FLOWER SHOP

Supervisor:	Mr. Pham Cong Danh	
Batch.	T5.2308.M0	
Group	Group 1	
Serial No	Enrollment Number	Student Name
1.	Student1501858	Nguyen Lam Chi Nguyen
2.	Student1501207	Vo Hoang Khai
3.	Student1506037	Ho Thi Bich Lien
4.	Student1501216	Tran Thi Thao

Month: 07 **Year**: 2024

This is to certify that

Mr.   **NGUYEN LAM CHI NGUYEN**

Mr.   **VO HOANG KHAI**

Mrs.   **HO THI BICH LIEN**

Mrs.   **TRAN THI THAO**

Have successfully Designed & Developed

FLOWER SHOP

Submitted by:

Mr. PHAM CONG DANH

Date Of Issue:

10/7/2024

Authorized Signature:

Ho Thi Bich Lien

Content

Content

Acknowledge

Synposis

Acknowledge

As we understand that the eProject is a step-by- step learning environment that closely simulates the class-room and Lab based learning environment into actual implementation. An electronic, live juncture on the machine that allows you to.

Practice step by step i.e. laddered approach.

Build a application.

Usage of certain utilities in applications designed by user.

Single program to unified code leading to a complete application.

Learn implementation of concepts in a phased manner.

Enhance skills and add value.

Work on real life projects.

Give a real life scenario and help to create applications more complicated and useful.

We would like to send a great thank to our professor and others student for the addorable supports during the time in project.

 Beside serveral mistakes we had made in the project, we hope to have more oppotunities to widen our knownledge .

Your Sincerely,

   Team Group 01.

Synposis

The Objective of this program we aim is to give a sample project to work on real life projects. These applications help us build a larger more robust application.

The objective is not to teach us JavaFX but to provide us with a real life scenario and help us create basic applications using the tools.

This project is meant for students like us who have completed the module of JavaFX. These programs should be done in the Lab sessions with assistance of the faculty if required.

The FLOWER SHOP brings you flowers from around the world such as gerberas, lotus flowers, wisteria flowers, hydrangeas, king flowers, dahlias, blue-eyed chrysanthemums, forget-me-nots, peonies, orchids, flower shore....

And get back eprojects@aptech.ac.in as the assigned schedule.

REVIEW 1

Problem Defination

FLOWER SHOP is a famous flower selling software on the market. Now FLOWER SHOP wants to give manager a florist software to be designed to display the variety of flowers available in the store.

The software allows managers to browse and search for flower samples using schedules or filter by flower categories and attributes.

Users must register for an account to view customer invoices, payment methods, create invoices...., they must register an account.Users who already have an account can log in. And do not have the right to change any information other than their own.

The software allows administrators (business owners and their appointees) to view and update (when required) existing products, categories, flower attributes,employees information...

If management wants to change sensitive information (e.g. remove flowers, ban customers, etc.), they must have the "Administrator" permission level, otherwise they will not be able to access these functions. In addition to managing products, categories, orders, etc. Admins can also view reports on products, orders, and customers.

Customer Requirements

I. Customer's Specific Requirements

The software allows users to register and log in to become a employee, can use all software functions such as service evaluation, offline flower buying, offline order payment.…

II. Functional Requirements

There are 2 main actors:

- .Users
- .Administrator

1. Users

1.1. Users

Users can view described product information to make selections.

a.Description

Users can visit and see outstanding flowers in the product catalog, flowers by type.

b.Functional requirements

Users can view product and product detail information by flowers.

1.2. .Users can search for flowers

a. Description

Users can search for flowers.

b.Functional requirements

Users can search for flowers.

1.3. Users can search for flowers

a .Description

Users view detailed information about flowers such as name, price,...

b.Functional requirements

Users view information about flowers such as title, description, price.

1.4. Users can log in:

a.Description

After registering an account, users can log in. Once logged in, they

can modify their personal information, including their password.

They can also access more features: rate the service, place flowers in

the cart.

b. Functional requirements:

User must have an existing account and not be a banned account.

Users must provide Email and Password to log in.

1.5. Users can log in:

a. Description:

Users can change their profile information such as name, date of birth,... and recover password

b. Functional requirements:

Users can change their profile information such as name, date of birth, etc. when they log in. Users can also change to a new password

1.6 Users can create sell flowers :

a. Description:

Users can find the type of flower they want by searching. Users can create sell flowers and make payments. They will be asked to provide billing and payment information.

b. Functional requirements:

User must be logged in to use this function. Users can only create invoice of flowers available in the software. Users need to provide the information below to complete the order. Users can pay in cash or pay via card for orders .

1.7 User can view historical invoice

a. Description:

Users view flower sell history.

b. Functional requirements:

Users view invoice history or order status in their account information.

2. Admin

Admin must be logged in to perform this function:

2.1. Admin Can Manage Invoice (florists):

a. Description:

The software must allow the Admin to interact with the invoice. When a users create invoice, it will be displayed in the order list and detailed information about these orders.

b. Functional requirements:

Invoice must contain: customer_id, flower_id, name, quantity, price,date,payment,Customer ID,id employ. Admin can view a list of existing invoice and view information. Admin can change or cancel invoice

2.2. Administrators can manage software content

a. Description:

The software must allow the Admin to interact with the User's Account. Admin can view the list of existing registered users. Administrators can ban users who violate the software's rules by deleting the user.

b. Functional requirements:

- Must be logged in as Admin to ban users

- Admin can search for user by Username and Email

- Admin can ban (delete) users.

2.3. Administrators can manage software content

a. Description:

Administrators can manage software content, add products, edit product prices, change inventory quantities, and can also disable them from the software.

b. Functional requirements:

Admin provides image content, product descriptions, etc

2.4. Admin can manage flowers:

a. Description:

The software must allow the administrator to interact with the flowers. Admin can view the list of available flowers. Admin can add new flowers and modify flower name, description, status, category, etc. Admin can delete flowers. Administrators can view vehicle details when clicking on the flower.

b. Functional requirements:

To add or update flowers, admins must provide:Flower ID,Flower Name, Color,Price Input,Quantity, Image,…

Other requirements:

Must be logged in as an administrator to add or update flowers, Name must be unique, Category must exist, Must be an Admin to delete products, Can only delete products that are not part of an order user's products, Admin can search products by title, Admin can filter by category

2.5. Admin can manage flower types:

a. Description:

The software must allow the administrator to interact with the flower. Administrators can view the list of existing flower types, insert new types, and modify type names. As an Admin, they can delete flower types that do not contain any existing flowers. Flowers without flowers are empty flowers.

b. Functional requirements:

To add or modify a flower type, the administrator must provide:Flower ID, Flower Name .

Other requirements:

Must be logged in as admin to add or modify categories, Name must be unique, ID of category must exist, Must be Admin to delete categories, Can only delete empty categories.

2.6. Admin can register an account :

a. description

Admin must register a new account. Once they have an account, they can log in and access more features. Users who are not logged in cannot browse products.

b. Functional requirements

admin must provide the information below to create an account:

- Name

- Address

- Phone

- Email (must be unique)

- Password

- Date Of Birth

-Role

And some additional information.

System Requirements:

Hardware / Software requirement

1. **Hardware**:

   Software Server

   - Processor Intel Core I3 or higher.
   - Memory 6 GB RAM or greater.
   - Modem/ADSL Internet access is required.

   Client

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

2. **Software**:

   Server:

   - Software server: JavaFX
   - Operation System: Window 7 or later.
   - Databases: SQL.

   Client:

   - Operation System Window 7 or higher.

REVIEW 1 - TASK SHEET

Project Ref.	Project Title: Game Store		Date of Preparation of Activity Plan: 27/05/2024		
No.	Task	Start Date	Actual Days	Member	Status
01	Acknowledgment			-Vo Hoang Khai	Completed
02	Synopsis			- Vo Hoang Khai	Completed
03	Customer Requirement Specification			- Nguyen Lam Chi Nguyen	Completed
04	Functional Requirement Specification	27/5/2024	31/5/2024	- Tran Thi Thao	Completed
05	System Requirements			- Ho Thi Bich Lien	Completed
06	Development Software			- Ho Thi Bich Lien	Completed

