University of Santo Tomas

College of Information and Computing Sciences

España, Manila

Software Engineering I

Software Design Document
for
Datu Shop Inventory Management System

Client: Datu Shop

Prepared By:
Abrigo, Nathanael Chris O.
Daliuag, Ronan Manuel D.
Dela Paz, Angelo Daniel A.
Entrata, Joshua Kyle K.
Lorenzo, Lex Zedrick M.
Padrejuan, Shaun Kristoffer C.
Salvador, Fabian III R.
Sebastian, Nigel Haim N.

3CSC

Submitted To
Asst. Prof. Janette E. Sideño

Table of Contents

1. System Overview **1**
2. System Architecture **2**
2.1 Architectural Design 2
2.2 Decomposition Description 4
3. Data Design **12**
3.1 Data Description 12
3.2 Data Dictionary 13
4. Component Design **17**
4.1 Session Management 17
4.2 Inventory Management 18
4.3 Search and Filter 21
4.4 Transaction History 22
4.5 Sales Dashboard 23
4.6 Account Management 23
4.7 Low Stocks Notification 25
4.8 Edit Account 26
5. Human Interface Design **28**
5.1 Overview of User Interface 28
5.2 Screen Mockups 30
6. Appendices **48**
6.1. Minutes of the Meeting with Client 48

1. System Overview

Datu Shop's primary service is the selling and servicing of motorcycle parts. The shop is located in Quezon City and is considered a Micro-Small-Medium Enterprise (MSME) in terms of manpower and size.

The primary problem is inefficient inventory management in keeping stock and availability of particular items and parts. In terms of the shop's methods, only the staff can identify the status of a part if they check for the part inside their inventory, which may take time and effort due to the existing inventory management.

In line with these problems, TigreTech proposed a web-based Inventory Management System to help maintain an organized structure of inventory keeping and transaction logging, all done online with the use of either a mobile device or a laptop to help speed up the search for a part's status and stocks, with a focus on making notifications about whether a part is critically low in stock to inform the owner. Included are the added features of having a search filter based on categories, tags, and keywords and a sales section to see how many parts are sold in a given timeframe, alongside a list of fast-selling products and low-stock products.

The goals of the project are as follows:

a. To integrate a management system that works with real-time data to see the stocks of a part. From the stocks, the availability of the parts will also be determined. This objective will address the shop's old pen-and-paper stock management method. Included are the search function and the low notifications feature, which will notify the owner about the critically low stock of a part to be noted for procurement.

b. To integrate a transaction history to see which parts are added, used, returned, swapped, and removed from the inventory. This objective will detail the shop's process, seeing used or returned parts. Likewise, this feature will reflect its stock in the database in real time.

c. To integrate a sales dashboard to see the sales and count overview of the shop. Additional dashboard features are the top-selling parts, low-stock parts, and the profits per category of parts. This objective uses real-time data to construct a quick overview of the shop's business model.

2. System Architecture

2.1 Architectural Design

The system will be developed using Python with the Django framework for the backend and React for the front end. Django has several built-in features, like token authentication and a built-in admin panel for building the backend. The main database will be PostgreSQL, an open-source relational database that fits well with Django projects. All pertinent database setups are handled in the settings file of the Django project, and all models will also be compiled in the models file, making for a simple configuration for database objects. All methods will be defined in the views file, and all dynamic URL rerouting methods will be stored in the URLs file in the project.

React will be used for the front end to create a dynamic and responsive user interface. User inputs will be captured through React components, which will be mapped to specific views in the backend for their respective functions.

There are two prominent user roles: Admin and Employee. Both users can use most of the features of the Inventory Tracking System, namely transaction history, search and filter, inventory management, and low stock notification. The admin has two additional capabilities: viewing the sales dashboard and export feature, and account management. The rundown of the capabilities of the two users is shown below using the Use-Case diagram and the Swimlane diagram.

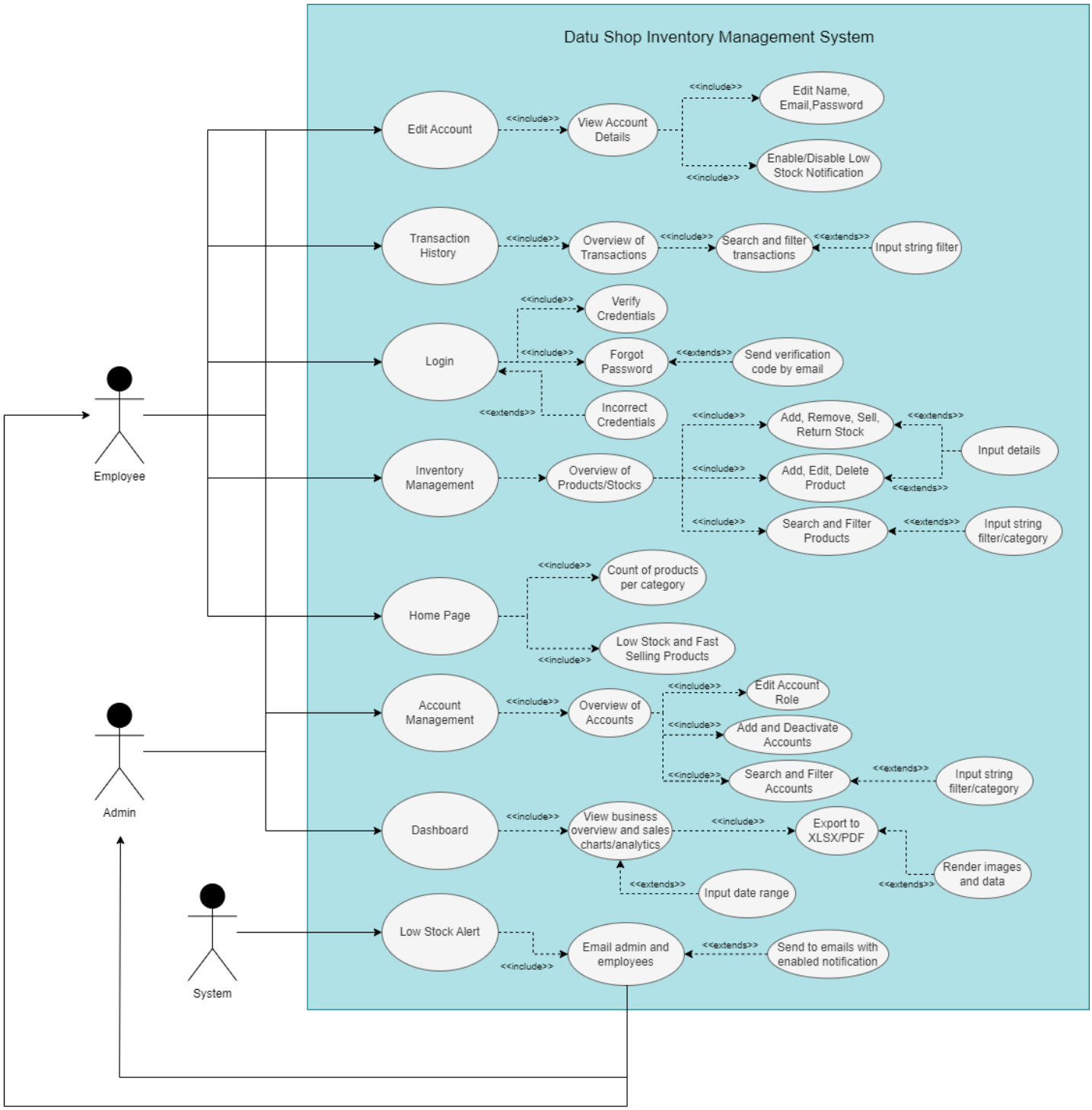

Figure 2.1: Detailed Use-Case Diagram

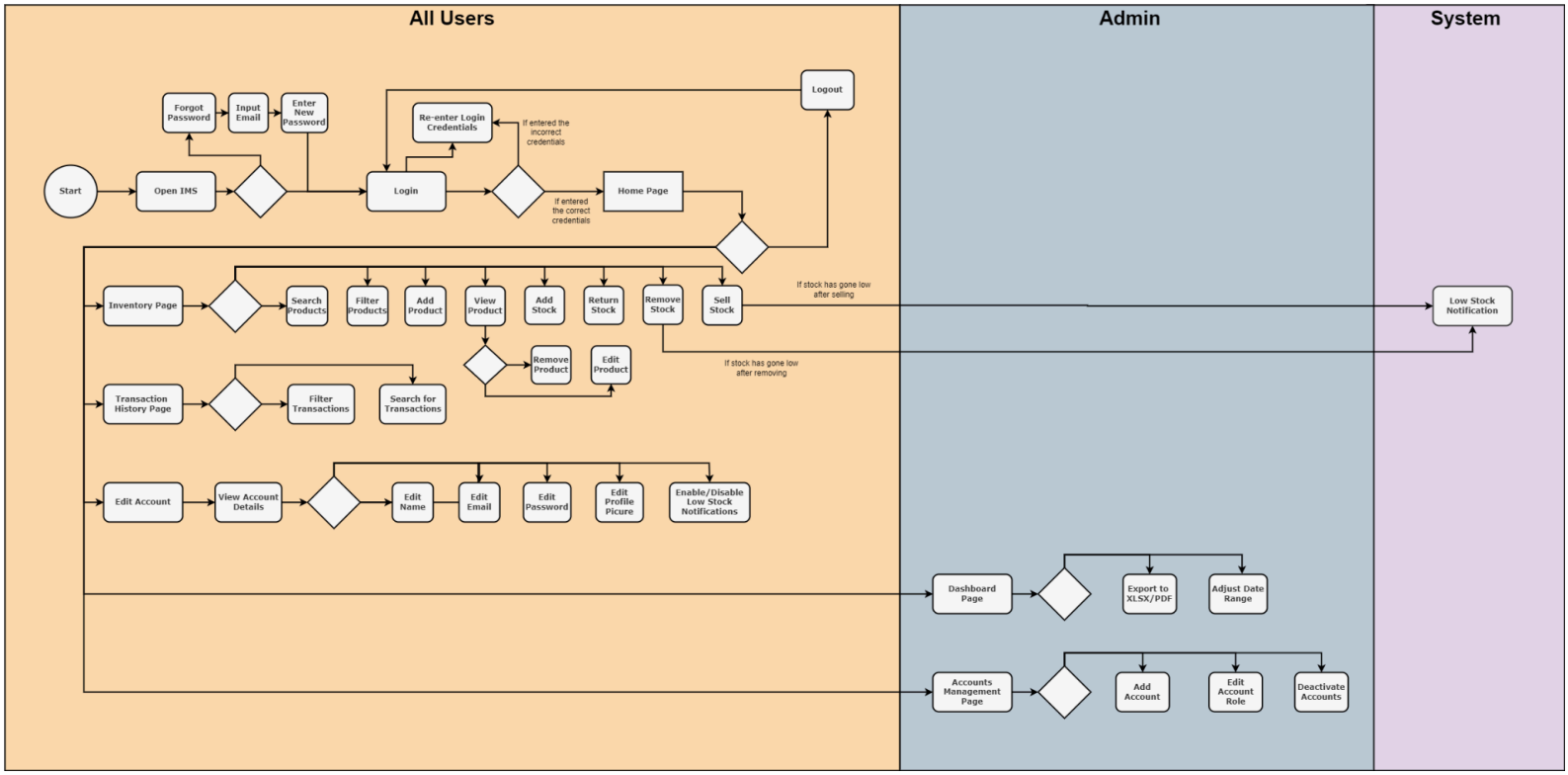

Figure 2.2: Swimlane Diagram

2.2 Decomposition Description

The proposed system can be divided into seven key features, namely Session Management, Inventory Management, Search and Filter, Transaction History, Sales Dashboard, Account Management, and Low Stock Notification. All seven features have three main components: the user, the System/UI, and the database. The user's device is also a component of the Sales Dashboard since it is needed to store the saved dashboard. The user's email is needed to send messages and notifications of low stock for the Low Stock Notification and also to receive the code needed to reset a password for a user. The interactions between components are detailed in the systems sequence diagrams below for each main feature.

