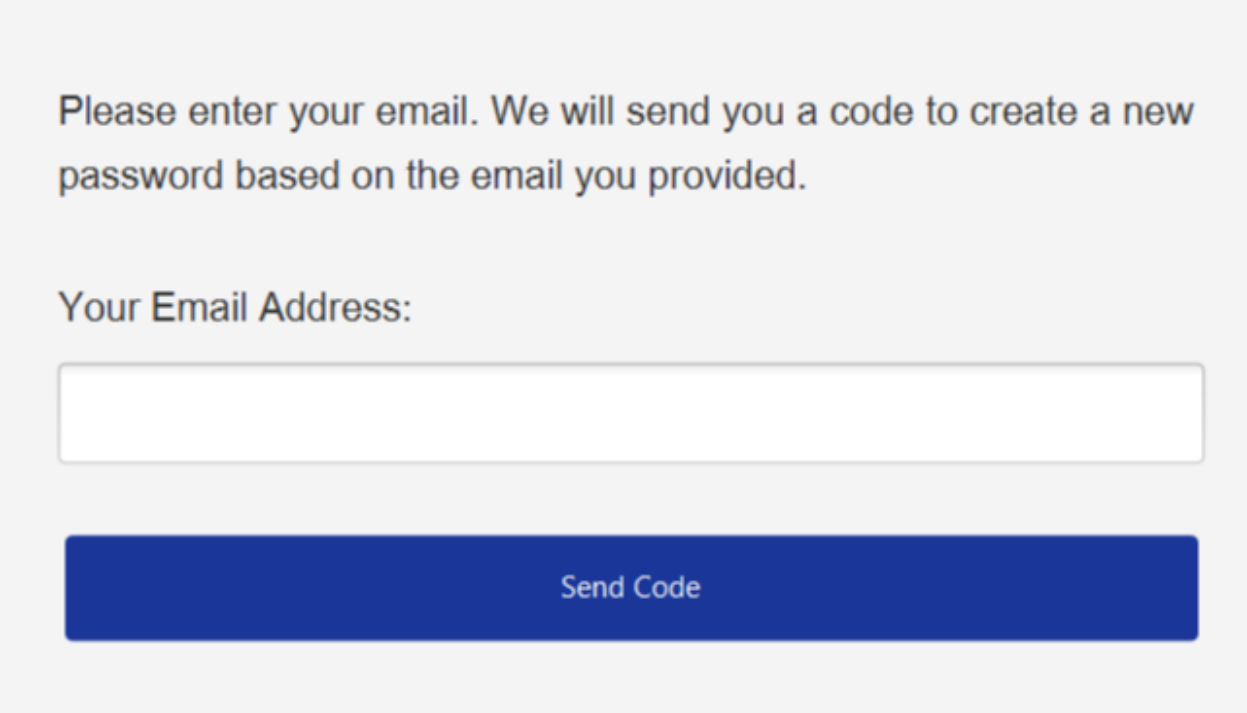
4. Search




```java
@FXML
private void searchItem() {
    String searchItem = OrderSearchItem.getText().trim().replaceAll("\\s+", "");
    productList.clear();

    if (searchItem.isEmpty()) {

        productList.addAll(datalist());
    } else {

        String sql = "SELECT p.MedicineID, p.Item ,p.MedicineName, pd.BatchNo, pd.ExpireDate, p.SellingPrice, pd.Quantity1 "
                + "FROM tblProduct p "
                + "INNER JOIN tblProductDetails pd ON p.MedicineID = pd.MedicineID";

        try {

            Connection conn = DB.ConnectDB.getConnectDB();
            PreparedStatement pst = conn.prepareStatement(sql);
            ResultSet rs = pst.executeQuery();

            while (rs.next()) {
                String medicineName = rs.getString("MedicineName").replaceAll("\\s+", ""); //
                if (medicineName.toLowerCase().contains(searchItem.toLowerCase())) {
                    TableProduct data = new TableProduct(rs.getString("MedicineID"),
                            rs.getString("MedicineName"),
                            rs.getString("BatchNo"),
                            rs.getDate("ExpireDate"),
                            rs.getString("Item"),
                            rs.getInt("SellingPrice"),
                            rs.getInt("Quantity1"));
                    productList.add(data);
                }
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    tableProduct.setItems(productList);
}
```