REVIEW 2
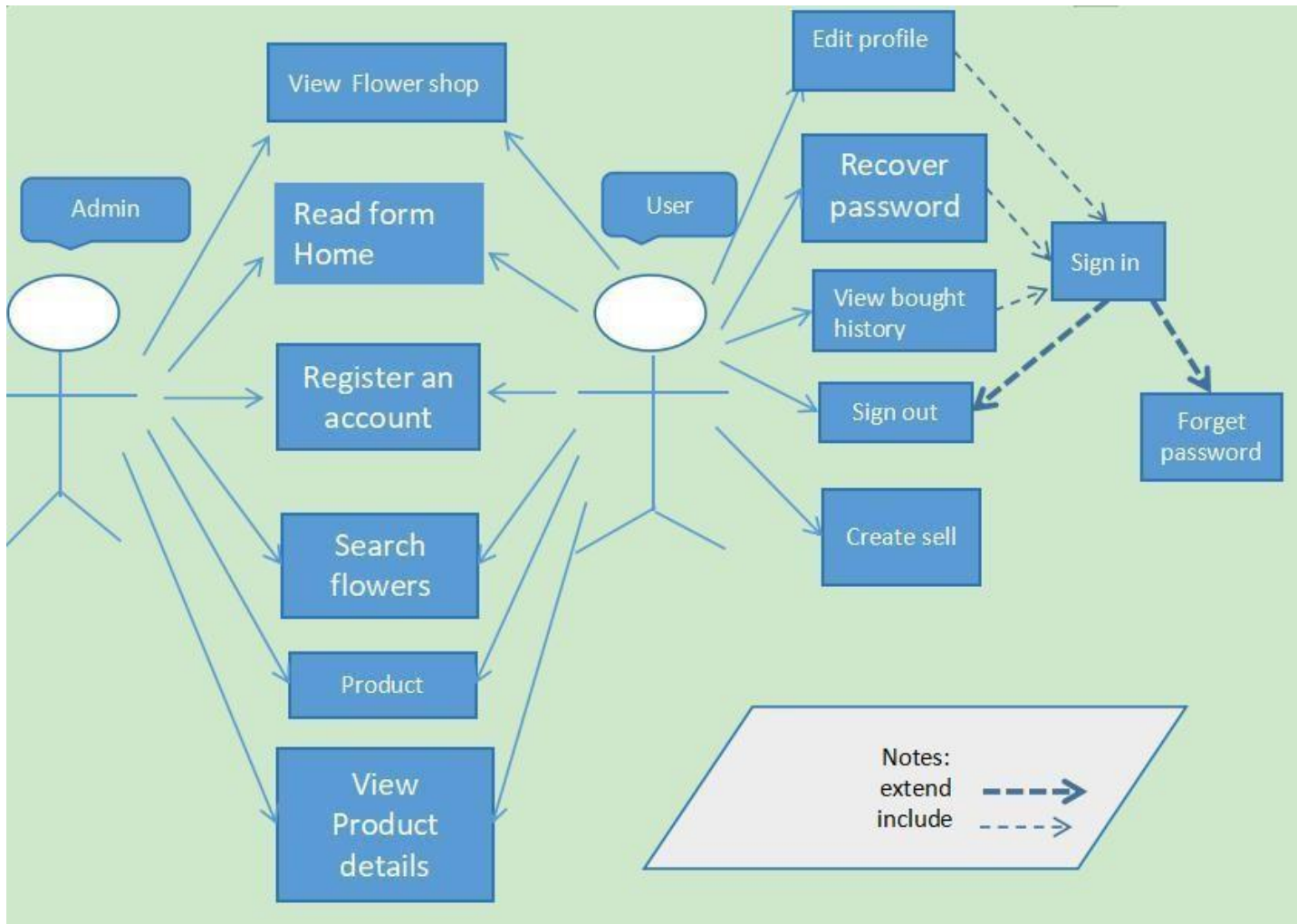
I. Admin and Users

1. Admin

1.1 View Flowers Shop

Author		
Use Case Name	View Flowers Shop	
Actors	Admin/Users	
Description	View information Flowers Shop	
Requirements		
Status	Show information Flowers Shop System	
Basic flow	Actor action: 1. Actor click "Sign in" in form Register	System Response: 2. The system will direct to the Form Home.
Alternative flow		
Exception		

1.2 Read form Home

Author		
Use Case Name	Read form Home	
Actors	Admin/Users	
Description	Read the Home form about flowers…	
Requirements		
Status	Show form Home	
Basic flow	Actor action: 1. Actor clicks "Home" on the Navigation Bar.	System Response: 2. The system displays statistical charts
Alternative flow		

Exception		

1.3   Register an account

Author		
Use Case Name	Register an account	
Actors	Admin/Users	
Description	Users want to register to become an user of Flowers Shop Management System	
Requirements		
Status	Success: The account will be created Fail: Refill information	
Basic flow	Actor action: 1. Actor click "create account" button. 3. Actor inputs Login information from and turn to Sign In form and click ' back to Login ' button	System Response: 2. . System redirects to Register software and displays Registration form with the following controls: -'Name' text field - 'Email' text field - 'Password' text field - 'Address' text field -'Date of birth' Date -'Role'checkbox - 'create account' button - 'back to Login' button 4. System validates the information. 5. System inserts the account into database.

1.4 Search Flowers

Author	
Use Case Name	Search flowers
Actors	Admin/Users
Description	Admin want to search for flowers on their schedule.
Requirements	
Status	Success: Show all flowers that match with input schedule. Fail: No flowers is show.
Basic flow	Actor action: 1. Actor input 'name and ID', then click 'Search' button.	System Response: 2. System redirects to detail, show available flowers.
Alternative flow		
Exception		

1.5 View flowers details

Author	
Use Case Name	View flowers details
Actors	Admin/Users
Description	Actor want to view flowers details.
Requirements	
Status	Show flowers detail information.
Basic flow	Actor action: 1. Actor click on 'Product Details' button.	System Response: 2. System redirects to Product details, showing Flowers detail information.
Alternative flow		

Exception		

2. User

2.1 Login

Author		
Use Case Name	Sign in	
Actors	User	
Description	User who has registered an account can Sign in.	
Requirements		
Status	Success: User is logged in to flowers shop Fail: Refill information.	
Basic flow	Actor action: 1. Actor clicks on 'Sign in' button. 3. Actor inputs Email and Password, then click the 'Sign in' button.	System Response: 2. System redirects to Sign in form with the following controls: - 'Email' text field - 'Password' text field -Show password Check - 'Sign in' button 4. System check the email and password. 5. System redirects to Home.
Alternative flow	1. Actor click "Forget Password" button. 3. Actor provide email and click "Send ". 5. Actor use new password to sign in.	2. System redirect to Forget password form. 4. System check user's email in database. If correct, send new password to user's email. Otherwise, return message "Invalid not Name and Email".
Exception	1. Actor inputs invalid email and password.	System redirects to Login page with message:"Invalid email or password'

2.2 Edit profile

Author		
Use Case Name	Edit profile	
Actors	User and Admin	
Description	User who has logged in can update profile	
Requirements		
Status	Success: Actor can see and change their info Fail: Refill information	
Basic flow	Actor action: 1. Actor click on "Save changes" button. 3. Actor update their profile, then click Save changes.	System Response: 2. System show form with following control: Profile detail : - ' Email'text field - ' Name' text field - 'Phone' text field - 'Address' text field - 'Date of Birth' field 4. System validates the information. 5. System update account database and show update success message .

2.3 Sign out

Author		
Use Case Name	Sign out	
Actors	User and Admin	
Description	User who has logged in can sign out.	
Requirements		
Status	User can sign out flowers shop	
Basic flow	Actor action: 1. Actor clicks on 'Sign out' button.	System Response: 2.System delete user session data and redirect to form Home.
Alternative flow		
Exception		

2.4 View bought history

Author		
Use Case Name	View bought history	
Actors	User and Admin	
Description	User want to view bought history	
Requirements	User who has logged in	
Status		
Basic flow	Actor action: 1. User click on "Sell" button	System Response: 2. System redirect to Sell show View bought history information.

Alternative flow		
Exception		

2.5 Create Sell flowers

Author		
Use Case Name	Sell	
Actors	User and Admin	
Description	User want to create sell from sell flowers	
Requirements	User who has logged in	
Status		
Basic flow	Actor action: 1. User click on "Sell" button 3. User can change info, additional info like "Flower name", "Price", "Quantity"… 4. User click "Pay" button.	System Response: 2. The system redirects to the Sell form with Sell information (Flower ID, Flower Name, Quantity, Price, choose payment method...), when creating the sell. 5. System save payment data to database, show success message and redirect to Sell form.
Alternative flow		
Exception		