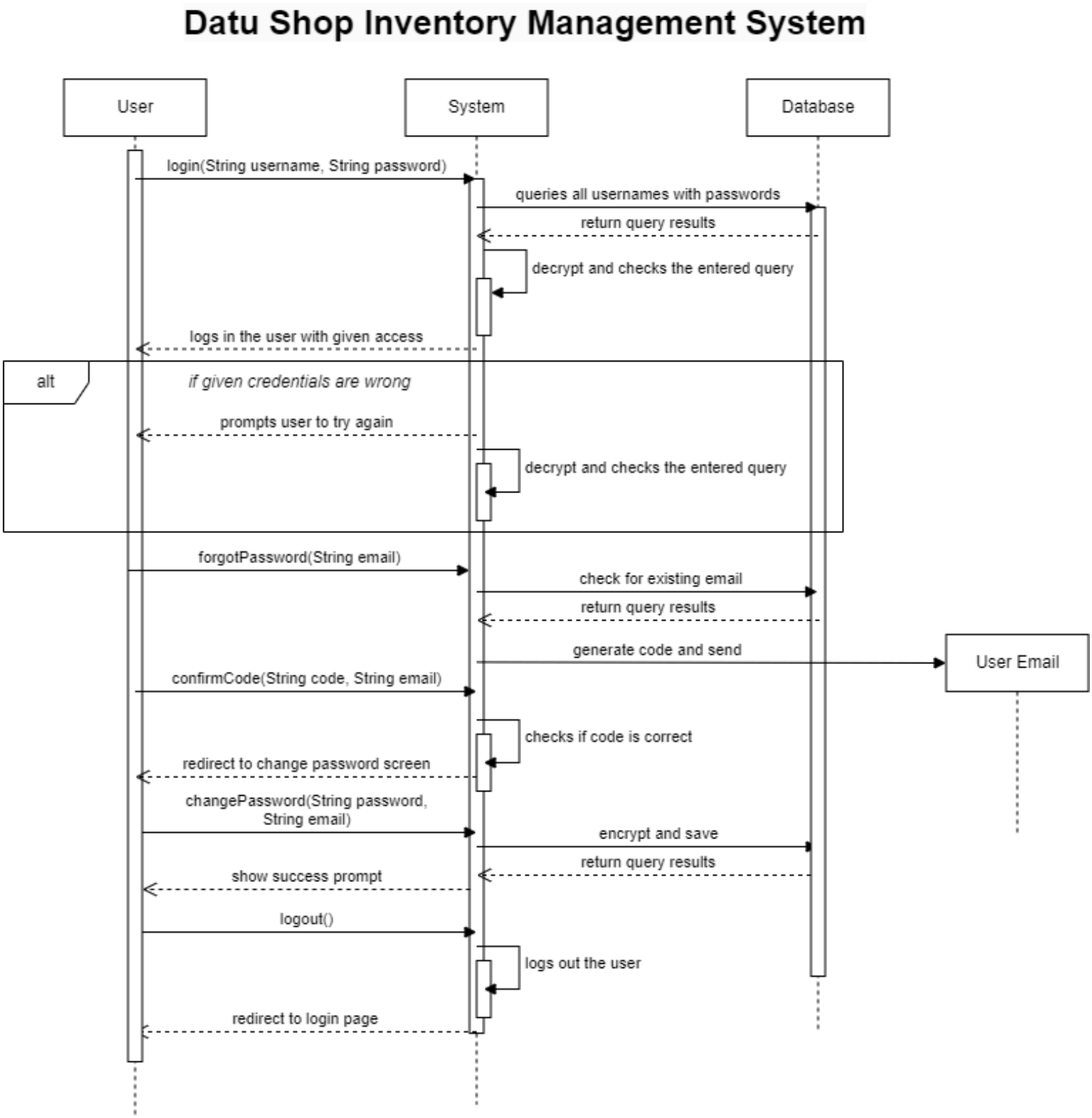

Figure 2.3.1: System Sequence Diagram: Session Management

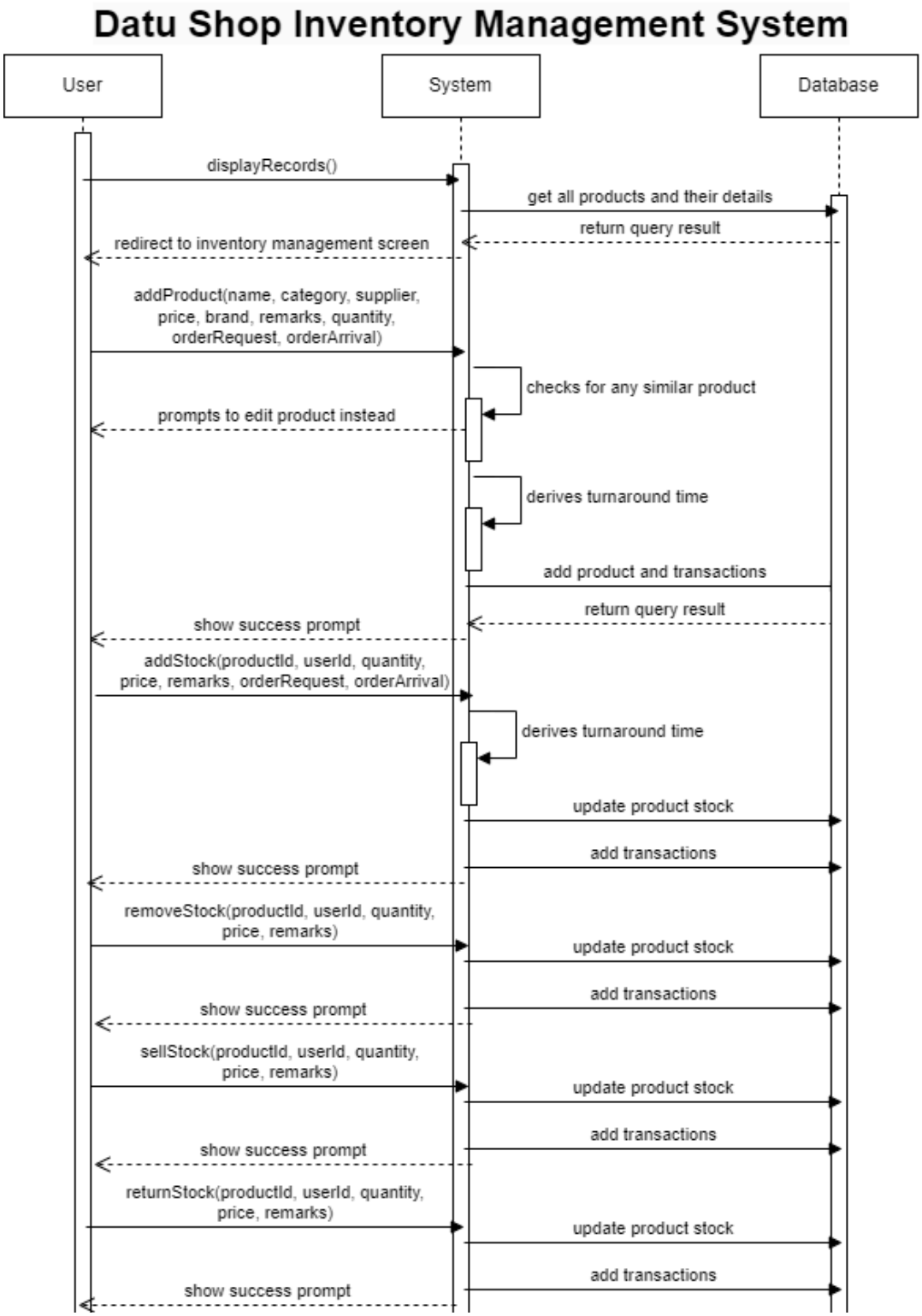

Figure 2.3.2: System Sequence Diagram: Inventory Management Part 1

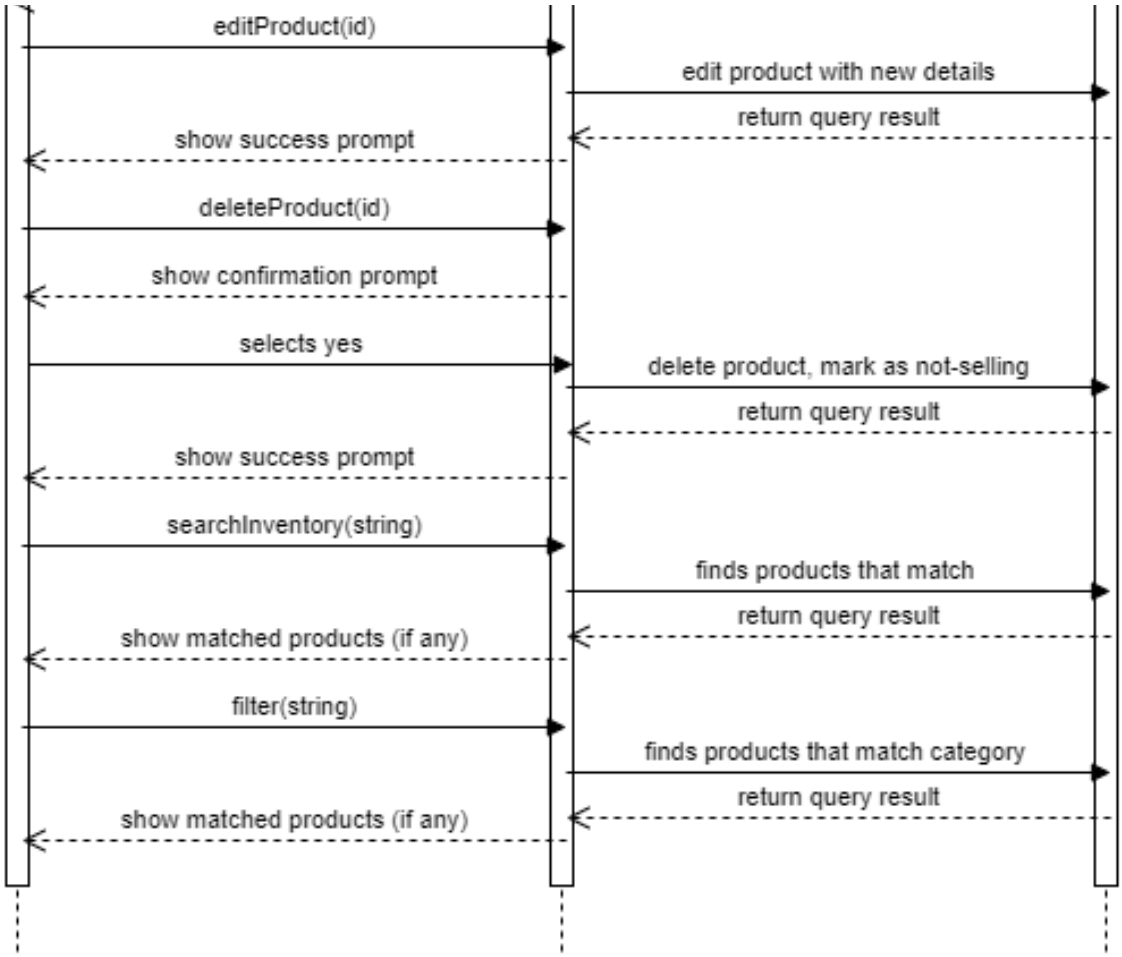

Figure 2.3.3: System Sequence Diagram: Inventory Management Part 2

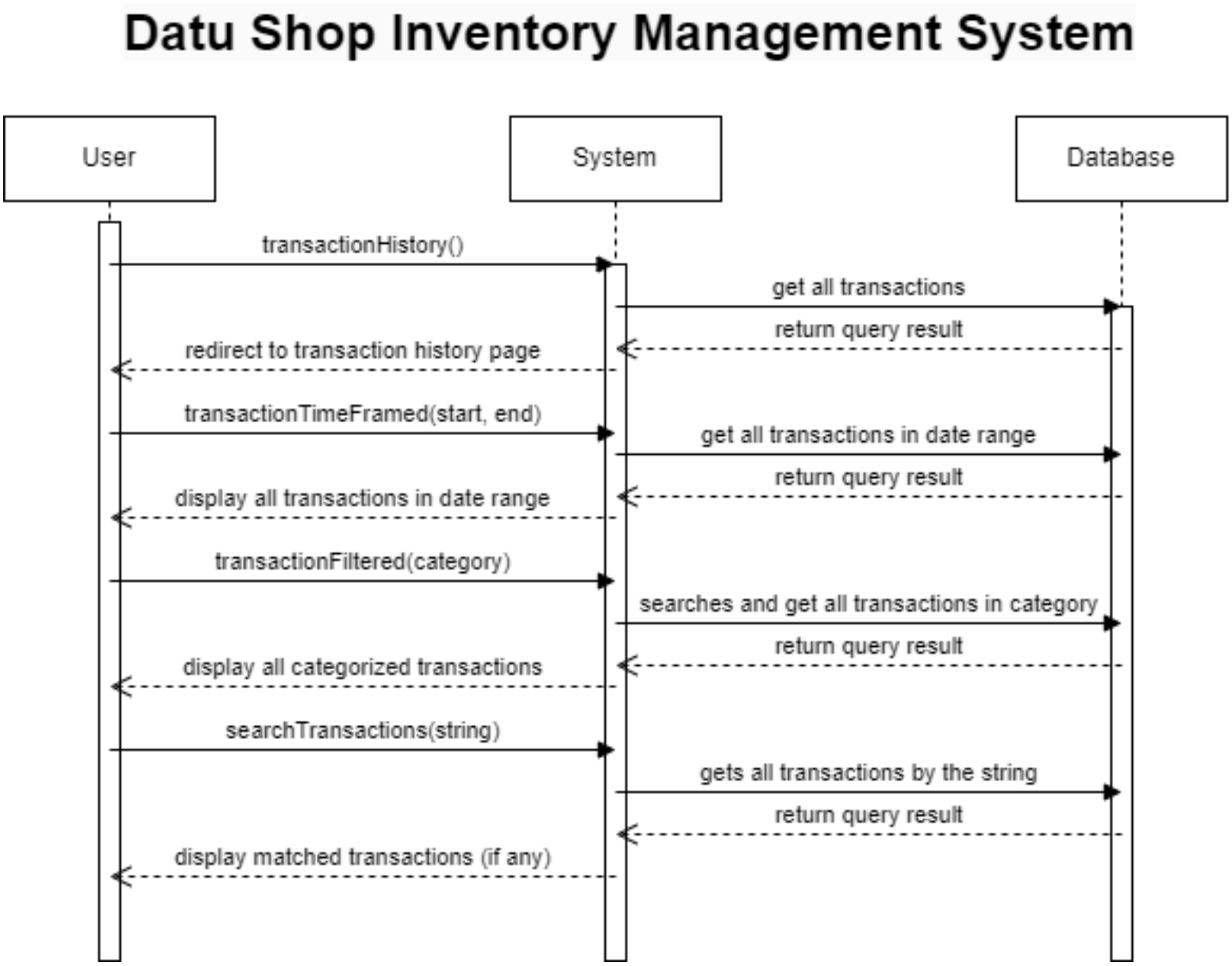

Figure 2.3.4: System Sequence Diagram: Transaction History

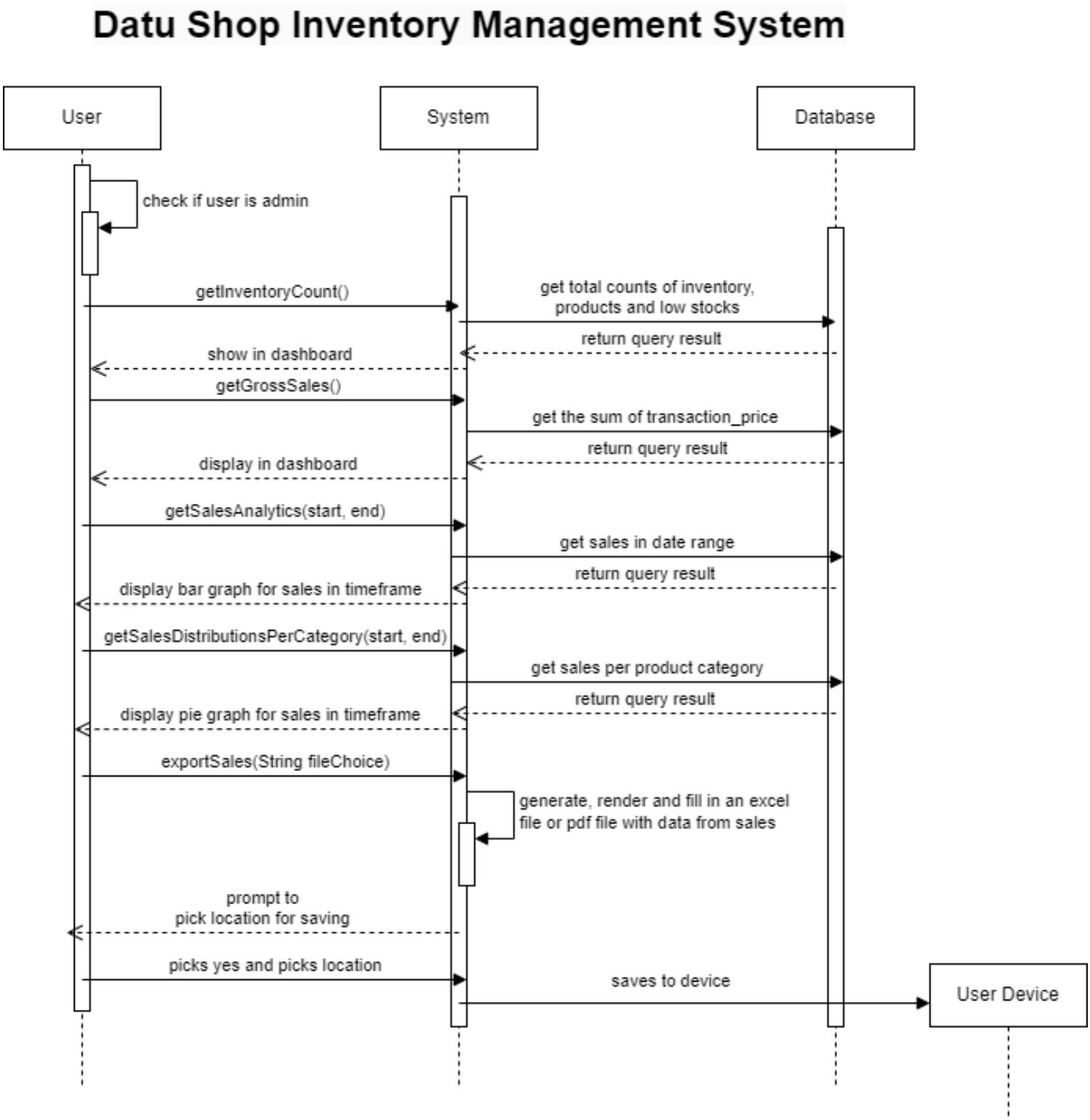

Figure 2.3.5: System Sequence Diagram: Sales Dashboard

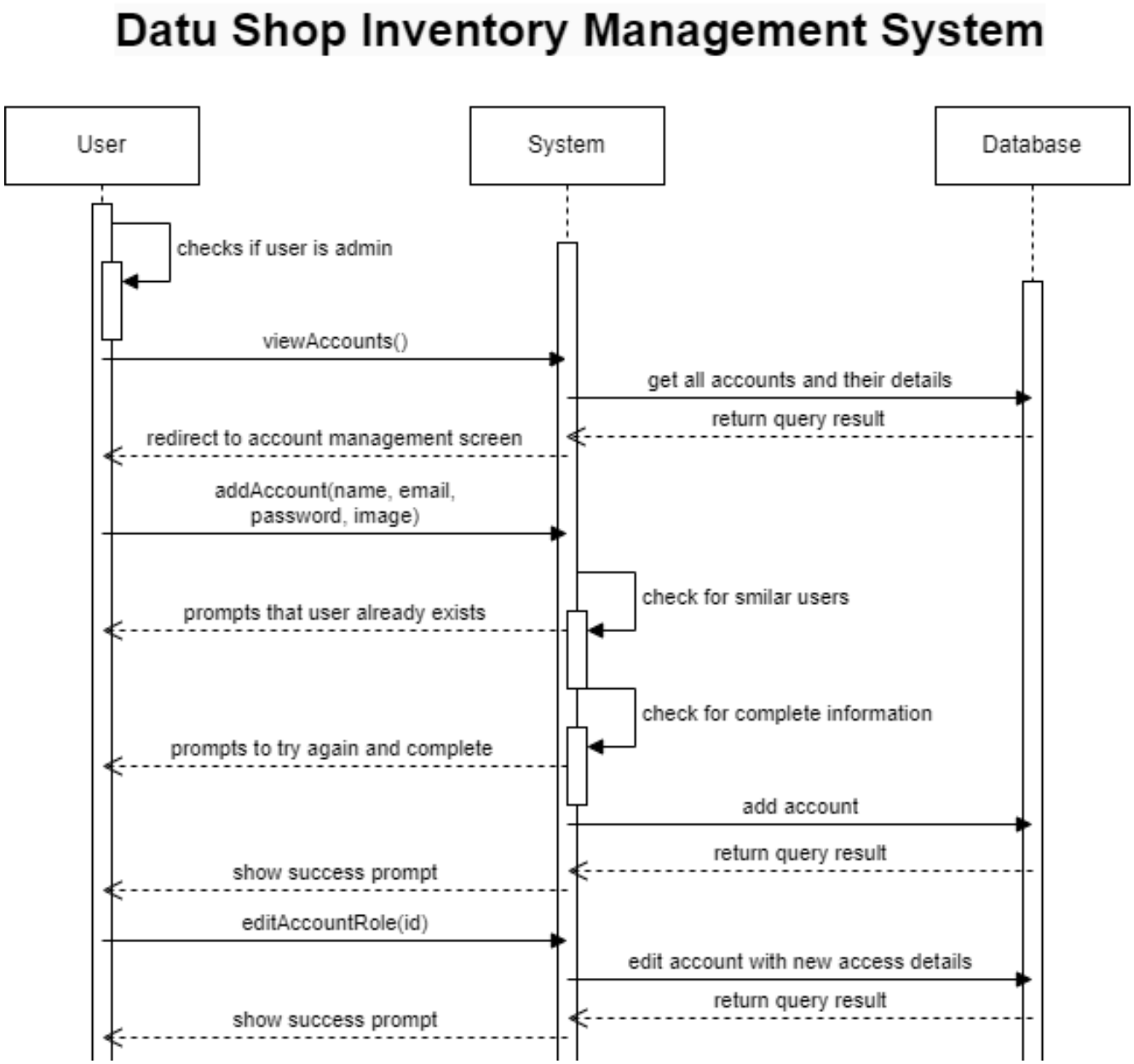