Coding Review 3

1. Send OTP Email in Forgot Password



1.1. Code

```java
@FXML
    private void Sendcode(ActionEvent event) throws Exception {
        if (event.getSource() == btnSendCode) {

            forgotpane.setVisible(false);
            verifypane.setVisible(true);
            resetpane.setVisible(false);
            sendOTP();

        }
    }

    @FXML
    private void VerifyCode(ActionEvent event) throws Exception {
        if (event.getSource() == btnVerify) {

            String otp = hide.getText();
            String enteredOtp = txtCode.getText().trim();

            if (otp.equals(enteredOtp)) {
                // OTP is correct, load the Staff Dashboard
                forgotpane.setVisible(false);
                verifypane.setVisible(false);
                resetpane.setVisible(true);
            } else {
                JOptionPane.showMessageDialog(null, "Invalid OTP. Please try again.");
            }
        }
    } private String generateOTP(int len) {
        String str = "ABCDEFGHIJKLMNOPQRSTUVWXYZ" + "0123456789";
        int n = str.length();
        StringBuilder otp = new StringBuilder();
        for (int i = 1; i <= len; i++) {
            otp.append(str.charAt((int) (Math.random() * str.length())));
        }
        return otp.toString();
    }

    public void sendOTP() {
        int len = 6;
        String otp = generateOTP(len);
        hide.setText(otp);
        String host = "smtp.gmail.com";
        final String username = "mingken036@gmail.com";
        final String password = "phch ducw memx hopt";
        Properties props = new Properties();
        props.put("mail.smtp.host", host);
        props.put("mail.smtp.port", 465);
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.starttls.enable", "true");
        props.put("mail.smtp.debug", true);
        props.put("mail.smtp.socketFactory.port", 465);
        props.put("mail.smtp.socketFactory.class", "javax.net.ssl.SSLSocketFactory");
        props.put("mail.smtp.socketFactory.fallback", false);

        try {
            Session session = Session.getDefaultInstance(props, null);
            session.setDebug(true);

            MimeMessage message = new MimeMessage(session);
            message.setText("Your OTP is " + hide.getText());
            message.setSubject("OTP For your Reset Account");
            message.setFrom(new InternetAddress(username));
            message.addRecipient(RecipientType.TO, new InternetAddress(txtemail.getText().trim()));
            message.saveChanges();
            try {
                Transport transport = session.getTransport("smtp");
                transport.connect(host, username, password);
                transport.sendMessage(message, message.getAllRecipients());
                transport.close();
                JOptionPane.showMessageDialog(null, "OTP has send to your Email id");
            } catch (Exception e) {
                JOptionPane.showMessageDialog(null, "Please check your internet connection");
            }

        } catch (Exception e) {

            e.printStackTrace();
            JOptionPane.showMessageDialog(null, e);
        }

    }
```

2. Export File PDF Button


```java
public void exportToPDF() {
        AlertMessage alert = new AlertMessage();
        FileChooser fileChooser = new FileChooser();
        fileChooser.setTitle("Save PDF");
        String fontPath = "src/font/arial.ttf";
        fileChooser.getExtensionFilters().addAll(new FileChooser.ExtensionFilter("PDF Files", "*.pdf"));
        File selectedFile = fileChooser.showSaveDialog(Export.getScene().getWindow());

        if (selectedFile != null) {
            try {

                PdfFont font = PdfFontFactory.createFont(fontPath, PdfEncodings.IDENTITY_H, true);

                PdfWriter writer = new PdfWriter(selectedFile.getAbsolutePath());
                PdfDocument pdf = new PdfDocument(writer);
                Document document = new Document(pdf);

                Paragraph title = new Paragraph("Bills Transaction").setBold().setFontSize(18);
                document.add(title);


                float[] columnWidths = {100, 100, 100, 100, 100, 100};
                Table table = new Table(columnWidths);

                TableView<TableBills> tableView = tableBills;
                for (TableColumn<TableBills, ?> column : tableView.getColumns()) {

                    table.addCell(new Cell().add(new Paragraph(column.getText()).setFont(font)));

                }

                ObservableList<TableBills> dataList = tableView.getItems();
                for (TableBills data : dataList) {
                    table.addCell(new Cell().add(new Paragraph(data.getBillID())));
                    table.addCell(new Cell().add(new Paragraph(data.getCustomerName())));
                    table.addCell(new Cell().add(new Paragraph(data.getCustomerPhone())));
                    table.addCell(new Cell().add(new Paragraph(String.valueOf(data.getTotalPrice()))));
                    table.addCell(new Cell().add(new Paragraph(data.getPaymentMethod())));
                    table.addCell(new Cell().add(new Paragraph(String.valueOf(data.getBillDate()))));

                }
                document.add(table);
                document.close();
                Desktop.getDesktop().open(selectedFile);

                alert.successMessage("Export successful!");
            } catch (IOException e) {
                alert.errorMessage("Export failed: " + e.getMessage());
            }
        }
    }
```