Use Case Diagram 2

Manager Users

Manager Product

Manager Sell

Manager Supplier

Manager Inventory

ADMIN

Notes:
extend ----->
include ----->

View Home

View Product

View Product details

Manager Sell

Manager Edit Profile

View Inventory

Manager Customer

Sign in

Sign out
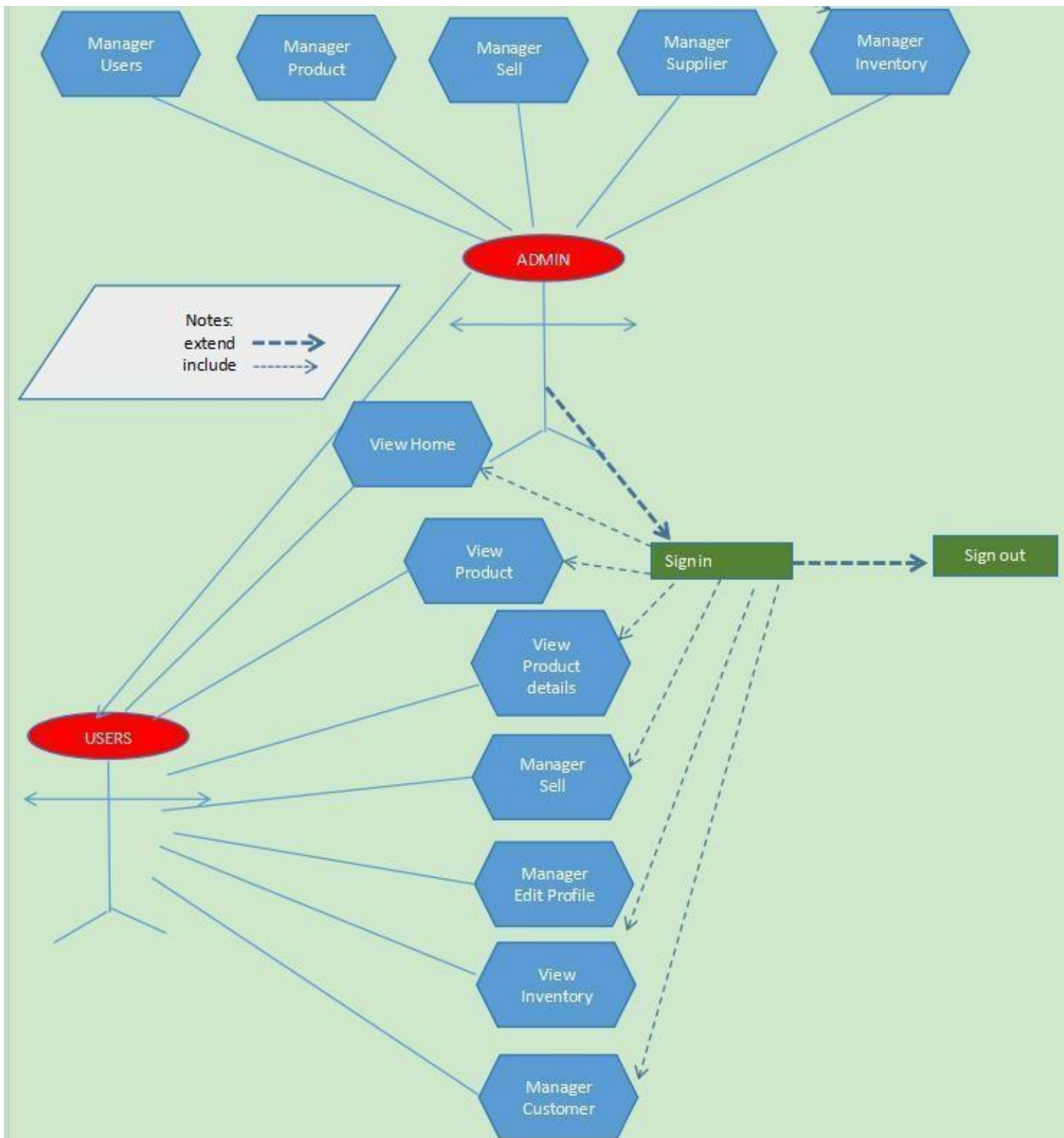
USERS

II. ADMIN and USERS
1. Admin
1.1 Admin Sign in

Author		
Use case name	Sign in	
Actors	Admin	
Description	Admin can Sign in to form Home	
Requirements	Admin provide email and password	
Status	Success: show in message successful. Fail: show message failed.	
Basic flow	Actor action: 1. Actor type email and password, then click "Sign in" button.	System response: 2. System validate information. If correct, redirect to system form Home. Otherwise, show failed login message.
Alternative flow		
Exception	- Actor type invalid email and password.	System show message: - "Wrong email or password"

1.2 Admin Sign Out

Author		
Use case name	Sign out	
Actors	Admin and User	
Description	Admin can sign out of the flower shop.	
Requirements		
Status		
Basic flow	Actor action: 1. Actor click "Sign out" link	System response: 2. System delete admin session data.
Alternative flow		

Exception		

1.3 Admin manage all functions of the moderator

Author		
Use case name	Manage all functions	
Actors	Admin	
Description	Admin can add, update, delete information for users	
Requirements	Admin is logged in	
Status		
Basic flow	Actor action: 1. Add, update, delete information for users, manager the process create invoice,add product,... 3. Update or create new and delete information as requested by the admin.	System response: 2. Navigate to the management page according to each function for processing. 4. Save the added and updated data to the database, and return to the management page.
Alternative flow		
Exception		

1.4 Admin add Product

Author	
Use case name	Manage all product
Actors	Admin
Description	Admin can create product
Requirements	Admin is logged in
Status	
Basic flow	Actor action:	System response:
	1. Click on the add product button. 3. Add information for the product , click on the add button.	2. Navigate to the page for adding product information. 4. System show add product detail form with following control: -"Upload" image - 'Flower ID' text field - 'Flower Name' text field - 'Status' choose - ' Price ' text field - 'Add' button - 'Update' button - 'Reset' button - 'Delete' button -"search" text field 6. System validates the information. 7. System update account database and show update

		success message .
Alternative flow		
Exception	Actor action: 'Name',Upload,'Flower ID', 'Flower Name','Status',' Price ',field left blank or incorrect format.	System displays message to actor: 'Name',Upload,'Flower ID', 'Flower Name','Status',' Price cannot be blank or incorrect format.

1.5 Admin delete product

Author		
Use case name	Manage all product	
Actors	Admin	
Description	Admin can delete product	
Requirements	Admin is logged in	
Status		
Basic flow	Actor action create : 1. Actor click Delete	System response: 2. System delete invoice in order database.
Alternative flow		
Exception		

1.6 Admin update product

Author		
Use case name	Manage all product	
Actors	Admin	
Description	Admin can update product	
Requirements	Admin is logged in	
Status		
Basic flow	Actor action: 1. Click on the update button. 3. Add information for the product , click on the update button.	System response: 2. Navigate to the page for updating product information. 4. System show add product detail form with following control: -"Upload" image

		- 'Flower ID' text field
		- 'Flower Name' text field-'Status' choose
		- ' Price ' text field
		- 'Add' button
		- 'Update' button
		- 'Reset' button
		- 'Delete' button
		-"search" text field
		5. System validates the information.
		6. System update account database and show update success message .
Alternative flow		
Exception	Actor action: 'Name',Upload,'Flower ID', 'Flower Name','Status',' Price ',field left blank or incorrect format.	System displays message to actor: -'Name',Upload,'Flower ID', 'Flower Name','Status',' Price cannot be blank or incorrect format.

1.7 Admin reset products

Author	
Use case name	Manage all products
Actors	Admin
Description	Admin can create products
Requirements	Admin is logged in

Status		
Basic flow	Actor action create : 1. Click on the add button. 3. Add information for the flower , click on the add button.	System response: 2. Navigate to the page for adding product information. 4. System show add flower form with following control: -"Upload" image - 'Flower ID' text field - 'Flower Name' text field - 'Status' choose - ' Price ' text field - 'Add' button - 'Update' button - 'Reset' button - 'Delete' button -"search" text field 5. System validates the information. 6. System update account database and show update success message .
Alternative flow		
Exception	Actor action: - 'Flower ID' cannot be blank or incorrect format. - ' Flower Name' must choose. - ' Price Input' cannot be	System displays message to actor: - 'Flower ID' cannot be blank or incorrect format. - ' Flower Name' must choose.

	blank or incorrect format.	- ' Price ' cannot be
		blank or incorrect format.
	- 'Status' cannot be blank or incorrect format.	
	-'Image' must not be empty and must be in the correct form	- 'Status' cannot be blank or incorrect format.
		-'Image' must not be empty and must be in the correct form

1.8 Admin delete products detail

Author	
Use case name	Manage all products
Actors	Admin
Description	Admin can update products
Requirements	Admin is logged in
Status	
Basic flow	Actor action: 1. Click on the update button. 3. Edit information for the flower, click on the update button.	System response: 2. Navigate to the page for editing product information. 4. System show update flowers form with following control: - 'Flower ID' text field - 'Name' text field - 'Description' text field - 'Price ' text field - 'Quantity' text field - 'Color' text field

		- 'Image' text field
		- 'Add' button
		- 'Update' button
		- 'Reset' button
		- 'Delete' button
		6. System validates the information.
		7. System update account database and show update success message .
Alternative flow		
Exception	Actor action:	System displays message to actor:
	1. 'Flower ID' field left blank or incorrect format.	- 'Flower ID' cannot be blank or incorrect format.
		- 'Name' must choose.
	3. 'Name' field left blank or incorrect format.	
		- 'Description' cannot be blank or incorrect format.
	4. 'Description' field left blank or incorrect format.	
		- 'Price' cannot be blank or incorrect format.
	5. 'Price Input' field left blank or incorrect format.	
		- 'Quantity' cannot be blank or incorrect format.
	6. 'Quantity' field left blank or incorrect format.	-'Image' must not be empty and must be in the correct form
	7. 'Images' incorrect format.	

1.9 Admin delete products detail

Author		
Use case name	Manage all products detail	
Actors	Admin	
Description	Admin can delete products detail	
Requirements	Admin is logged in	
Status		
Basic flow	Actor action create : 1. Actor click delete	System response: 2. System delete bill in order database
Alternative flow		
Exception		

2. Users

2.1 Users login

Author		
Use case name	Sign in	
Actors	Users	
Description	User who has registered an account can sign in.	
Requirements	Users provide email and password.	
Status	Success: show login message successful. Fail: show message failed.	
Basic flow	Actor action: 1. Actor clicks on 'Login' button. 3. Actor inputs Email and Password, then click the	System Response: 2. System redirects to Login form with the following controls: - 'Email' text field - 'Password' text field

	'Sign in' button.	-Show password Check
		- 'Sign in' button
		4. System check the email and password.
		5. System redirects to Home..
Alternative flow	1. Actor click "Forget Password" button.	2. System redirect to Forget password form.
	3. Actor provide email and click "Send ".	4. System check user's email in database. If correct, send new password to user's email. Otherwise, return message "Invalid not email or password'
	5. Actor use new password to login.	
Exception	1. Actor inputs invalid email and password.	System redirects to Login page with message: "Invalid email or password'.

2.2 Admin sign out

Author	
Use case name	Sign out
Actors	Admin
Description	Admin can log out of the system.
Requirements	
Status	
Basic flow	Actor action: 1. Actor click "Sign out" button.	System response: 2. System delete admin session data.