Figure 2.3.6: System Sequence Diagram: Account Management Part 1

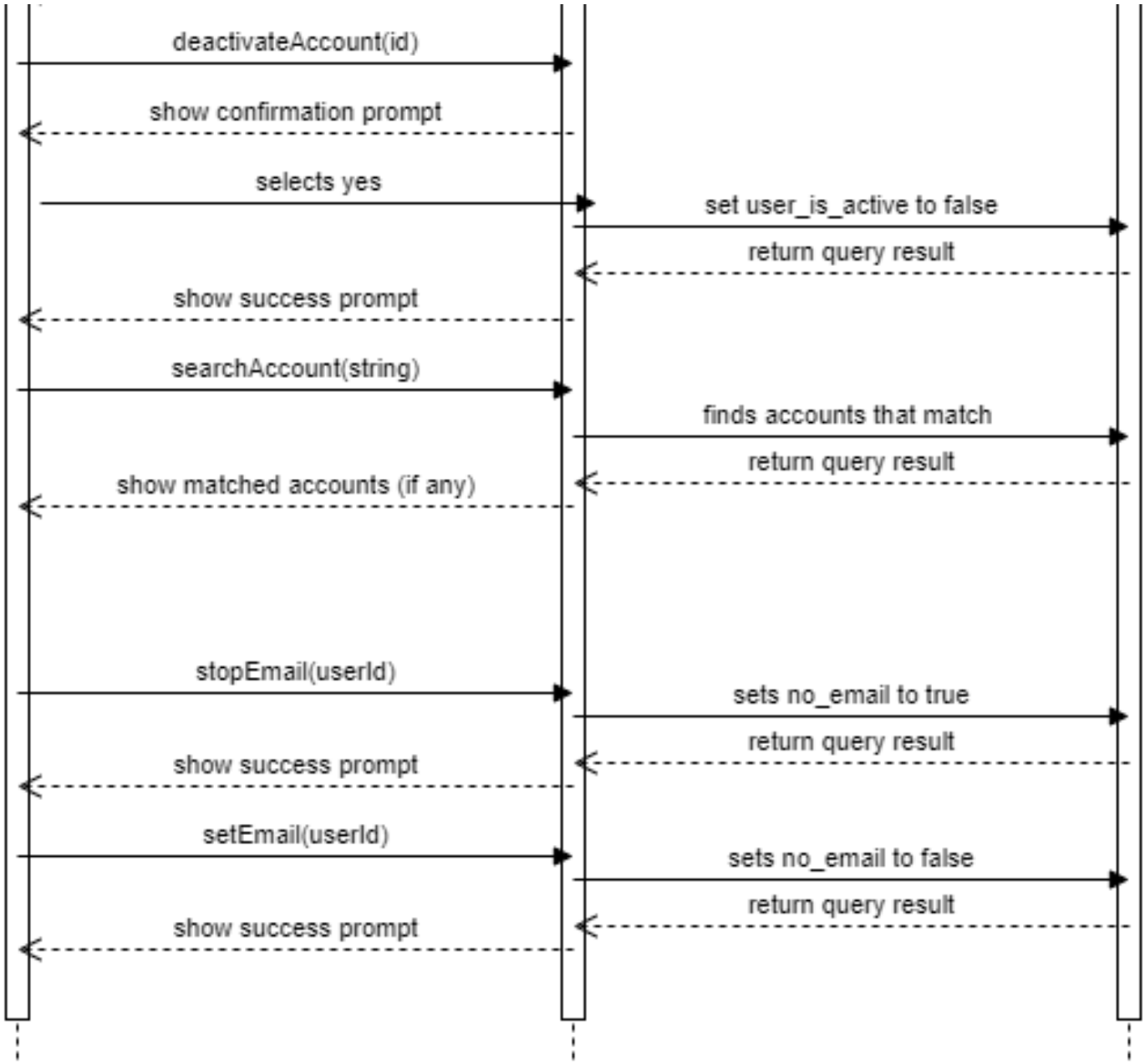

Figure 2.3.7: System Sequence Diagram: Account Management Part 2

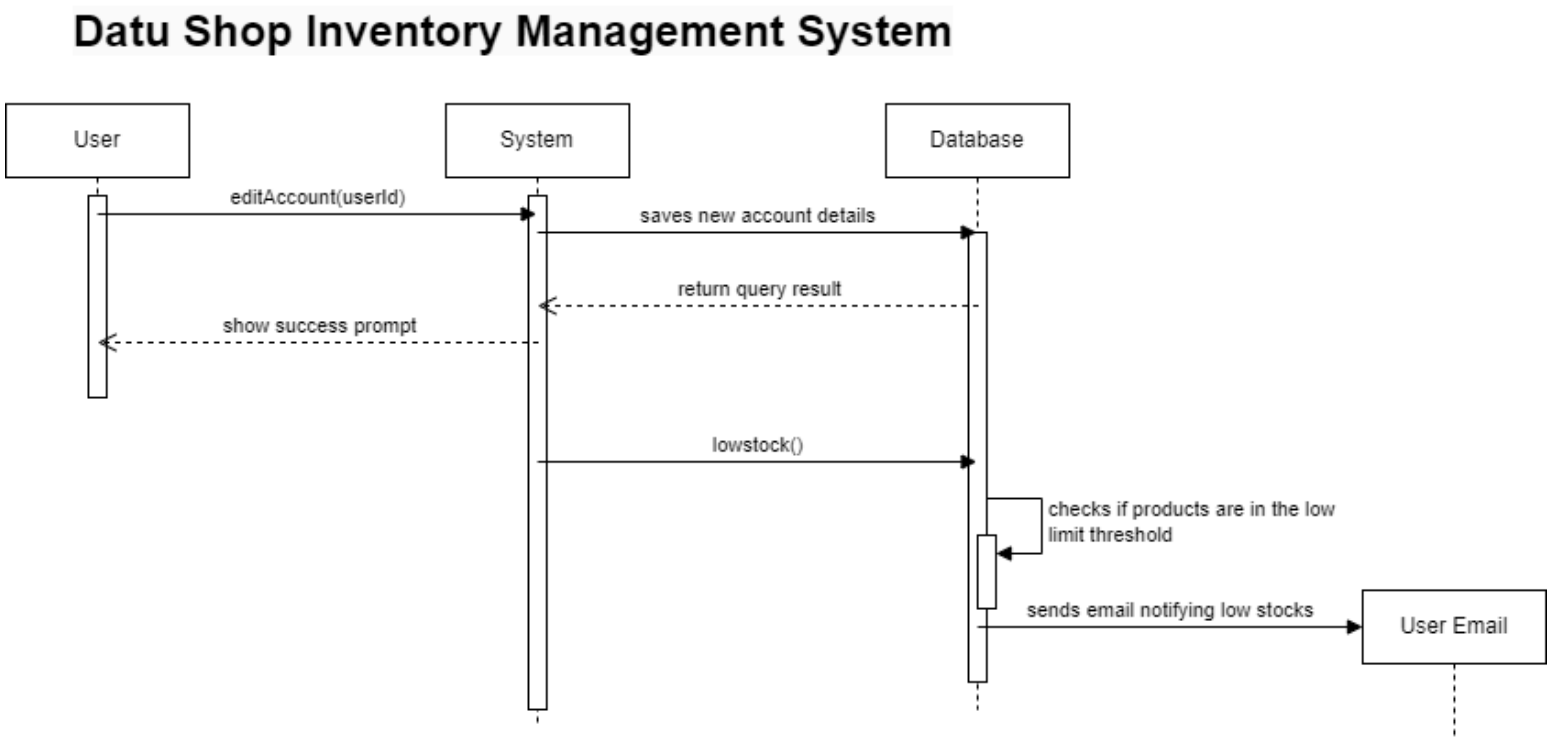

Figure 2.3.8: System Sequence Diagram: Low Stock Notification and Edit Account

3. Data Design

3.1 Data Description

The system's data structure is organized into four interrelated tables: Product, Supplier, Transaction, and User. The UML diagram illustrates the relationships between these tables and the attributes associated with each.