3. Notification Expired Product


```java
@FXML
    private void checkAllProductsExpiryInfo() {
        List<ProductExpiryTable> expiringProductsList = fetchExpiringProductsFromDatabase();


        Platform.runLater(() -> {
            expiryCountLabel.setText("There are " + expiringProductsList.size() + " items that will expire within 1 month.");
        });
    }

    @FXML
    private void showProductExpiryInfoForm() {
        List<ProductExpiryTable> expiringProductsList = fetchExpiringProductsFromDatabase();

        try {
            FXMLLoader loader = new FXMLLoader(getClass().getResource("/sam/ProductExpiryInfo.fxml"));
            Parent root = loader.load();
            ProductExpiryInfoController productExpiryInfoController = loader.getController();
            productExpiryInfoController.setExpiringProducts(expiringProductsList);

            Stage stage = new Stage();
            stage.setScene(new Scene(root));
            stage.initStyle(StageStyle.UNDECORATED);
            stage.show();

        } catch (IOException ex) {
            System.out.println("Error loading ProductExpiryInfo.fxml: " + ex.getMessage());
        }
    }

    private List<ProductExpiryTable> fetchExpiringProductsFromDatabase() {
        List<ProductExpiryTable> expiringProductsList = new ArrayList<>();
        String sql = "SELECT p.MedicineID, p.MedicineName, pd.ManufacturingDate, pd.ExpireDate "
                + "FROM tblProductDetails pd "
                + "JOIN tblProduct p ON pd.MedicineID = p.MedicineID";

        try (Connection conn = DB.ConnectDB.getConnectDB(); PreparedStatement pst = conn.prepareStatement(sql); ResultSet rs = pst.executeQuery()) {

            while (rs.next()) {
                String medicineID = rs.getString("MedicineID");
                String medicineName = rs.getString("MedicineName");
                LocalDate manufacturingDate = rs.getDate("ManufacturingDate").toLocalDate();
                LocalDate expireDate = rs.getDate("ExpireDate").toLocalDate();

                long daysLeft = ChronoUnit.DAYS.between(LocalDate.now(), expireDate);

                if (daysLeft >= 0 && daysLeft <= 30) {
                    long daysUntilExpired = ChronoUnit.DAYS.between(LocalDate.now(), expireDate);
                    ProductExpiryTable product = new ProductExpiryTable(medicineID, medicineName, manufacturingDate, expireDate, daysUntilExpired);
                    expiringProductsList.add(product);
                }
            }

        } catch (Exception e) {
            System.out.println("Error fetching product details: " + e.getMessage());
        }

        return expiringProductsList;
    }
```

TASK SHEET CODING REVIEW 1, 2, 3

Project: Pharmacy			**Date of preparation of activity plan:**
#	**Task**	**Prepared by**	**Status**
1	Coding Review 1	Hoang Gia Huy	Completed
2	Coding Review 2	Hoang Gia Huy	
3	Coding Review 3	Hoang Gia Huy	
4	Document	Hoang Gia Huy & Tran Nhat Linh	
5	PowerPoint	Nguyen Anh Minh	
6	Video	Nguyen Anh Quan	
Review			Signature of instructor
			Mr. Pham Cong Danh

CODE PART

Project: Pharmacy		Date of preparation of activity plan:		
#	**Code Part**	**Main Coder**	**Supporter**	**status**
1	Login	Hoang Gia Huy	N/A	Completed
2	Register	Hoang Gia Huy	N/A	Completed
3	Forgot Password	Hoang Gia Huy	N/A	Completed
4	Email	Hoang Gia Huy	Nguyen Anh Minh	Completed
5	Admin Dashboard	Tran Nhat Linh	N/A	Completed
6	Staff Dashboard	Tran Nhat Linh	N/A	Completed
7	Order Transaction	Tran Nhat Linh	N/A	Completed
8	Category	Nguyen Anh Minh	N/A	Completed
9	Import Medicine Details	Nguyen Anh Minh	N/A	Completed
10	Import Order Receipt	Doan Duc Do	N/A	Completed

11	Supplier	Doan Duc Do	N/A	Completed
12	Account	Nguyen Anh Minh	N/A	Completed
13	Export PDF	Nguyen Anh Minh	N/A	Completed
14	Staff	Nguyen Anh Quan	N/A	Completed
15	Payroll	Nguyen Anh Quan	N/A	Completed
16	Timekeeping	Nguyen Anh Quan	N/A	Completed
17	Check in-out	Nguyen Anh Quan	Nguyen Anh Minh	Completed
18	Search	Nguyen Anh Quan & Nguyen Anh Minh & Tran Nhat Linh	N/A	Completed
19	Customer	Tran Nhat Linh	N/A	Completed
Review		Signature of instructor		
		Mr. Pham Cong Danh		