Alternative flow		
Exception		

2.3 Admin create manage user accounts

Author		
Use case name	Manage user accounts in form employee.	
Actors	Admin	
Description	Admin can create user account	
Requirements	Admin is logged in	
Status		
Basic flow	Actor action create : 1. Actor click on the add button on the system.	System response: 2. Navigate to the system on table. 3. Save the added data to the database and navigate to the system on table .
Alternative flow		
Exception	- Actor enters incorrect information into the input fields.	System show message: - error of user account information

2.4. Admin update manage user accounts

Author		
Use case name	Manage user accounts	
Actors	Admin	
Description	Admin can update user account	
Requirements	Admin is logged in	
Status		
Basic flow	Actor action create :	System response:

	1. Actor click on the update button .	2. Navigate to the users information updating in the table.
		3. Save the added data to the database and navigate to the system.
Alternative flow		
Exception	- Actor enters incorrect information into the input fields.	System show message: - error of user account information

2.5. Admin delete manage user accounts

Author		
Use case name	Manage user accounts	
Actors	Admin	
Description	Admin can delete user account	
Requirements	Admin is logged in	
Status		
Basic flow	Actor action create : 1. Actor click the Delete button.	System response: 2.System delete users account database.
Alternative flow		
Exception		

DATA FLOW DIAGRAM (DFD)
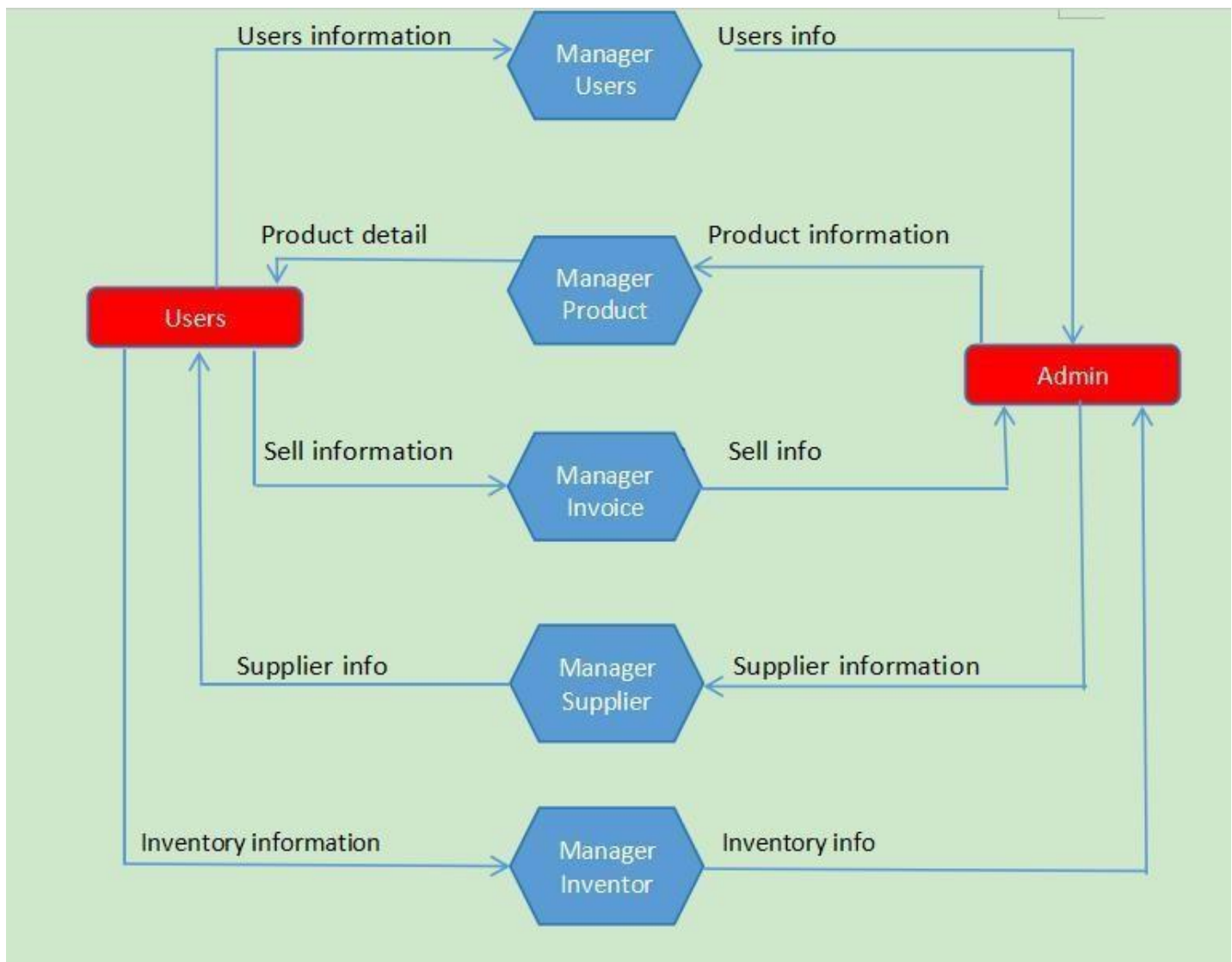
ENTITY RELATIONSHIP DIAGRAM (ERD)

Customer *
- 🔑 idcus
- CustomerID
- CustomerName
- Address
- Phone
- Email
- Password
- gender
- status

Invoice *
- flower_id
- name
- quantity
- price
- date
- payment
- CustomerID
- idemploy

Product
- 🔑 ProductID
- flower_id
- name
- status
- price
- image
- date

ProductDetail
- 🔑 iddetail
- ProductDetailID
- mass
- price
- color
- NhacungcapID
- des
- ProductID

Employee
- 🔑 idemploy
- EmployeeId
- EmployeeName
- Address
- EmployeePhone
- EmployeeEmail
- EmployeePass
- Dateofbirth
- status
- role

Invoice_info
- total
- date
- id

Nhacungcap
- 🔑 idsupplier
- NhacungcapID
- fullname
- email
- phone
- address
- priceinput
- quantity
- status
- ProductID

Hangtonkho
- 🔑 idinven
- InventoryID
- ProductID
- Quantity
- DateAddes
- LastUpdated
- supplier

xcvbn

Table Design:

1. Table Customer:

Column Name	Data Type	Allow Nulls
idcus	int	☐
CustomerID	int	☑
CustomerName	nvarchar(MAX)	☑
Address	nvarchar(MAX)	☑
Phone	nvarchar(MAX)	☑
Email	nvarchar(MAX)	☑
Password	nvarchar(MAX)	☑
gender	int	☑
status	nvarchar(MAX)	☑
		☐

2. Table Employee:

Column Name	Data Type	Allow Nulls
idemploy	int	☐
EmployeeId	int	☑
EmployeeName	nvarchar(MAX)	☑
Address	nvarchar(MAX)	☑
EmployeePhone	nvarchar(MAX)	☑
EmployeeEmail	nvarchar(MAX)	☑
EmployeePass	nvarchar(MAX)	☑
Dateofbirth	date	☑
status	nvarchar(MAX)	☑
role	nvarchar(MAX)	☑
		☐

3. Table Invoice(sell):

LAPTOP-MA66S7MN....afx - dbo.Invoice

	Column Name	Data Type	Allow Nulls
🔑	id	int	☐
	customer_id	int	☑
	flower_id	int	☑
	name	nvarchar(MAX)	☑
	quantity	int	☑
	price	nvarchar(MAX)	☑
	date	date	☑
	payment	nvarchar(MAX)	☑
	CustomerID	int	☑
	idemploy	int	☑
			☐

4. Table Invoice Info:

LAPTOP-MA66S7MN....dbo.Invoice_info

	Column Name	Data Type	Allow Nulls
▶	total	nvarchar(MAX)	☑
	date	date	☑
	id	int	☑
			☐

5. Table Product:

Column Name	Data Type	Allow Nulls
ProductID	int	☐
flower_id	int	☑
name	nvarchar(MAX)	☑
status	nvarchar(MAX)	☑
price	nvarchar(MAX)	☑
image	nvarchar(MAX)	☑
date	date	☑
		☐

6. Table Product details:

Column Name	Data Type	Allow Nulls
iddetail	int	☐
ProductDetailID	int	☑
mass	nvarchar(MAX)	☑
price	nvarchar(MAX)	☑
color	nvarchar(MAX)	☑
NhacungcapID	int	☑
des	nvarchar(MAX)	☑
ProductID	int	☑
		☐

7. Table Supplier:

Column Name	Data Type	Allow Nulls
🔑 idsupplier	int	☐
NhacungcapID	int	☑
fullname	nvarchar(MAX)	☑
email	nvarchar(MAX)	☑
phone	nvarchar(MAX)	☑
address	nvarchar(MAX)	☑
priceinput	nvarchar(MAX)	☑
quantity	nvarchar(MAX)	☑
status	nvarchar(MAX)	☑
ProductID	int	☑
		☐

8. Table Inventory:

Column Name	Data Type	Allow Nulls
🔑 idinven	int	☐
InventoryID	int	☑
ProductID	int	☑
Quantity	nvarchar(MAX)	☑
DateAddes	date	☑
LastUpdated	date	☑
supplier	int	☑
		☐