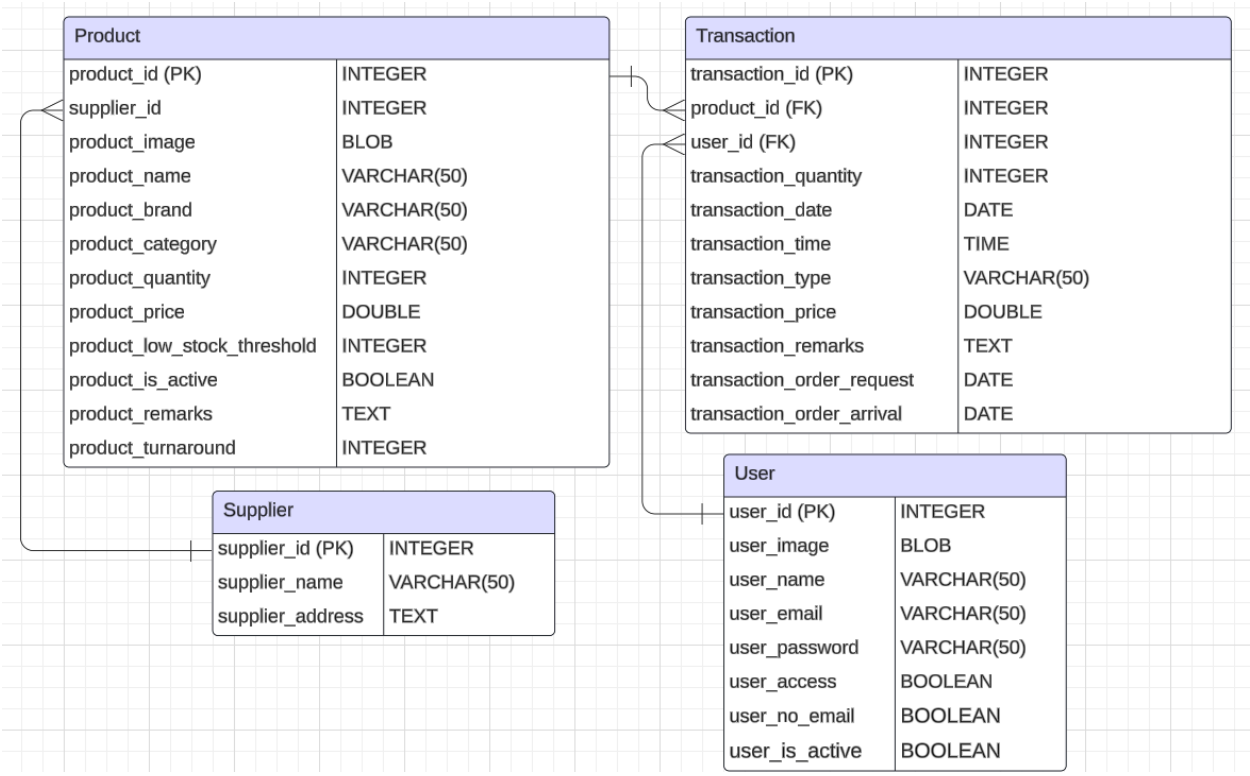

Figure 3.1. Entity Relationship Diagram

3.2 Data Dictionary

Table 3.1 Data Dictionary Table

Product				
Field Name	Data Type	Field Length	Constraint	Description
product_id	INTEGER		PRIMARY KEY NOT NULL	Primary key of the table
supplier_id	INTEGER		FOREIGN KEY NOT NULL	From supplier table
product_image	BLOB		NOT NULL	Image of the product

product_name	VARCHAR	50	NOT NULL	Name of the product
product_brand	VARCHAR	50	NOT NULL	Brand of the product
product_category	VARCHAR	50	NOT NULL	Category of the product
product_quantity	INTEGER		NOT NULL	Quantity of the product
product_price	DOUBLE		NOT NULL	Average price of the product
product_low_stock_threshold	INTEGER		NOT NULL	The low stock threshold of the product
product_is_active	BOOLEAN		NOT NULL	The flag whether the product is still active or deleted.
product_remarks	TEXT		NULLABLE	Details of the product
product_turnaround	INTEGER		NOT NULL	The turnaround time of the product's order_sent and order_received

Supplier				
Field Name	Data Type	Field Length	Constraint	Description

supplier_id	INTEGER		PRIMARY KEY NOT NULL	Primary key of the table
supplier_name	VARCHAR	50	NOT NULL	Name of the supplier
supplier_address	TEXT		NOT NULL	Address of the supplier

Transaction				
Field Name	Data Type	Field Length	Constraint	Description
transaction_id	INTEGER		PRIMARY KEY NOT NULL	Primary key of the table
product_id	INTEGER		FOREIGN KEY NOT NULL	From Product table
user_id	INTEGER		FOREIGN KEY NOT NULL	From user table
transaction_quantity	INTEGER		NOT NULL	Number of stocks being transacted
transaction_date	DATE	YYYY-MM-DD	NOT NULL	Date of the transaction
transaction_time	TIME	HH-MM-SS	NOT NULL	Time of the transaction
transaction_type	VARCHAR	50	NOT NULL	Type of the transaction
transaction_price	DOUBLE		NOT NULL	Price of the product sold

transaction_remarks	TEXT		NULLABLE	Details of the transaction
transaction_order_request	DATE		NULLABLE	Date when the order is requested
transaction_order_arrival	DATE		NULLABLE	Date when the order arrived

User				
Field Name	Data Type	Field Length	Constraint	Description
user_id	INTEGER		PRIMARY KEY NOT NULL	Primary key of the table
user_image	BLOB		NOT NULL	Image of the user
user_name	VARCHAR	50	NOT NULL	Name of the user
user_email	VARCHAR	50	NOT NULL	Email of the user
user_password	VARCHAR	50	NOT NULL	Password of the user
user_access	VARCHAR	50	NOT NUL L	The level of access granted to the user
user_no_email	BOOLEAN		NOT NULL	The flag whether to turn on email notifications or

				not.
user_is_active	BOOLEAN		NOT NULL	The flag whether the user is a valid user or not.

4. Component Design

4.1 Session Management

Session Management controls access to the system by authenticating users' login and providing a secure logout mechanism. This is also responsible for identifying and differentiating the user's roles in accessing the IMS. Different functionalities are available for different user roles.

```
login(String email, String password)
    check the User database if the email exists
    if the email exists and is not deactivated:
        get encrypted password and role from the User database
        decrypt the encrypted password
        if password matches with the decrypted password:
            go to Homepage for the specified role
        else:
            prompt a login error
    else:
        prompt a login error

forgotPassword(String email)
    check if the email exists in the User database
        if the email exists and is not deactivated:
            generate confirmation_code
            save confirmation_code
            send confirmation_code to email
        else:
```

```
                    prompt an error

confirmCode(String confirmation_code, String email)
    check if confirmation_code matches with the saved confirmation_code
        if match:
            go to change password page
        else:
            prompt error

changePassword(String new_password, String email)
        encrypt new_password
        edit password for email

logout()
    close the active session
    go to login page
```

4.2 Inventory Management

Inventory Management holds the responsibility of displaying, adding, modifying, and removing items from the database, ensuring accurate and up-to-date records.

```
displayRecords()
    get records that are not set to product_isSelling from the Product
    database using a query
    store records in a list
    for every record in records:
        display record in column


addProduct(String productName, String productCategory, String supplier,
Double price, String brand, String remarks, Integer quantity, Date
orderRequest, Date orderArrival)
    If product_name and supplier != from the Product database:
```

```
                    turnaroundTime = orderRequest - orderArrival, stored as number
of days
                    add product to Product database
                    call addTransactions(transaction_type, user_id, product_id,
                    transaction_quantity, transaction_price, transaction_remarks)
            Else:
                    send a warning message
                    If product_category, price  and brand != form the Product database
                            add product to the Product database
                            call addTransactions(transaction_type, user_id, product_id,
                            transaction_quantity, transaction_price,
                            transaction_remarks, transaction_order_request,
                            transaction_order_arrival)
                    else:
                            prompt an error
```

addStock(Integer productId, Integer userId, Integer quantity, Double price, Date orderRequest, Date orderArrival, String remarks)

```
            get product_stock and product_price from the row of product_id in the
            Product database
            turnaroundTime = orderRequest - orderArrival, stored as number of days
            updatedQuantity = product_stock + quantity
            update row of product_id with updatedQuantity in Product database
            call addTransactions(transaction_type, user_id, product_id,
transaction_quantity, transaction_price, transaction_remarks, transaction_order_request,
transaction_order_arrival)
```

removeStock(Integer productId, Integer userId, Integer quantity, Double price, String remarks)

```
            get product_stock from the row of product_id in the Product database
            updatedStock = product_stock - quantity
            update row of product_id with updatedStock in Product database
            call addTransactions(transaction_type, user_id, product_id,
            transaction_quantity, transaction_remarks)
```

sellStock(Integer productId, Integer quantity, Integer userId, Double transactionPrice, String transactionRemarks)

```
get product with product_id from Product database
    if (product_quantity > 0) and (product_quantity - quantity >= 0)
        product_quantity = product_quantity - quantity
        update row of product_id with product_quantity in the
        Product database
        Call addTransaction(product_type, user_id, product_id,
        transaction_quantity, transaction_price,
        transaction_remarks)
```

returnStock(Integer productId, Integer userId, Integer quantity, Double price, String remarks)

```
get product_quantity of product_id in Product database
updatedQuantity = product_quantity + quantity
update row of product_id with product_quantity in the Product database
Call addTransaction(product_type, user_id, product_id,
transaction_quantity, transaction_price, transaction_remarks)
```

editProduct(Integer productId)

```
if product_id exists in the database:
    go to edit_page that shows item details
    if save button is pressed:
        if inputs are correct:
            update database
            call addTransactions(transaction_type, user_id,
            product_id, transaction_quantity, transaction_price,
            transaction_remarks)
            go to Homepage
        else:
            prompt error page with error message
else:
```

```
            prompt error page with error message

deleteProduct(Integer productId)
    if product_id exists in the database:
        show pop-up confirmation for deleting
        if confirmed:
            set isSelling to false
            call addTransactions(transaction_type, user_id, product_id,
            transaction_quantity, transaction_price,
            transaction_remarks)
    else:
        prompt error page with error message
```

4.3 Search and Filter

Search and Filter allows the users to efficiently find specific products within the inventory, applying filters per category and accessing comprehensive details and status for effective inventory management.

searchInventory(String searchString)

SQL query using searchString for the Where clause

searchTransaction(String searchString)

SQL query using searchString for the Where clause

searchAccounts(String searchString)

SQL query using searchString for the Where clause

filter(String filterString)

SQL query using filterString for the Where clause

4.4 Transaction History

Transaction History showcases the transactions of parts happening in the store. An addition of parts means that a possible restock or a possible return of a part has happened. A subtraction of parts means a sold part or a part gets thrown out due to wear and tear. This will also track who made changes in the inventory.

addTransaction(String transactionType, Integer userId, Integer, productId, Integer transactionQuantity, Double transactionPrice, String transactionRemarks)

get date and time

add all relevant variables to the Transaction database

transactionHistory()

If user or owner is in session:

get transactions from the Transaction database

print transactions

else:

Redirects to the login page

transactionTimeFramed(DateTime start, DateTime end)

If user or owner is in session:

get transactions from the Transaction database from start to end

print transactions

else:

Redirects to the login page

transactionFiltered(String category)

If category is Sold

get all sold transactions in the Transaction database

print transactions

If category is removed

get removed transactions in the Transaction database

print transactions

If category is added

get all added transactions in the Transaction database

print transactions

4.5 Sales Dashboard

Export Sales exports the number of parts sold by the user in either an Excel or a PDF file in a given timeframe. This helps analyze which parts sell most, given a day, week, or month. This is mainly used to make data-driven decisions on which brand of a part or which category of parts sells the most.

getInventoryCount()

```
    get the total number of merchandise inventory
    get the total number of products in the database
    get the count of all low stock products in the database
    return values
```

getGrossSales()

```
    get all sold products in the Transaction database
    return sum all transaction_price
```

getSalesAnalytics(DateTime start, DateTime end):

```
    get transactions based on start to end datetime
    render transactions
```

getSalesDistributionsPerCategory(DateTime start, DateTime end):

```
    get sales per category available in the database in the given timeframe
    return values
```

exportSales(String fileChoice):

```
    if transactions are rendered:
        take snapshot
        If confirm button is clicked:
            convert image to fileChoice of xlsx or pdf
            prompt export location
            export
```

4.6 Account Management

The Account Management component manages the account system used to filter the role of the users in the system and the login credentials used to access the features

themselves. An owner role can add or deactivate accounts for the system. Owners can also edit roles for accounts in the system.

viewAccounts()

get all of the users
prints the information and a button with an Edit and Delete
button with the user.

addAccount(String name, String email, String password, Blob image)

Prompts the user with the account details through a form
If the Save button is clicked:
    If the form has all required values:
        If no equal information in the database:
            Saves the account information in the database
            Redirect the user back to the View accounts page
        else:
            Prompts the user again
            Prints the message “User already existed”
    else:
        Prompts the user again
        Prints the message “Please input all information for the account.”

editAccountRole(Integer userId)

Gets the ID of the User clicked on the database
If the saved button was clicked:
    The new information overwrites the old information
    Redirects the user to the View accounts page
Else if the cancel button is clicked:
    Redirects the user to the View accounts page

deactivateAccount(Integer userId)

Prompts the user to confirm deactivation
If the confirm button is clicked:

Get row using the user_id

change the role of the particular row to deactivated

Redirect the user to the View Accounts page

Else if the cancel button is clicked:

Redirect the user to the View Accounts page

stopEmail(id userId):

Prompts the user to confirm noEmail

If the confirm button is clicked:

get user through user_id

change user_access to Admin_noEmail

save user_details

Else

Redirects the user to the view accounts page

setEmail(id userId):

Prompts the user to confirm noEmail

If the confirm button is clicked:

get user through user_id

change user_access to Admin

save user_details

Else

Redirects the user to the view accounts page

4.7 Low Stocks Notification

Low Stocks Notification component notifies the user if certain parts are low in stock. An email notification will be the primary medium for this feature.

isLowstock()

If the item is in the low stock threshold:

Gets all the users that will be sent an email to notify that the part is low of stock

else:

No email sent

4.8 Edit Account

All users in the system can edit their account details, like the email address for the low stock notification and login access, as well as the password for login access. Users can also toggle the low-stock notification for their emails.

editAccount(Integer userId)

Gets the ID of the User clicked on the database and prompts the form with the information in the input elements

If the saved button was clicked:

The new information overwrites the old information

Redirects the user to the My Account page

Else if the cancel button is clicked:

Redirects the user to the My Account page

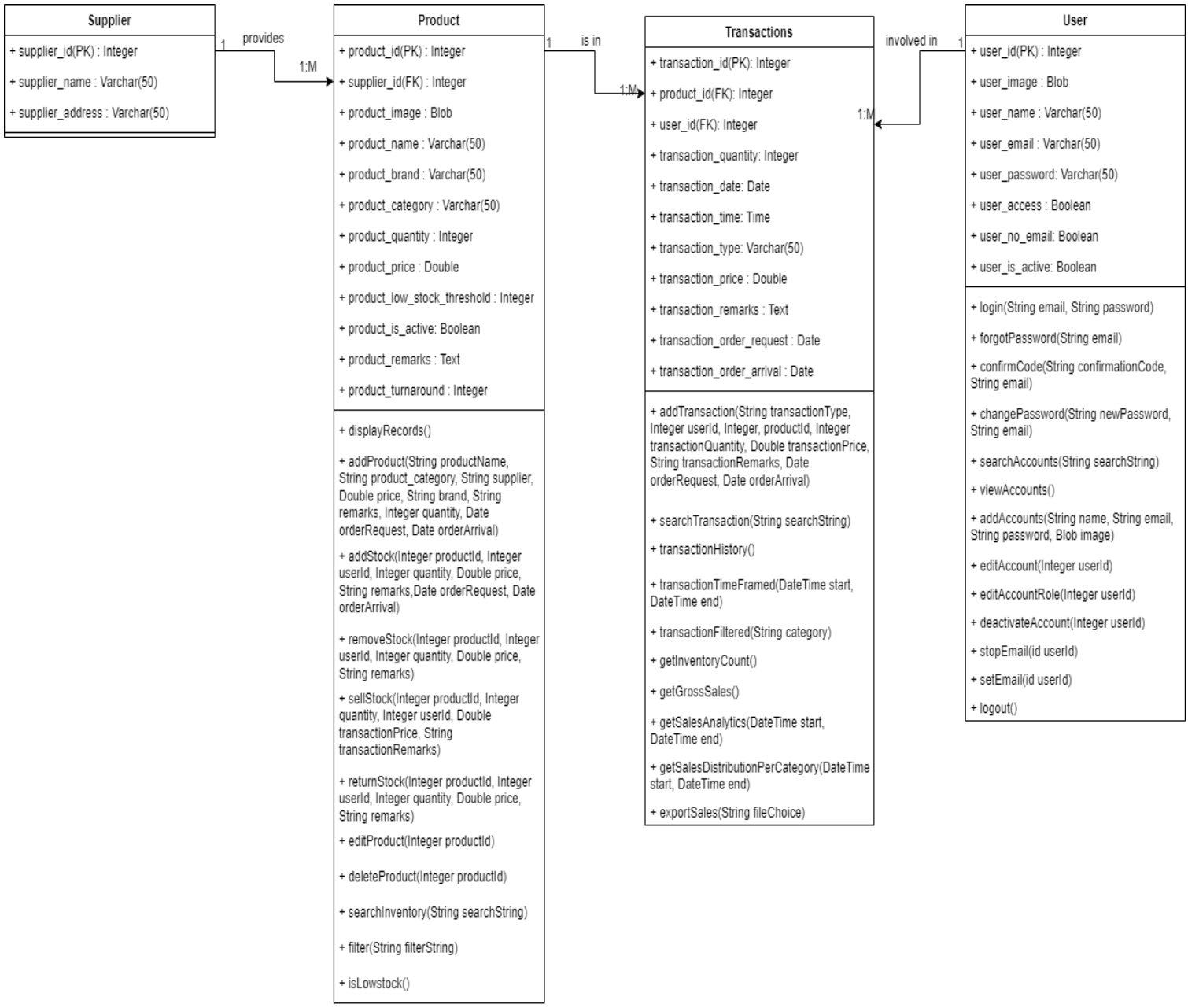

Figure 4.1 Class Diagram

5. Human Interface Design

5.1 Overview of User Interface

The system will start with a login page to prompt users to enter their credentials. There are subfeatures in this section, such as the forgot password feature, in which the user will get a verification code to their email for a password change.

After this section, the user is given the access rights of either employee or owner and is redirected to the Home Page, wherein a small overview of the business in terms of stocks per category is displayed. This is also the first introduction to a sidebar menu to navigate to the system's features. Users can edit their account details by clicking on their profile picture on the top left corner of the pages. Users can modify their email, password, profile picture, and the feature to send low-stock notifications to their saved email address.

The Inventory Tab introduces the Inventory Management System. Products can be added, edited, or deleted. There are actions for adding, selling, returning, or removing stocks for quick and easy actions. When adding, selling, removing, or returning stocks, the user is prompted to enter the quantity, price, and any remarks about the transaction. These are needed for computations in the sales, and the remarks are present in the transaction history tab along with the user doing the action. Selecting the add product button will prompt the user to input the details of the product, such as the name, brand, category, initial quantity, and stock. When adding products or stocks, there would be an input for the date of the order request to the supplier and the date for the order received to update the turnaround time for each product. Viewing a product prompts the overview, where users can see the details of the product as well as the buttons to edit or delete. Editing a product prompts the same form with the initial values, subject to change. It is worth noting that flags are set in place to know whether a product hoping to be added already exists in the system, and there will be prompts to edit the product rather than add it. Deleting the product will not display it in the inventory system, but its details in the transaction history will still be retained for historical reference. Other features, such as a search function for product names, sorting functions for dates, prices, and alphabetical names, and the filter function for categories, are present for quality-of-life features for the inventory system.

The Sales Dashboard is an owner-privileged feature to see the business analytics regarding sales. Added views include a low-stock tab, wherein the list of low-stock items is

displayed for reference on stock buying, and a fast-selling tab, wherein the list of bought items regarding the number of transactions including the item is displayed. An export to PDF or Excel option is available to take a snapshot of the data and save it to the user's device.

The Transaction History tab displays all the transactions labeled as either "Added," "Sold', "Removed," or "Returned". This tab is read-only and cannot be edited, but a search function is present for specific transactions and a category filter for each label mentioned before.

The Account Management page displays all accounts present in the system. Accounts can be added, edited, or deactivated. Adding an account prompts the user to input pertinent details and access of the user in the system. An additional feature includes the checkbox for the notification of low stocks in the email, which can be turned on/off for each user. Viewing the account displays the details of the account as well as the option to edit or deactivate it. Editing an account displays the window to change the role or access of the account in the system. Deactivating an account strips the user of access to the system, but their details remain as they are sensitive information to the transaction history.

The figures below show the mock-ups and the concept of the user interface for the system.

5.2 Screen Mockups

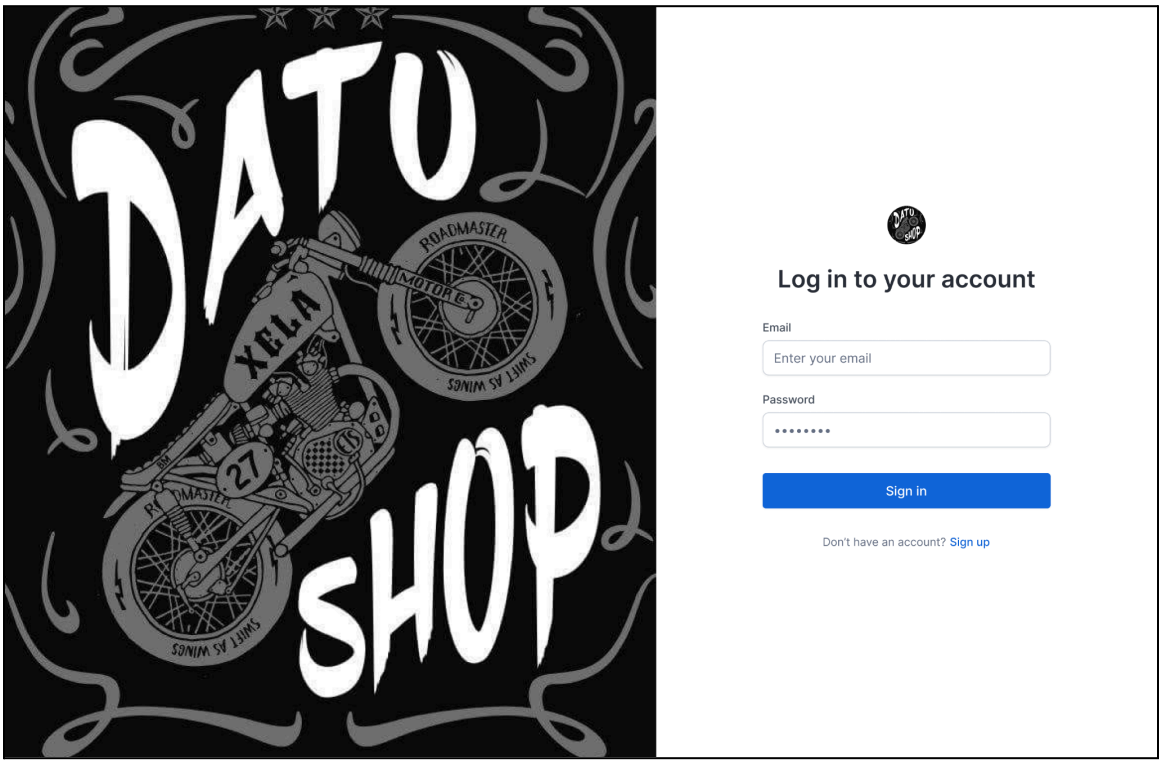

Figure 5.1 Login Screen

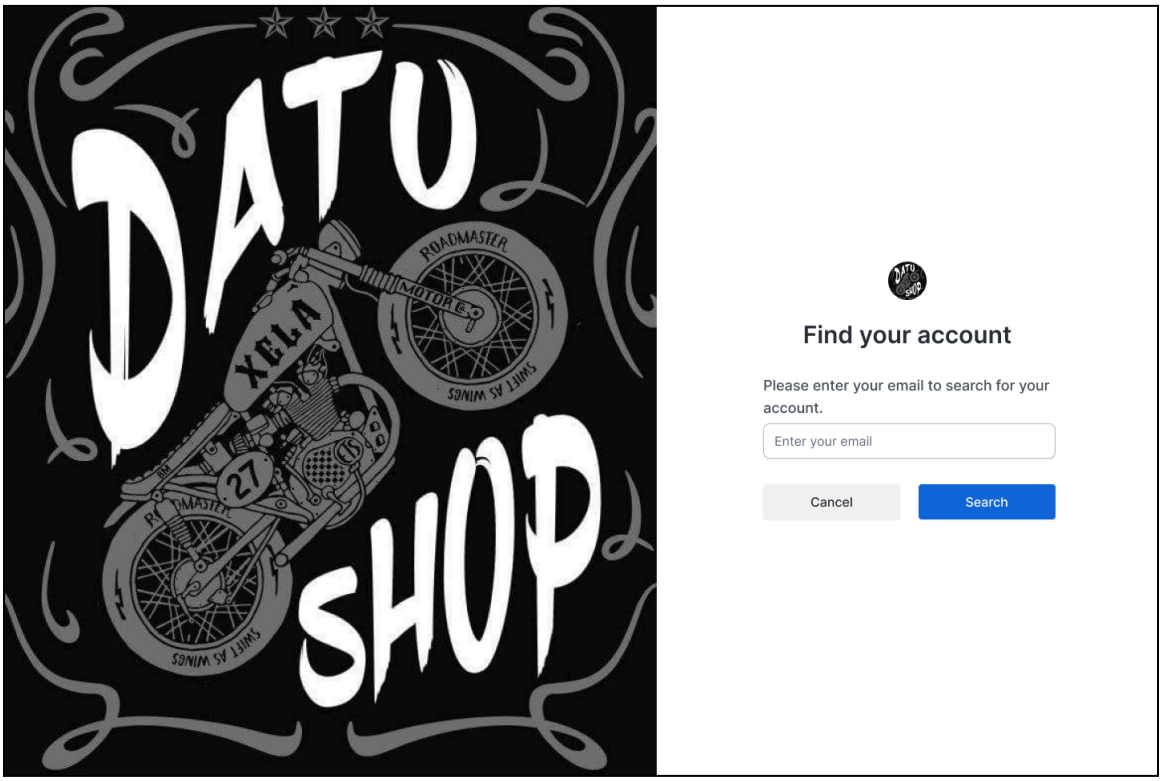

Figure 5.2 Login Screen (Find your account)

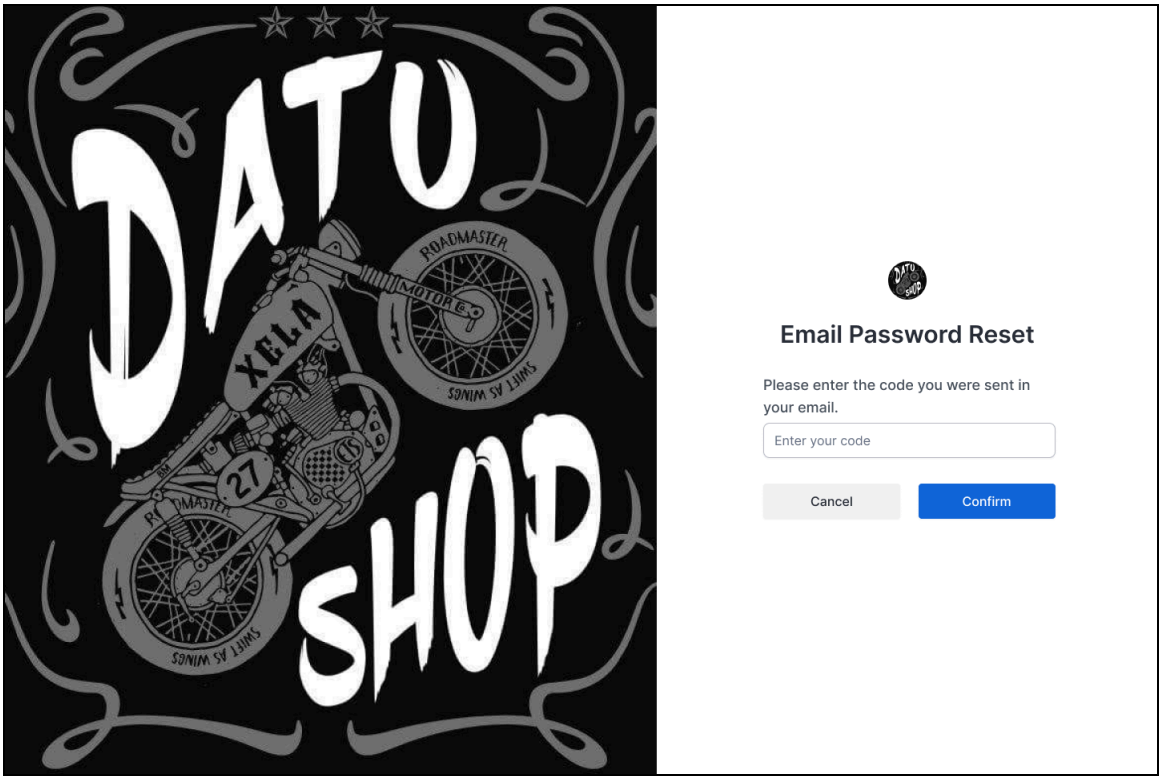

Figure 5.3 Login Screen (Email Password Reset)