REVIEW 2 TASKSHEET

Projec t Ref.	Project Title: Game Store		Date of Preparation of Activity Plan: 31/05/2024		
No.	Task	Start Date	Actual Days	Member	Status
01	Use Case Visitor	31/05/24	7/06/24	-Vo Hoang khai	Completed
02	Use Case User			- Ho Thi Bich Lien	Completed
03	Use Case Moderation			- Nguyen Lam Chi Nguyen	Completed
04	Use Case Admin			- Tran Thi Thao	Completed
05	Data Flow Diagram			- Vo Hoang Khai	Completed
06	ERD			- Ho Thi Bich Lien	Completed
07	Table Design			- Tran Thi Thao	Completed
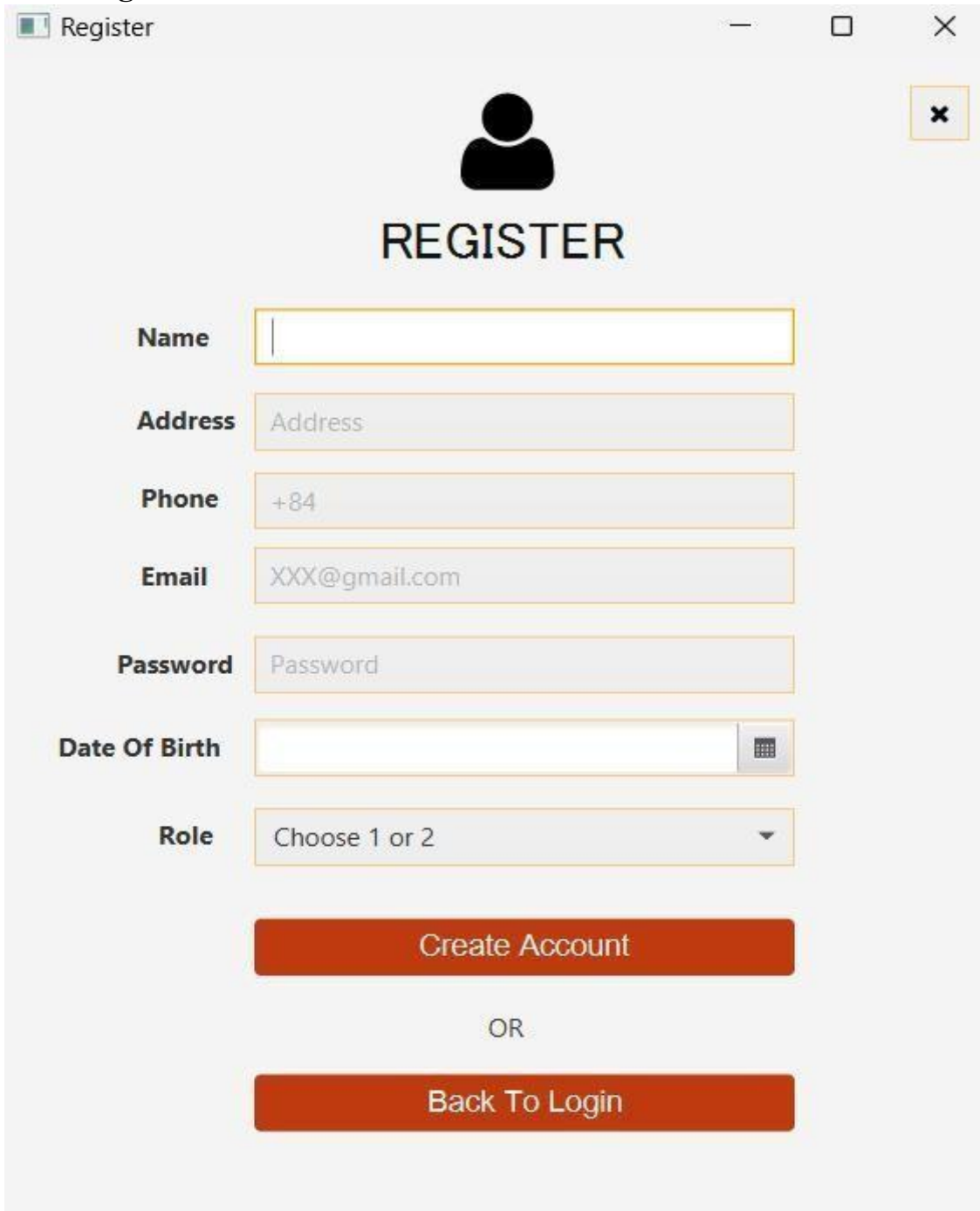
REVIEW 3

I.Sitemap:
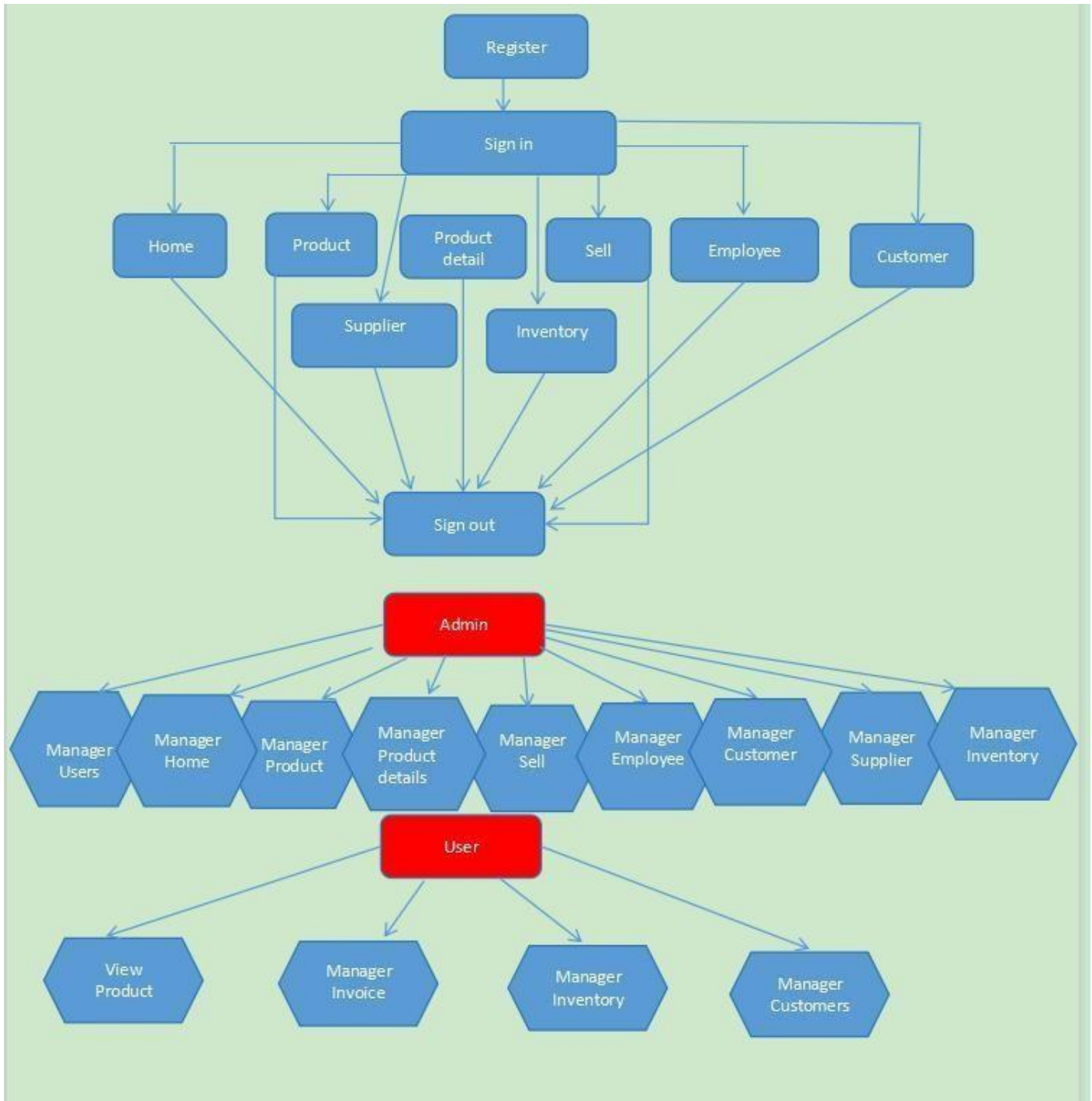
II. Screenshot:

1. Register form:

User fill full information:

Text field String Name ,Validate:not be empty.

Text field String Email ,Validate: not be empty,Maximum input is 30 characters

Text field String Address ,Validate:not be empty.

Text field String Password ,Validate:not be empty,Maximum input is characters.Choose date of birth,Validate:not be empty.

Role: Admin-Users(Default Admin).

Then click the "Create Account" button to complete the login process.

Administrators or users must register an account when logging into the flower shop .

++ Create Account button: Click to register a new account Coding for Register form:

```java
    public void EmployeeAdd() {
    String sql = "INSERT INTO Employee (EmployeeName, Address, EmployeePhone, EmployeeEmail, EmployeePass, Dateofbirth, role) "
            + "VALUES(?,?,?,?,?,?,?)";

connect = ConnectDB.getConnectDB();
    try {
        Alert alert;

        if (txtname.getText().isEmpty()
                || txtaddress.getText().isEmpty()
                || txtphone.getText().isEmpty()
                || txtemail.getText().isEmpty()
                || txtpass.getText().isEmpty()
                || picker_date.getValue() == null
                || combo_role.getSelectionModel().getSelectedItem() == null) {

            alert = new Alert(Alert.AlertType.ERROR);
            alert.setTitle("Error Message");
            alert.setHeaderText(null);
            alert.setContentText("Please Fill All Blank Fields");
            alert.showAndWait();

            } else {
            prepare = connect.prepareStatement(sql);
            prepare.setString(1, txtname.getText());
            prepare.setString(2, txtaddress.getText());
            prepare.setString(3, txtphone.getText());
            prepare.setString(4, txtemail.getText());
            prepare.setString(5, txtpass.getText());
            prepare.setDate(6, java.sql.Date.valueOf(picker_date.getValue()));
              prepare.setString(7, (String) combo_role.getSelectionModel().getSelectedItem());

        prepare.executeUpdate();

            alert = new Alert(Alert.AlertType.INFORMATION);
            alert.setTitle("Information Message");
            alert.setHeaderText(null);
            alert.setContentText("Successfully Added!");
            alert.showAndWait();
registerClear() ;

        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```
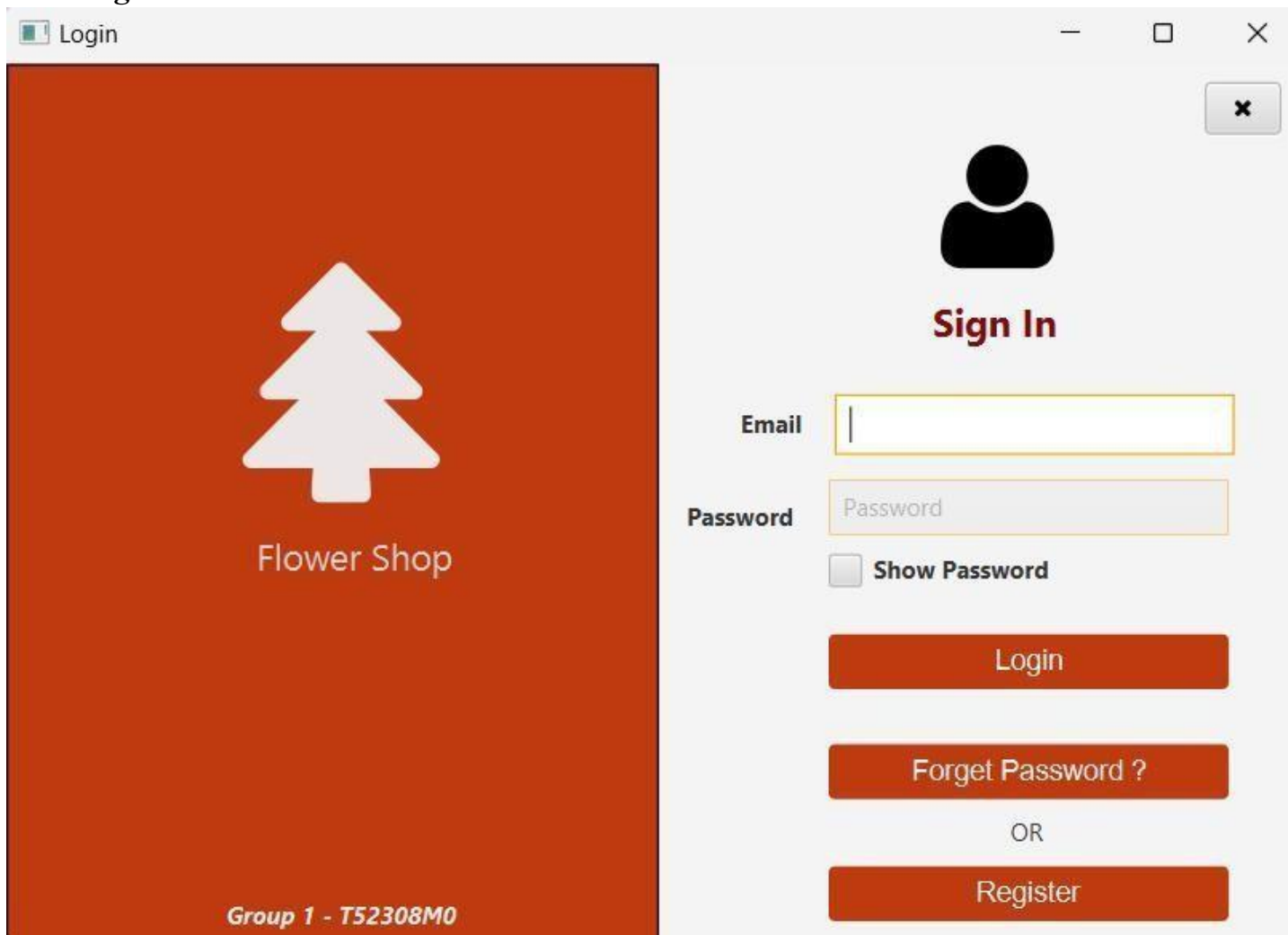
++Back to login button:Return to the Sign in page

```java
    private void mtlogin(ActionEvent event) throws IOException {
      Parent root = FXMLLoader.load(getClass().getResource("FXMLDocument.fxml"));
        Stage stage = new Stage();
        stage.setTitle("Login");
        stage.setScene(new Scene(root));
        stage.show();
        FRegister.getScene().getWindow().hide();
    }
```

2. Sign in form:

Users will fill in information log in:

Text field email,Validate:not be empty,Maximum input is 30 characters.

Text field Password ,Validate:not be empty.Not Checkbox Show password:

for the next sign in.

Login Button: Complete fill in and start validation to Register form

Coding for Login button

```java
public void login() {
    String sql = "SELECT * FROM Employee WHERE EmployeeEmail = ? and EmployeePass = ?";
    connect = ConnectDB.getConnectDB();
    try {
        prepare = connect.prepareStatement(string:sql);
        prepare.setString(i: 1, string:email.getText());
        prepare.setString(i: 2, string:password.getText());

        result = prepare.executeQuery();

        Alert alert;
        if (email.getText().isEmpty() || password.getText().isEmpty()) {
            alert = new Alert(at: Alert.AlertType.ERROR);
            alert.setTitle(title: "Error Message");
            alert.setHeaderText(headerText: null);
            alert.setContentText(contentText: "Please fill all blank fields");
            alert.showAndWait();
        } else {
            if (result.next()) {
                String userEmail = email.getText();

                // Validate email format and length
                if (userEmail.length() <= 30 && userEmail.matches(regex: "^[A-Za-z0-9._%+-]+@gmail
                    String role = result.getString(string:"role");  // Get user role from query res

                    alert = new Alert(at: Alert.AlertType.INFORMATION);
                    alert.setTitle(title: "Information Message");
                    alert.setHeaderText(headerText: null);
                    alert.setContentText("Successfully Logged In as " + role);
                    alert.showAndWait();

                    getData.username = userEmail;

                    // Close current login window
                    loginBtn.getScene().getWindow().hide();

                    // Open interface based on user role
                    Parent root;
                    if ("Admin".equals(anObject: role)) {
                        root = FXMLLoader.load(location: getClass().getResource(name: "Dashboard.fxml"));
                    } else {
                        root = FXMLLoader.load(location: getClass().getResource(name: "fUser.fxml"));
                    }

                    Stage stage = new Stage();
                    Scene scene = new Scene(parent:root);

                    // Add window drag functionality
                    root.setOnMousePressed((MouseEvent event) -> {
                        x = event.getSceneX();
                        y = event.getSceneY();
                    });
                    root.setOnMouseDragged((MouseEvent event) -> {
                        stage.setX(event.getScreenX() - x);
                        stage.setY(event.getScreenY() - y);
                    });

                    stage.initStyle(style: StageStyle.TRANSPARENT);
                    stage.setScene(value: scene);
                    stage.show();
                } else {
                    alert = new Alert(at: Alert.AlertType.ERROR);
                    alert.setTitle(title: "Error Message");
                    alert.setHeaderText(headerText: null);
                    alert.setContentText(contentText: "Invalid Email Format or Length (max 30 characters, must be @gm
                    alert.showAndWait();
                }
            } else {
                alert = new Alert(at: Alert.AlertType.ERROR);
```

```
                alert.setTitle(title: "Error Message");
                alert.setHeaderText(headerText: null);
                alert.setContentText(contentText: "Wrong Email/Password"
                alert.showAndWait();
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Click to go Forget password if you forget your password.

Coding for Forget Password?

```
public void likforget() {

    try {
        Alert alert = new Alert(a: AlertType.CONFIRMATION);
        alert.setTitle(title: "Confirmation Message");
        alert.setHeaderText(headerText: null);
        alert.setContentText(contentText: "You Are Forgetting Your Password !");
        Optional<ButtonType> option = alert.showAndWait();

        if (option.get().equals(obj: ButtonType.OK)) {
            // HIDE YOUR DASHBOARD FORM
            forgetbtn.getScene().getWindow().hide();

            // LINK YOUR LOGIN FORM
            Parent root = FXMLLoader.load(location: getClass().getResource(name: "Forgetpa
            Scene scene = new Scene(parent: root);
            Stage stage = new Stage();

            root.setOnMousePressed((MouseEvent event) -> {
                x = event.getSceneX();
                y = event.getSceneY();
            });

            root.setOnMouseDragged((MouseEvent event) -> {
                stage.setX(event.getScreenX() - x);
                stage.setY(event.getScreenY() - y);

                stage.setOpacity(value: .8);
            });

            root.setOnMouseReleased((MouseEvent event) -> {
                stage.setOpacity(value: 1);
            });

            stage.initStyle(style: StageStyle.TRANSPARENT);

            stage.setScene(value: scene);
            stage.show();
        }

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Register Button: Click to go Register page

Codding for Register button

```
private void nutdk(ActionEvent event) throws IOException {
    Parent root = FXMLLoader.load(location: getClass().getResource(name: "Register.fxm
    Stage stage = new Stage();
    stage.setTitle(value: "Register");
    stage.setScene(new Scene(parent: root));
    stage.show();
    fLogin.getScene().getWindow().hide();
}
```

After registering, Admin and User will go to the Sign in form.

Checkbox Show password: Show/Hide Password

Codding for show Password

```
public void showPassword() {

    if (ckshow.isSelected()) {
        pass_show.setText(value: password.getText());
        pass_show.setVisible(value: true);
        password.setVisible(value: false);
    } else {
        password.setText(value: pass_show.getText());
        pass_show.setVisible(value: false);
        password.setVisible(value: true);
    }

}
```

3.Forget password form:
 When you log in and forget your password, your password will be changed and you will
  be returned to the Sign in form.

   Users fill full information:

    Email:Text Field,Validate:not be empty,Maximum input is 30

   characters,Check if the email is already in the database.

    -Send Button: Send New Password to personal email
     ++Button : Send
Coding for button Send

```java
public void Employeeforget() {
    connect = ConnectDB.getConnectDB();

    // Validate input fields
    if (txtname.getText().isEmpty()
            || txtemail.getText().isEmpty()
            || txtpass.getText().isEmpty()) {
        showAlert(type: Alert.AlertType.ERROR, title: "Error Message", content: "Please fill all blank fields.");
        return;
    }

    // Confirmation dialog
    Alert confirmation = new Alert(at: Alert.AlertType.CONFIRMATION);
    confirmation.setTitle(title: "Confirmation Dialog");
    confirmation.setHeaderText(headerText: null);
    confirmation.setContentText(contentText: "Are you sure you want to get this user's new password?");
    Optional<ButtonType> option = confirmation.showAndWait();

    if (option.isPresent() && option.get() == ButtonType.OK) {
        final String checkSql = "SELECT * FROM Employee WHERE EmployeeName = ? AND EmployeeEmail = ?";

        try (PreparedStatement checkStmt = connect.prepareStatement(string: checkSql)) {
            checkStmt.setString(i: 1, string: txtname.getText());
            checkStmt.setString(i: 2, string: txtemail.getText());

            try (ResultSet rs = checkStmt.executeQuery()) {
                if (rs.next()) {
                    final String updateSql = "UPDATE Employee SET EmployeePass = ? WHERE EmployeeId = ?";

                    try (PreparedStatement updateStmt = connect.prepareStatement(string: updateSql)) {
                        updateStmt.setString(i: 1, string: txtpass.getText());
                        updateStmt.setInt(i: 2, i: rs.getInt(string: "EmployeeId"));