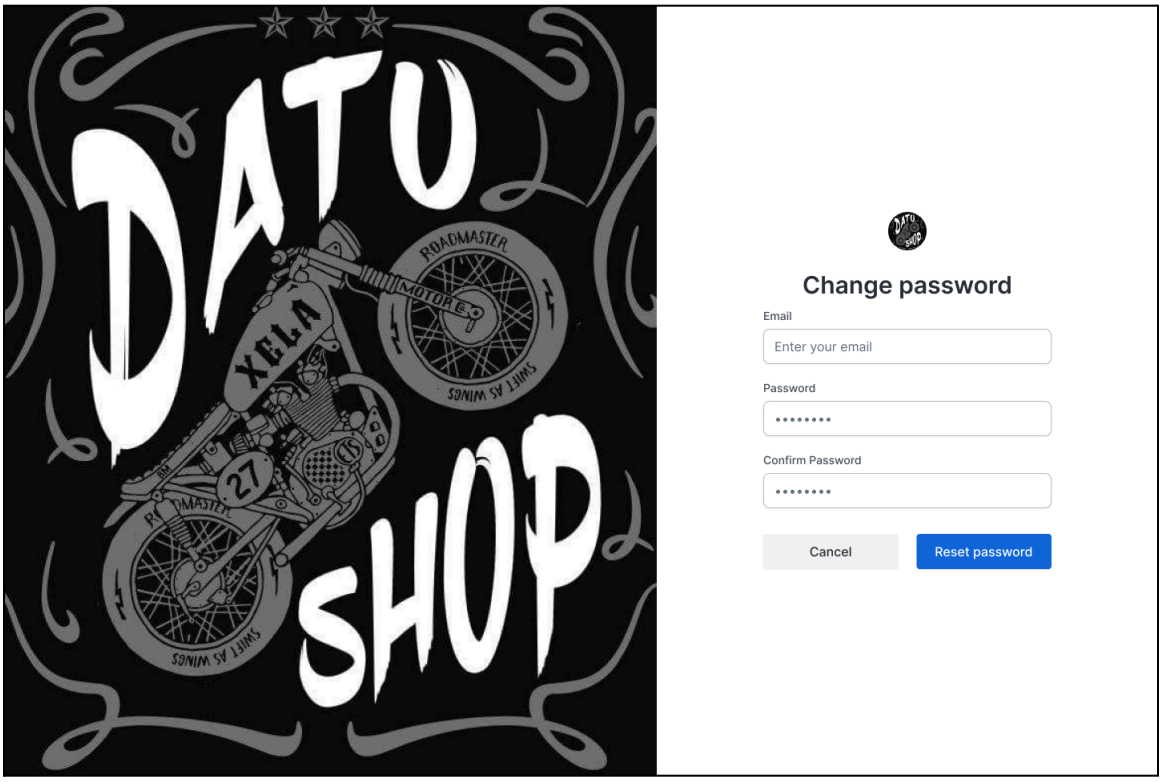

Figure 5.4 Login Screen (Change Password)

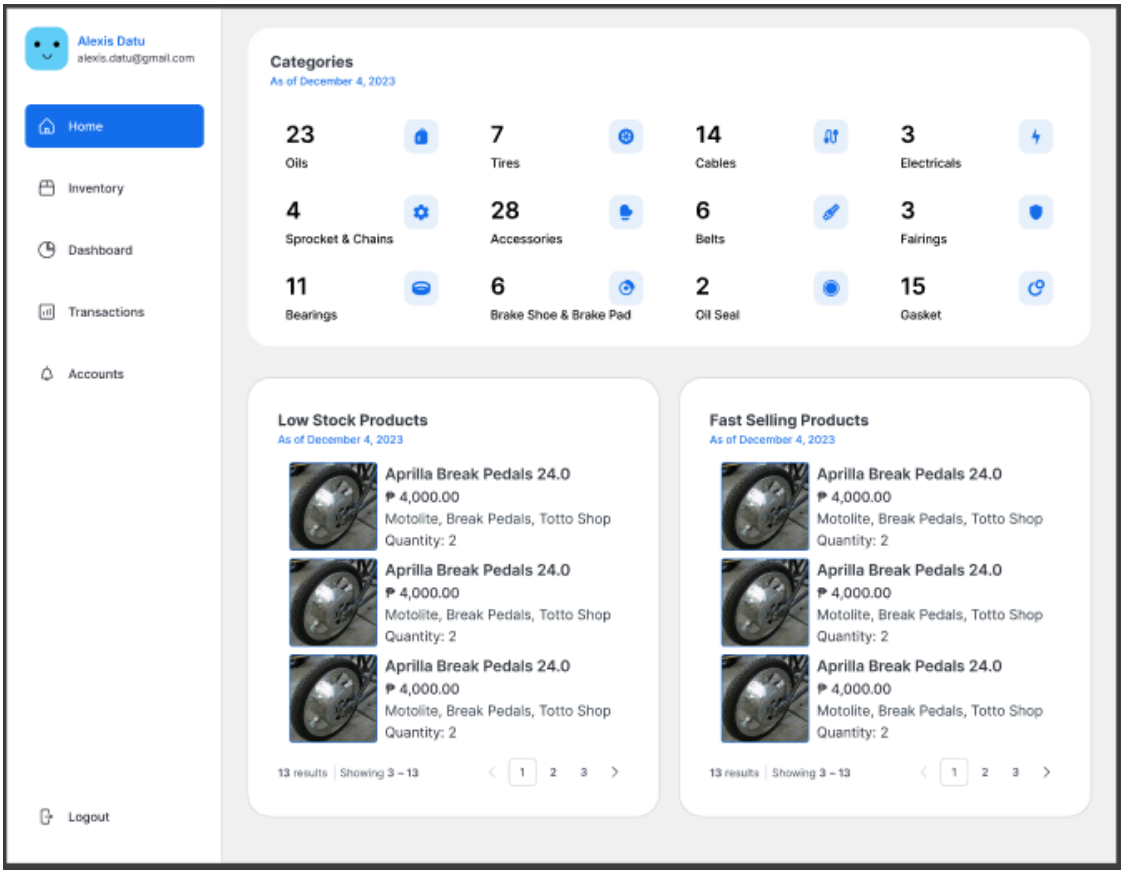

Figure 5.5 Home Screen

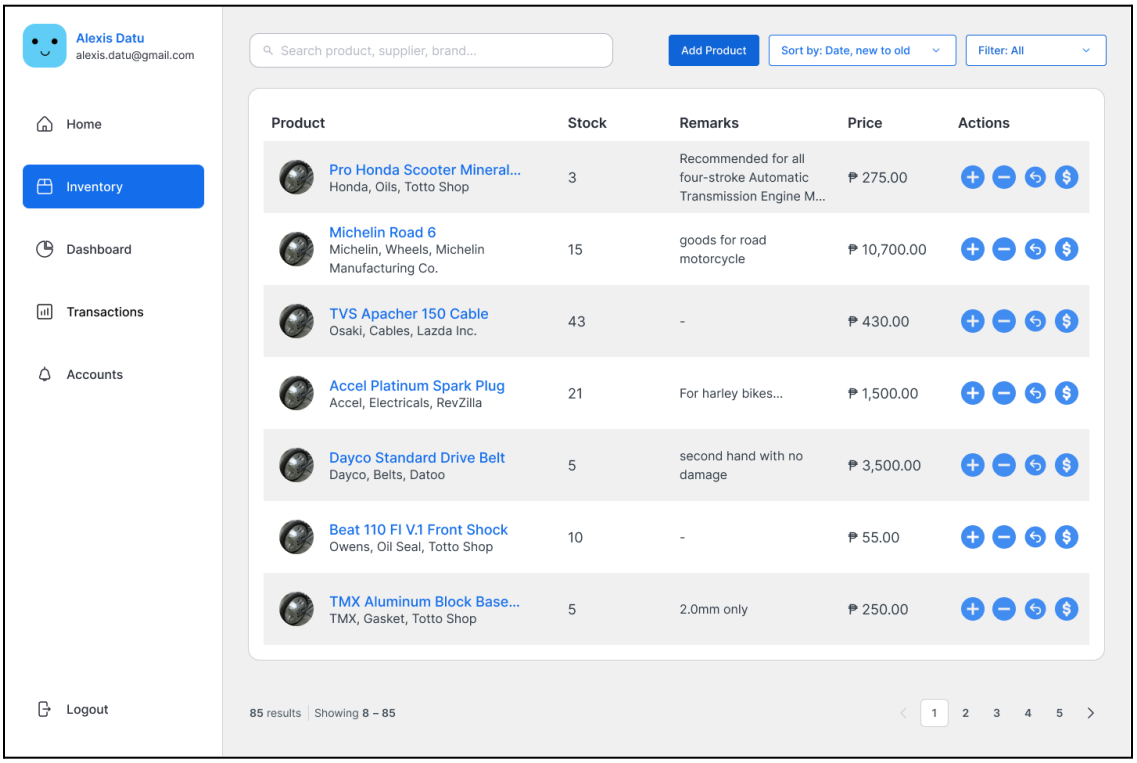

Figure 5.6 Inventory Screen

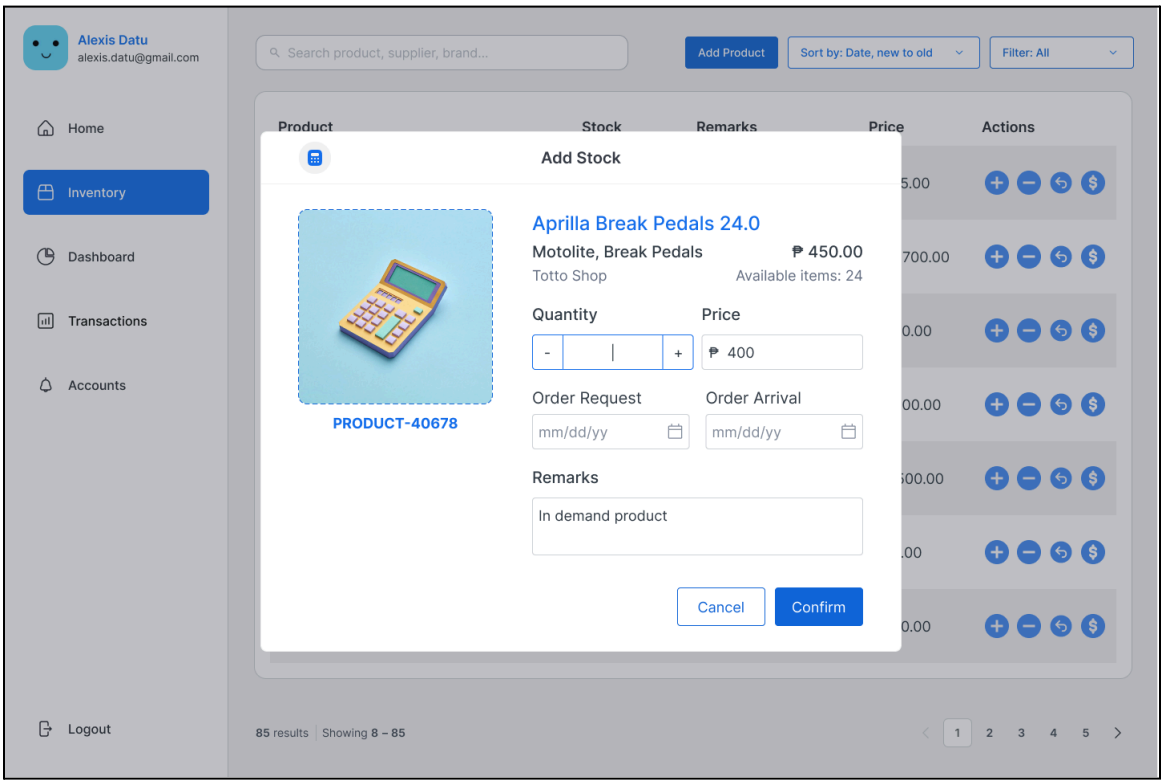

Figure 5.7 Inventory Screen (Add Stock)

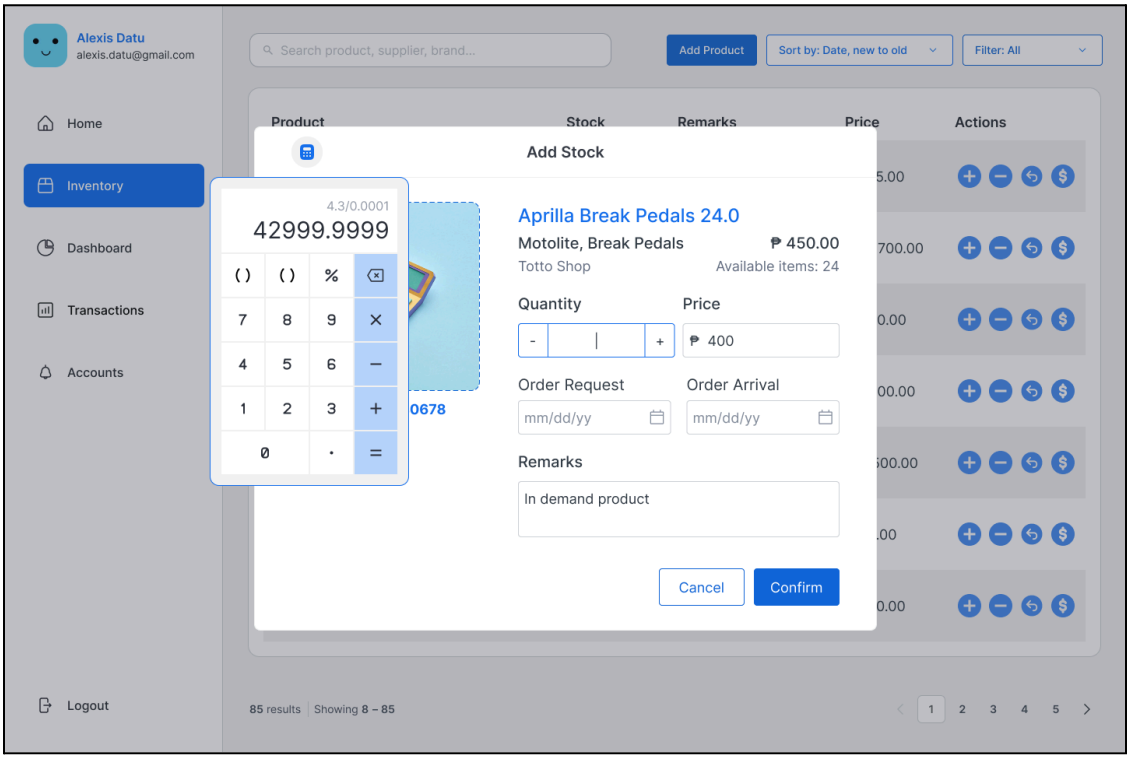

Figure 5.7.1 Inventory Screen (Floating Calculator)