                        int affectedRows = updateStmt.executeUpdate();

                        if (affectedRows > 0) {
                            showAlert(type: Alert.AlertType.INFORMATION, title: "Information Message", content: "Successfully!");
                        } else {
                            showAlert(type: Alert.AlertType.WARNING, title: "Warning Message", content: "Unsuccessful.");
                        }
                    }
                } else {
                    showAlert(type: Alert.AlertType.ERROR, title: "Error Message", content: "Invalid Name or Email.");
                }
                forgetClear();
            }
        } catch (Exception e) {
            showAlert(type: Alert.AlertType.ERROR, title: "Error Message", content: e.getMessage());
        }
    }
}
```

3. Home form:
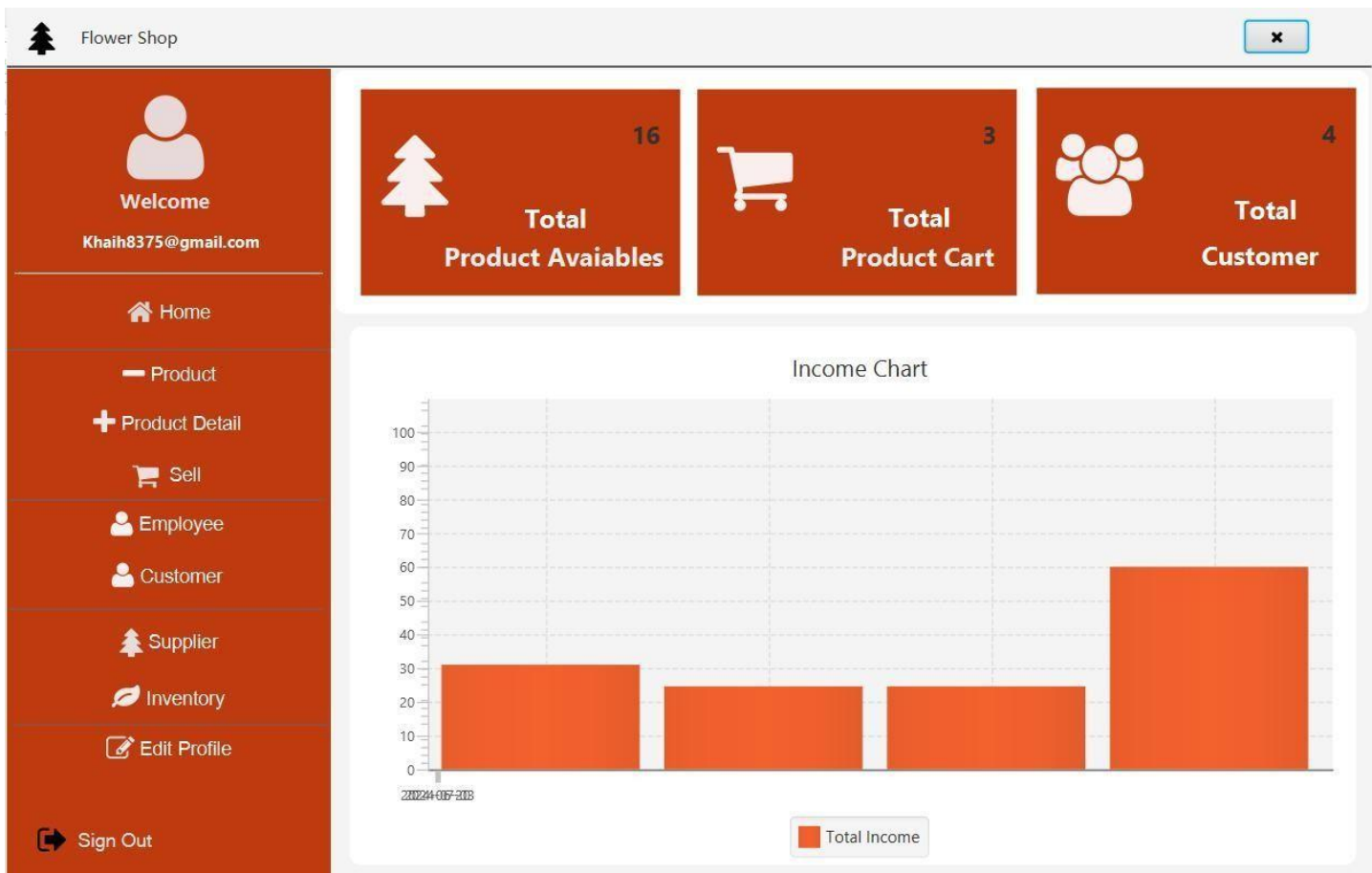
In the Home Admin and User forms, you will see Product Available, total product Cart, and total customers

++Codding for Home

```java
//show home
public void homeAF() {
    String sql = "SELECT COUNT(*) AS count_available FROM Product WHERE status = 'Available'";

    Connection connect = null;
    Statement statement = null;
    ResultSet result = null;

    try {
        connect = ConnectDB.getConnectDB();
        statement = connect.createStatement();
        result = statement.executeQuery(string: sql);

        int countAF = 0;

        if (result.next()) {
            countAF = result.getInt(string: "count_available");
        }
        home_sell.setText(value: String.valueOf(:: countAF));

    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        // Close resources in a finally block
        try {
            if (result != null) result.close();
            if (statement != null) statement.close();
            if (connect != null) connect.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

4. Product form:

In the Product form, we will add, update, reset, delete upload, Flower ID,

Flower name, status, price, and search for Flower .

Users fill full information:

Upload image:not be empty.

Text field,Validate:Flower ID,not be empty,IDs cannot be entered twice

,ID does not allow special characters to be entered.

Text field,Validate:Flower Name,not be empty,Maximum input is 20 characters.

Choose Status,Validate:not be empty,Choose 1 of 2.

Text field Price,Validate:not be empty.

Add Button:Add new products to the table.

Upload Button:Edit product information.

Reset button:delete the entered information.

Delete button:Delete the information in the table

Search: + Search product by ID shown in table below.

++Coding for Product and Product details

```java
//product
public void availableFlowersSelect() {
        flower_Data flower = availableFlower_tableView.getSelectionModel().getSelectedItem();
        int num = availableFlower_tableView.getSelectionModel().getSelectedIndex();

        if ((num - 1) < -1) {
            return;
        }

        product_flowerID.setText(value:String.valueOf(obj: flower.getFlower_id()));
        product_flowername.setText(value:flower.getName());
        product_flowerprice.setText(value:String.valueOf(obj: flower.getPrice()));

        getData.path = flower.getImage();

        String uri = "file:" + flower.getImage();

        image = new Image(string: uri, d: 154, d1:150, blm:false, blm1: true);
        product_imageView.setImage(value:image);

    }
```

5. Product detail form:

In the Product detail form, we will add, update, reset, delete,Flower ID,Quantity, description, price, color,supplier ID search for flower ID , Flower Name.

Users fill full information:

Choose Flower ID,price,Supplier ID:Choose 1 of 2 , not be empty.

Text field Description,quantity,Color,Validate:not be empty,Do not enter special characters.

Add Button:Add new products to the table.

Upload Button:Edit product information.

Reset button:delete the entered information.

Delete button:Delete the information in the table

Search: + Search product by ID, name shown in table below.

```java
//productdetail
    public void ProductFlowerId() {
        String sql = "SELECT flower_id FROM Product WHERE status IN ('Available', 'Not Available')";

        connect = ConnectDB.getConnectDB();

        try {
            ObservableList listData = FXCollections.observableArrayList();

            prepare = connect.prepareStatement(string: sql);
            result = prepare.executeQuery();

            while (result.next()) {
                listData.add(e: result.getInt(string: "flower_id"));
            }
            detailflowerid.setItems(value: listData);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

6. Sell form:

In the Sell form, the user creates an sell for available products such as

flower ID, flower name, enters the quantity,choose payment type,choose

Customer ID, and then selects Add to card.

After add to card we will select row Flower name just add when "Total" have return total amount . then click pay to payment for customer.Users fill full information:

Choose Flower ID,Flower Name,Payment Type,Customer ID:Choose 1 of 2,not be

empty,IDs cannot be entered twice,ID does not allow special characters to be

entered.

Text field Quantity,Validate:not be empty.

Pay Button:Go to the purchased checkout product

Reset button:delete the entered information.

Search: + Search product by ID shown in table below.

```java
//sell
    public void purchaseFlowerId() {
String sql = "SELECT status, flower_id FROM Product WHERE status IN ('Available', 'Not Available')";

        connect = ConnectDB.getConnectDB();

        try {
            prepare = connect.prepareStatement(string: sql);
            result = prepare.executeQuery();

            ObservableList listsp = FXCollections.observableArrayList();

            while (result.next()) {
                listsp.add(e: result.getInt(string: "flower_id"));
            }
            sell_flowerID.setItems(value: listsp);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

7. Employee form:

In the Employee form, you can add, update, delete, reset, code, name, address, phone, email, password, date of birth, status,role, and search for the employee.

Users fill full information:

Text field,Validate: ID,Name,Address,Phone,Password,Status:not be empty,,IDs cannot be entered twice,ID does not allow special characters to be entered.

Text field,Validate:Email,not be empty,Maximum input is 30 characters

Date picker Birth:not be empty.

Add Button:Add new employee to the table.

Upload Button:Edit employee information.

Reset button:delete the entered information.

Delete button:Delete the information in the table

Search: + Search Name,ID shown in table below.

```java
//Employee
public ObservableList<Employee> EmployeeListData() {
ObservableList<Employee> listData = FXCollections.observableArrayList();
        String sql = "SELECT * FROM Employee";
        connect = ConnectDB.getConnectDB();
        try {
            prepare = connect.prepareStatement(string: sql);
            result = prepare.executeQuery();

            Employee emp;

            while (result.next()) {
                emp = new Employee(EmployeeId:result.getInt(string: "EmployeeId"),
                    EmployeeName: result.getString(string: "EmployeeName"), Address: result.getString(string: "Address"),
                    EmployeePhone: result.getString(string: "EmployeePhone"), EmployeeEmail: result.getString(string: "EmployeeEmail"),
                    EmployeePass: result.getString(string: "EmployeePass"), DateofBirth: result.getDate(string: "Dateofbirth"), status: result.getString(string: "status"), role: result.getString(string:

                listData.add(e: emp);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return listData;
    }
```

8. Customer form:

In the Customer form, you can add, update, delete, reset, ID,Name, Address, Phone,Email, Password,Gender,Status, and search for the customer.

Users fill full information:

Text field,Validate: ID,Name,Address,Phone,Password,Status:not be empty.,IDs cannot be entered twice,ID does not allow special characters to be entered.

Text field,Validate:Email,not be empty,Maximum input is 30 characters.

Check Box Gender,Validate:Choose 1 of 2 , not be empty.

Add Button:Add new customer to the table.

Upload Button:Edit customer information.

Reset button:delete the entered information

Delete button:Delete the information in the table

Search: + Search Name, ID shown in table below.

```java
//customer
        //customershow
    public ObservableList<customer> CustomerListData() {
 ObservableList<customer> listData = FXCollections.observableArrayList();
        String sql = "SELECT * FROM Customer";
        connect = ConnectDB.getConnectDB();
        try {
            prepare = connect.prepareStatement(string: sql);
            result = prepare.executeQuery();

            customer cus;

            while (result.next()) {
                cus = new customer(CustomerID: result.getInt(string: "CustomerID"),
                        CustomerName: result.getString(string: "CustomerName"), Address: result.getString(string: "Address"),
                        Phone: result.getString(string: "Phone"), Email: result.getString(string: "Email"), Password: result.getString(string: "Password"), gender: result.getInt(string: "gender"),
                        status: result.getString(string: "status"));

                listData.add(e: cus);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return listData;
    }
```

9. Supplier form:

In the Supplier form, you can add, update, delete, reset, supplier ID ,Full name, email, phone, address,Product ID,Price input,quantity,status, search for the supplier ID.

Users fill full information:

Text field Supplier ID,Validate:IDs cannot be entered twice,ID does not allow special characters to be entered,not be empty.

Text field Full Name,Phone,Address,Price Input.,Quantity,Validate:not be empty

Text field:Email,Validate:not be empty,Maximum input is 30 characters.

Choose Status,Validate:not be empty.

Add Button:Add new supplier to the table.

Upload Button:Edit Supplier information.

Reset button:delete the entered information.

Delete button:Delete the information in the table

Search: + Search name,ID shown in table below.

```java
//Supplier
public void SupplierProductId() {
    String sql = "SELECT flower_id FROM Product WHERE status IN ('Available'

    connect = ConnectDB.getConnectDB();

    try {
        ObservableList listData = FXCollections.observableArrayList();

        prepare = connect.prepareStatement(string: sql);
        result = prepare.executeQuery();

        while (result.next()) {
            listData.add(e: result.getInt(string: "flower_id"));
        }
        supplier_productid.setItems(value: listData);


    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

10. Inventory form:

In the Inventory form, you can add, update, delete, reset,Flower ID, select Product, quantity,Date Add and Date Update to search by ID, name.

Users fill full information:

Text field:Flower ID,Quantity,Validate:IDs cannot be entered twice,ID does not allow special characters to be entered,not be empty.

Choose Product,Validate:not be empty,Choose 1 of 2.

Date picker Date Add,Date Update,Validate:not be empty.

Add Button:Add new Inventory to the table.

Upload Button:Edit Inventory information.

Reset button:delete the entered information.

Delete button:Delete the information in the table

Search: + Search name, ID shown in table below.

```java
//inventory
        public ObservableList<Inventory> InventoryListData() {
ObservableList<Inventory> listData = FXCollections.observableArrayList();
        String sql = "SELECT * FROM Hangtonkho";
        connect = ConnectDB.getConnectDB();
        try {
            prepare = connect.prepareStatement(string: sql);
            result = prepare.executeQuery();

            Inventory iven;

            while (result.next()) {
                iven = new Inventory(InventoryID: result.getInt(string: "InventoryID"),
                    ProductID: result.getInt(string: "ProductID"), Quantity: result.getString(string: "Quantity"),
                    DateAdded: result.getDate(string: "DateAddes"), LastUpdated: result.getDate(string: "LastUpdated"),
                    supplier: result.getInt(string: "supplier"));

                listData.add(e: iven);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return listData;
    }
```

11. Edit Profile:

In the Edit profile form, you can change your email, phone number, address, and date of birth. Contact information: phone number, email. After editing the profile, click the Save change button, the information will be changed.

Clicking Sign out will exit the flower shop.

Users fill full information:

Text field: Phone,Name,Address,Validate:not be empty,email maximum input is

30 characters

Date Picker Birth,Validate:not be empty.

Save Changes button:Save the edited information

```java
//edit profile
public void savechange() {
    connect = ConnectDB.getConnectDB();

    // Validate input fields
    if (edit_email.getText().isEmpty()
            || edit_phone.getText().isEmpty()
            || edit_address.getText().isEmpty()
            || edit_date.getValue() == null
            || edit_name.getText().isEmpty()) {
        showAlert(type: Alert.AlertType.ERROR, title:"Error Message", content: "Please fill all blank fields.");
        return;
    }

    // Validate phone number (Vietnam format: starts with 09, 08, 07, 05, or 03 and has 10 digits)
    String phoneNumberPattern = "^(09|08|07|05|03)\\d{8}$";
    Pattern pattern = Pattern.compile(regex:phoneNumberPattern);
    Matcher matcher = pattern.matcher(input:edit_phone.getText());

    if (!matcher.matches()) {
        showAlert(type: Alert.AlertType.ERROR, title:"Error Message", content: "Invalid phone number format. Please enter a valid Vi
        return;
    }

    // Validate name length
    if (edit_name.getText().length() > 30) {
        showAlert(type: Alert.AlertType.ERROR, title:"Error Message", content: "Name cannot exceed 30 characters.");
        return;
    }

    // Confirmation dialog
    Alert confirmation = new Alert(at:Alert.AlertType.CONFIRMATION);
    confirmation.setTitle(title:"Confirmation Dialog");
    confirmation.setHeaderText(headerText:null);
    confirmation.setContentText(contentText:"Are you sure you want to update this user's information?");
    Optional<ButtonType> option = confirmation.showAndWait();

    if (option.isPresent() && option.get() == ButtonType.OK) {
        final String checkSql = "SELECT * FROM Employee WHERE EmployeeEmail = ?";

        try (PreparedStatement checkStmt = connect.prepareStatement(string: checkSql)) {
            checkStmt.setString(i: 1, string: edit_email.getText());

            try (ResultSet rs = checkStmt.executeQuery()) {
                if (rs.next()) {
                    // Email is valid, proceed with updating other fields
                    final String updateSql = "UPDATE Employee SET EmployeePhone = ?, Address = ?, Dateofbirth = ?, EmployeeName = ? WHERE EmployeeEmail = ?";

                    try (PreparedStatement updateStmt = connect.prepareStatement(string: updateSql)) {
                        updateStmt.setString(i: 1, string: edit_phone.getText());
                        updateStmt.setString(i: 2, string: edit_address.getText());
                        updateStmt.setDate(i: 3, date: java.sql.Date.valueOf(date: edit_date.getValue()));
                        updateStmt.setString(i: 4, string: edit_name.getText());
                        updateStmt.setString(i: 5, string: edit_email.getText());

                        int affectedRows = updateStmt.executeUpdate();

                        if (affectedRows > 0) {
                            showAlert(type: Alert.AlertType.INFORMATION, title:"Information Message", content: "Information updated successfully!");
                        } else {
                            showAlert(type: Alert.AlertType.WARNING, title:"Warning Message", content: "Update unsuccessful.");
                        }
                    }
                } else {
                    showAlert(type: Alert.AlertType.ERROR, title:"Error Message", content: "Invalid email. Cannot update information.");
                }
                editClear();
            }
        } catch (Exception e) {
            showAlert(type: Alert.AlertType.ERROR, title:"Error Message", content: e.getMessage());
        }
    }
}
```

Work list:
1.Nguyen Lam Chi Nguyen
 - Register and Sign in
 - Permission

- Create account, Back to login
- Show password,Login,Forget Password,Register.
- Send
- Save Changes

2.Vo Hoang Khai

-Product and Product detail, Home

-Add, Update, Reset, Delete, Search

3.Tran Thi Thao

-Sell

-Add to cart,Reset, Pay, Delete cart

4.Ho Thi Bich Lien

-Supplier and Inventory

-Add, Update, Reset, Delete, Search

-Sign out.

REVIEW 3 - TASK SHEET

Project Ref.	Project Title: Game Store		Date of Preparation of Activity Plan: 7/06/2024		
No.	Task	Start Date	Actual Days	Member	Status
01	Site Map	7/6/2024	17/6/2024	-Tran Thi Thao -Nguyen Lam Chi Nguyen - Vo Hoang Khai -Ho Thi Bich Lien	Completed
02	Screenshot			- Vo Hoang Khai -Ho Thi Bich Lien -Tran Thi Thao -Nguyen Lam Chi Nguyen	Completed