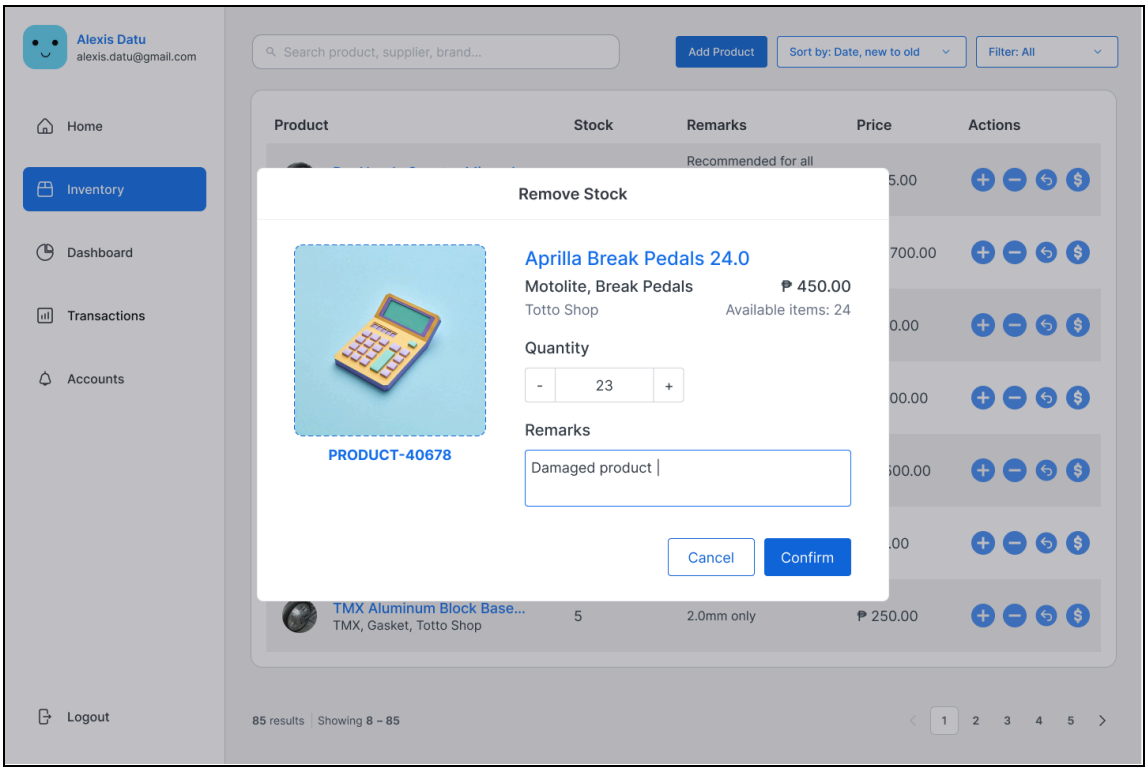

Figure 5.8 Inventory Screen (Remove Stock)

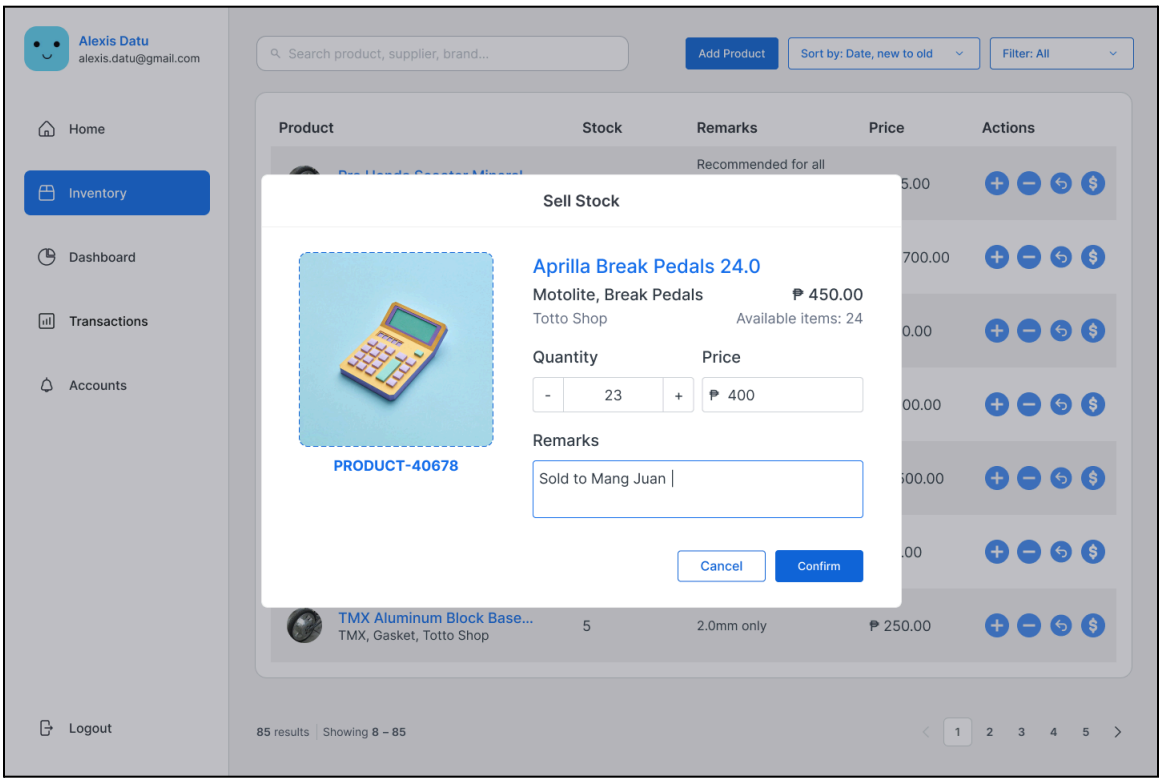

Figure 5.9 Inventory Screen (Sell Stock)

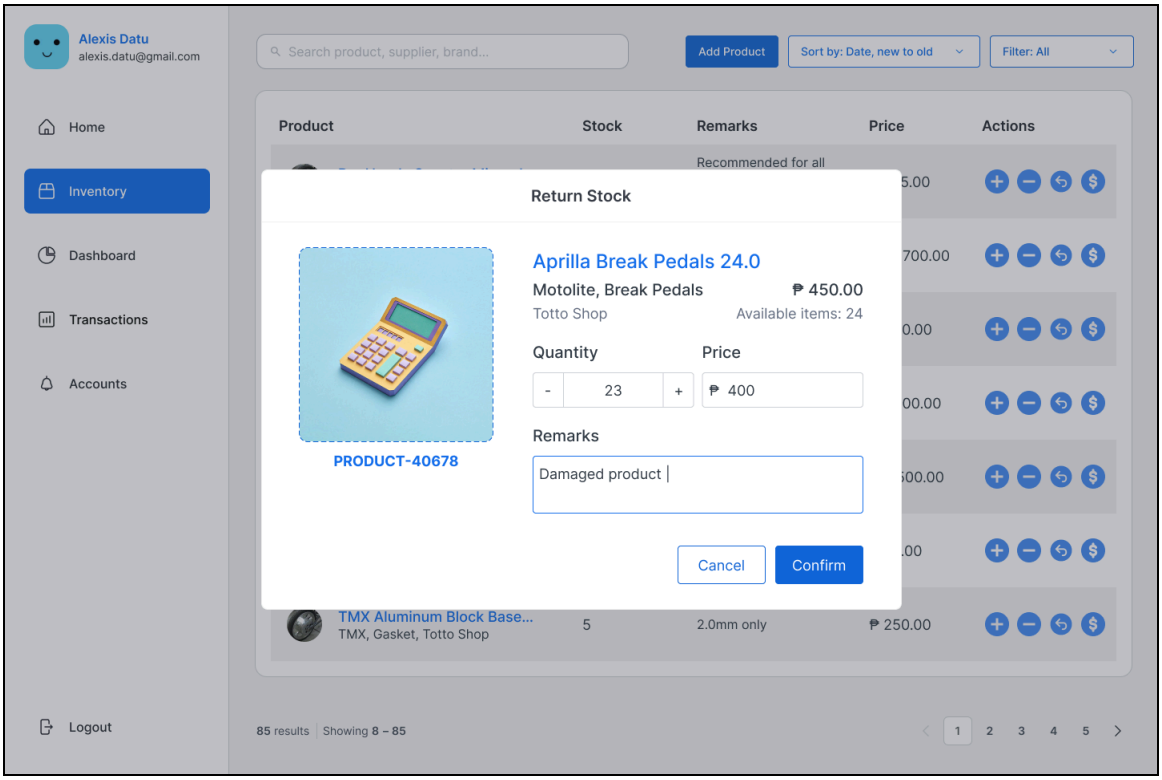

Figure 5.10 Inventory Screen (Return Stock)

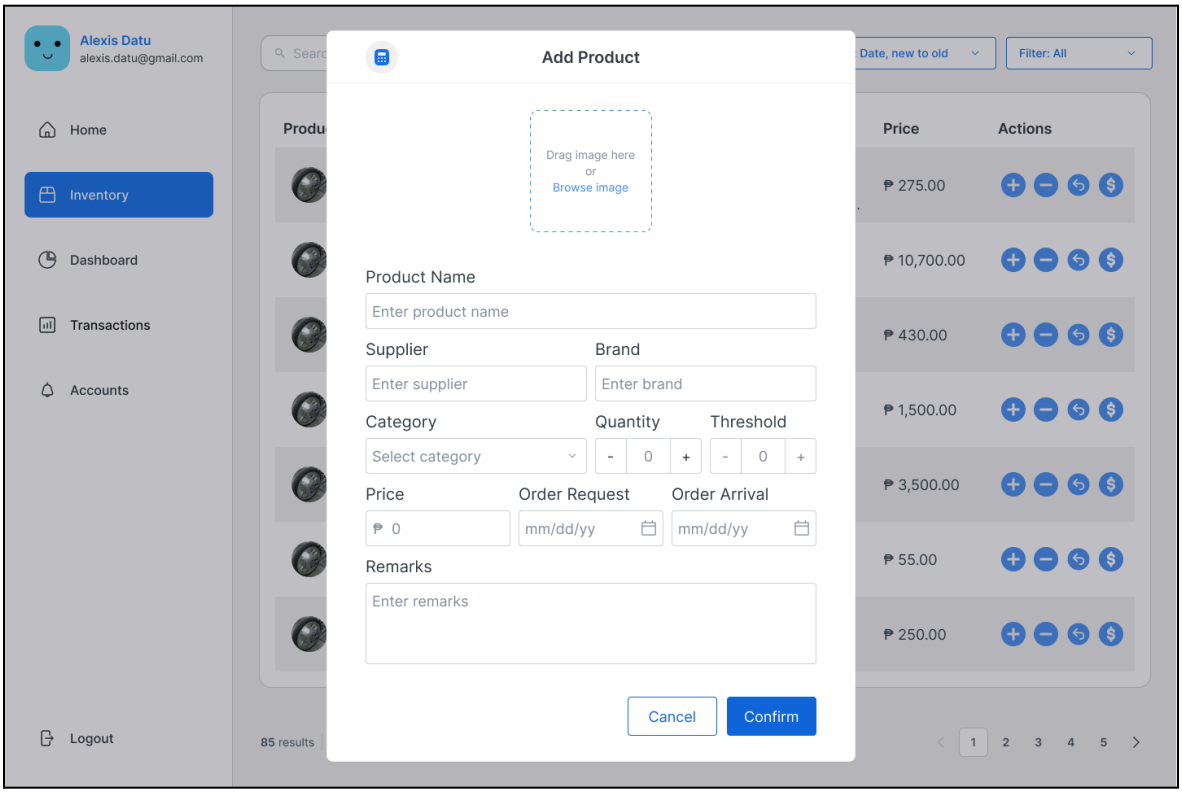

Figure 5.11 Inventory Screen (Add Product)

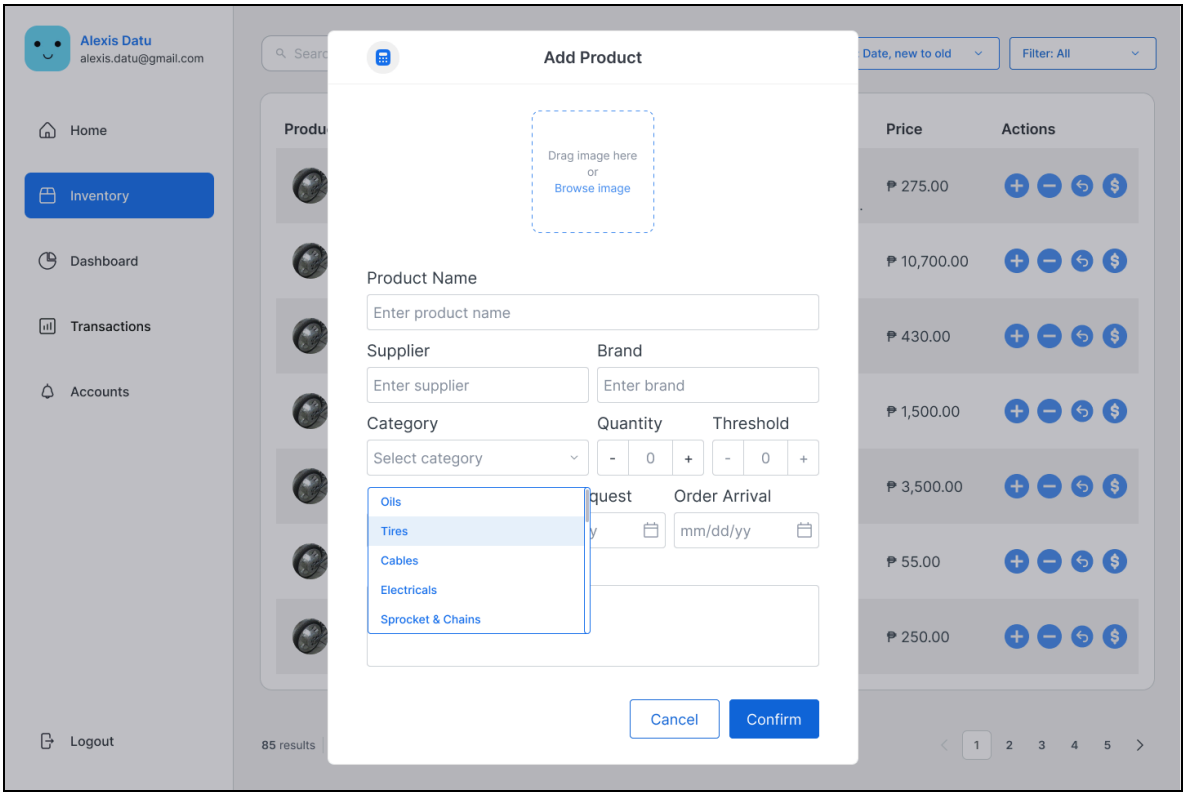

Figure 5.12 Inventory Screen (Categories)

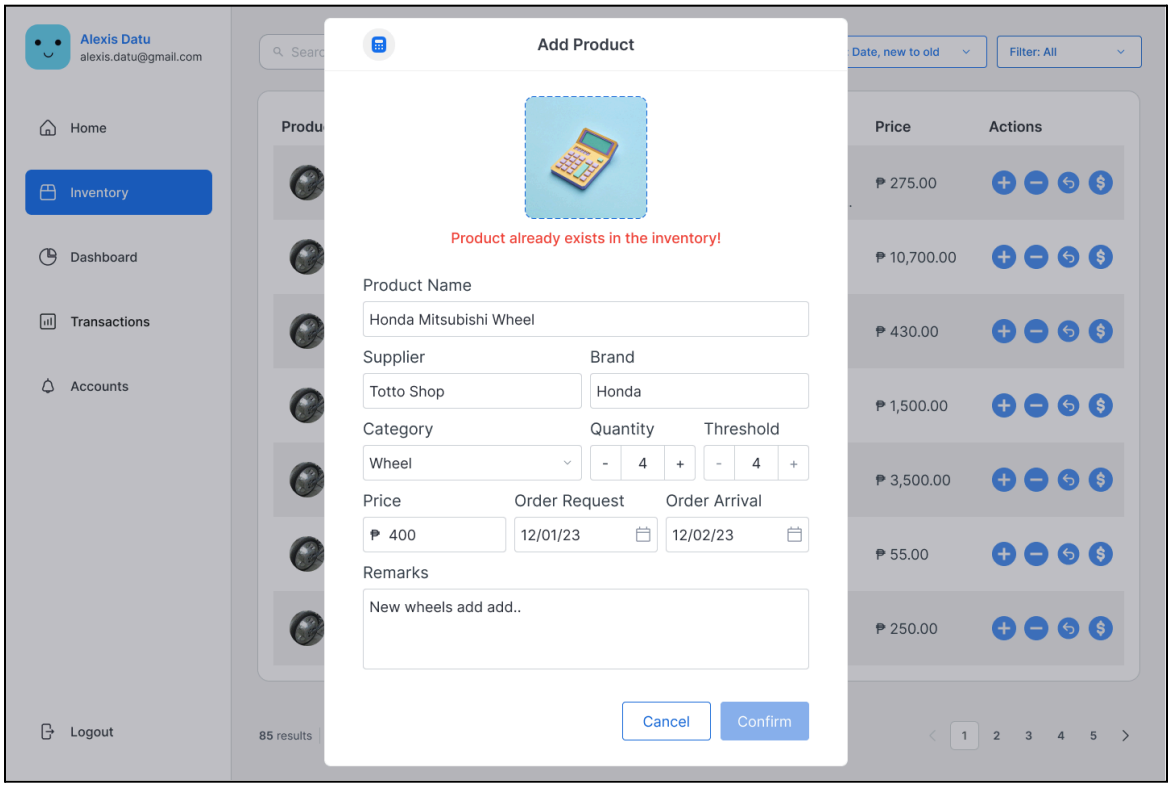

Figure 5.13 Inventory Screen (Add Product Warning)

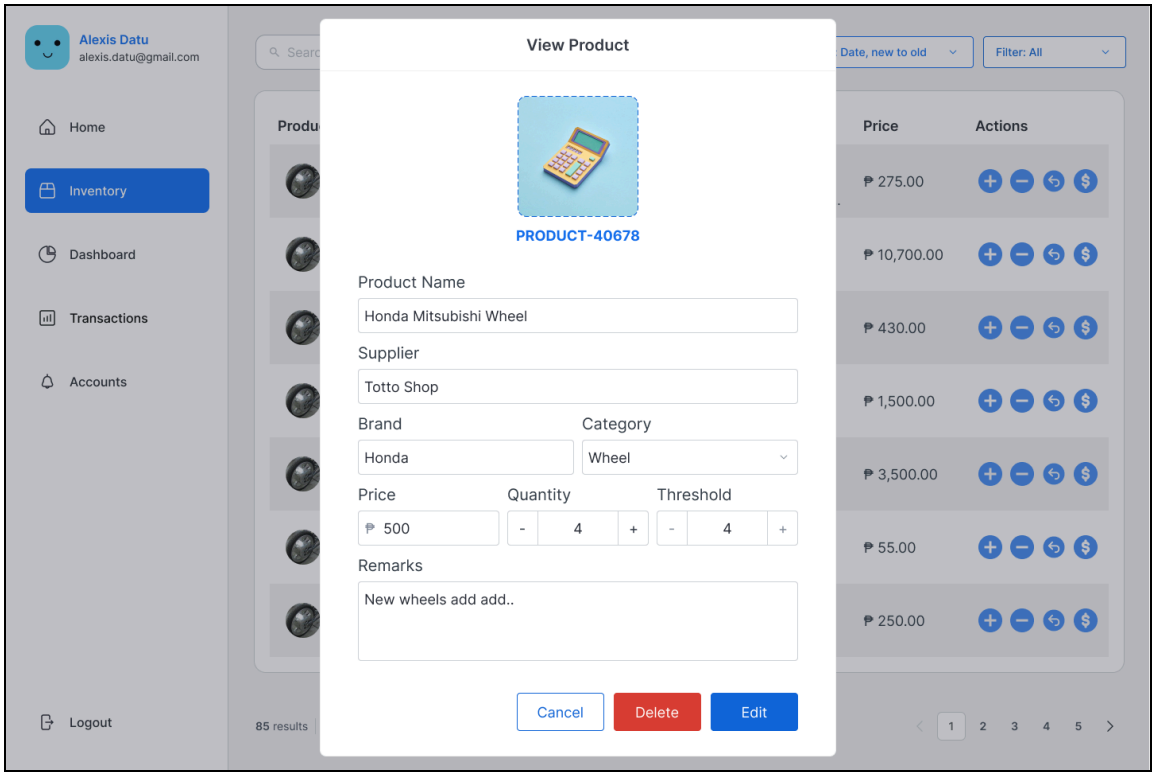

Figure 5.14 Inventory Screen (View Product)

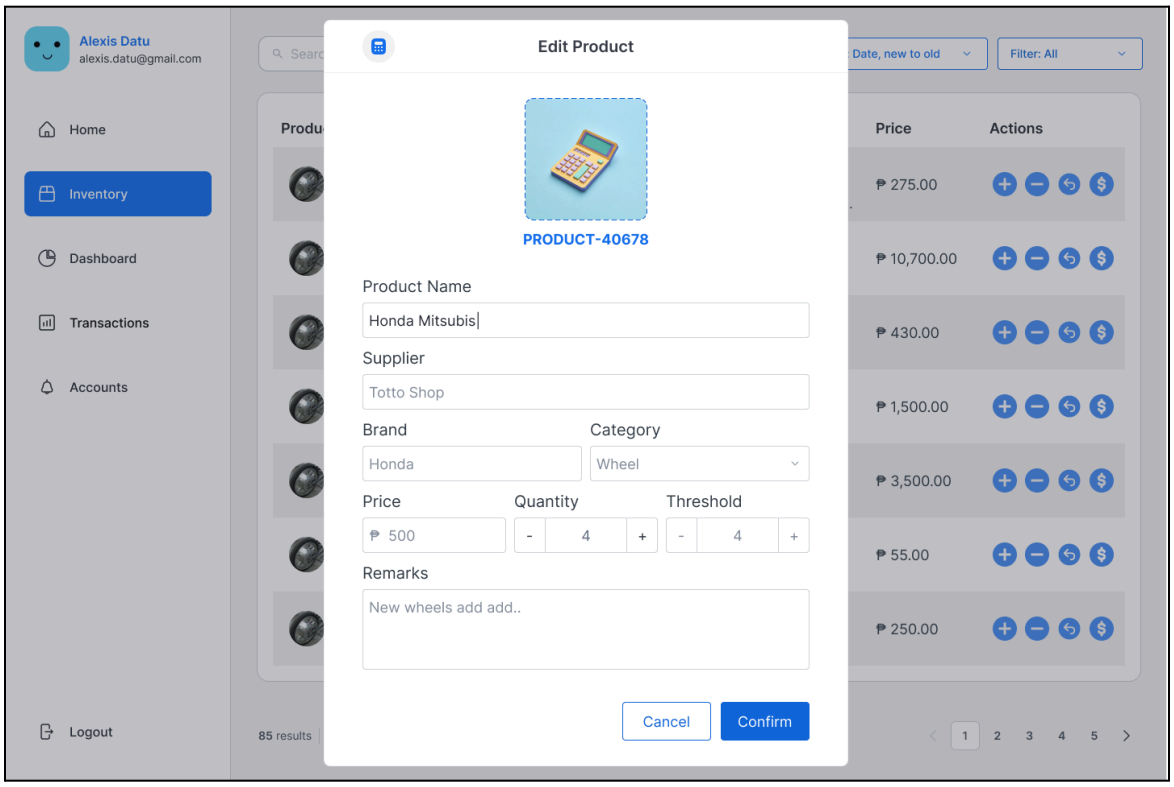

Figure 5.15 Inventory Screen (Edit Product)

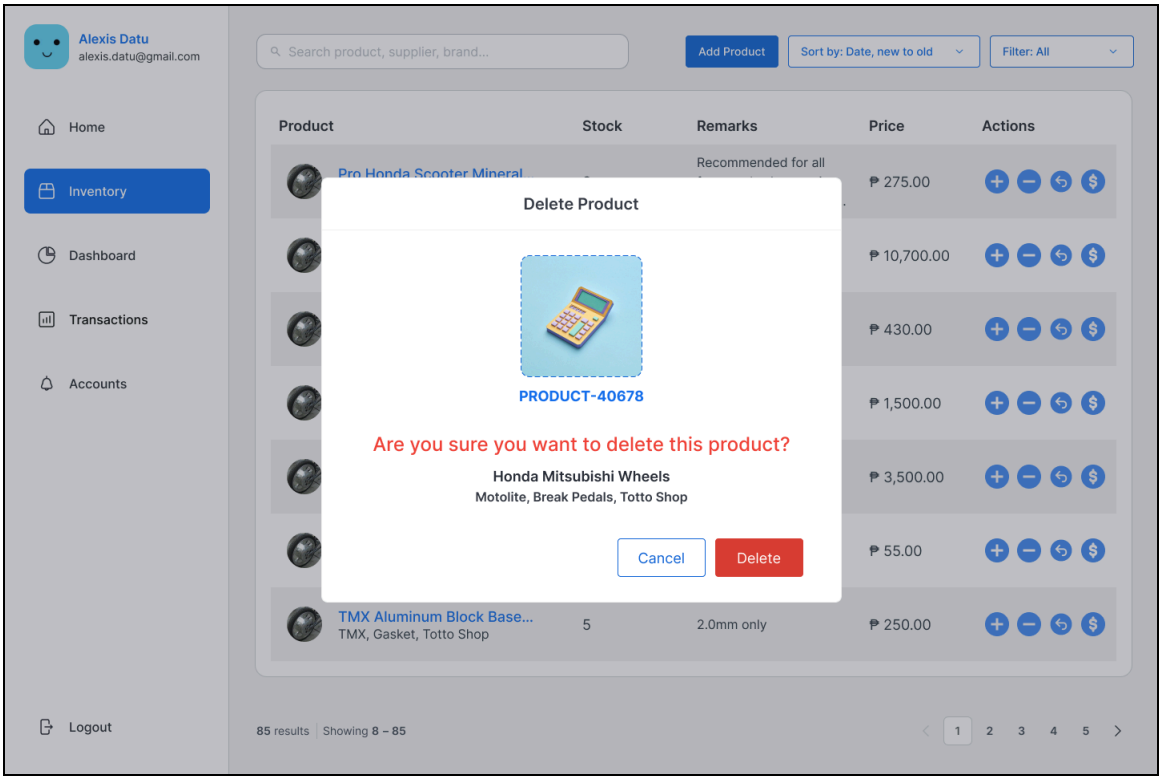

Figure 5.16 Inventory Screen (Delete Product)

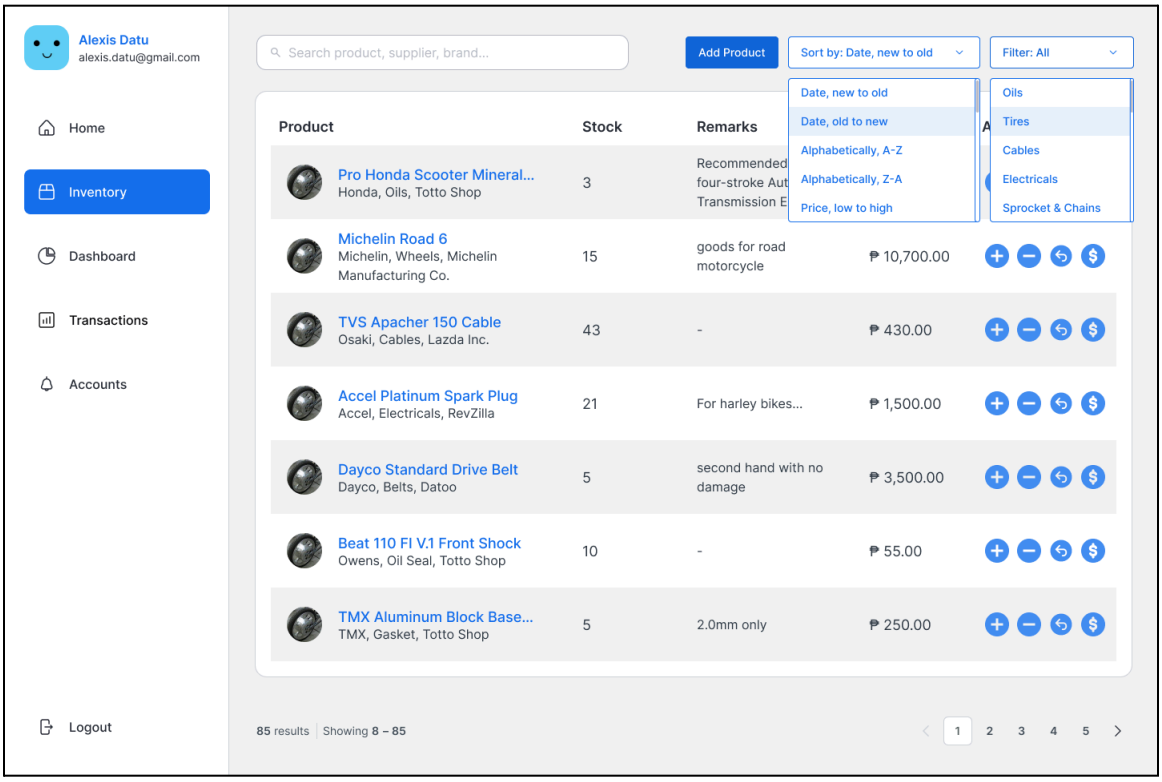

Figure 5.17 Inventory Screen (Sort and Filter)

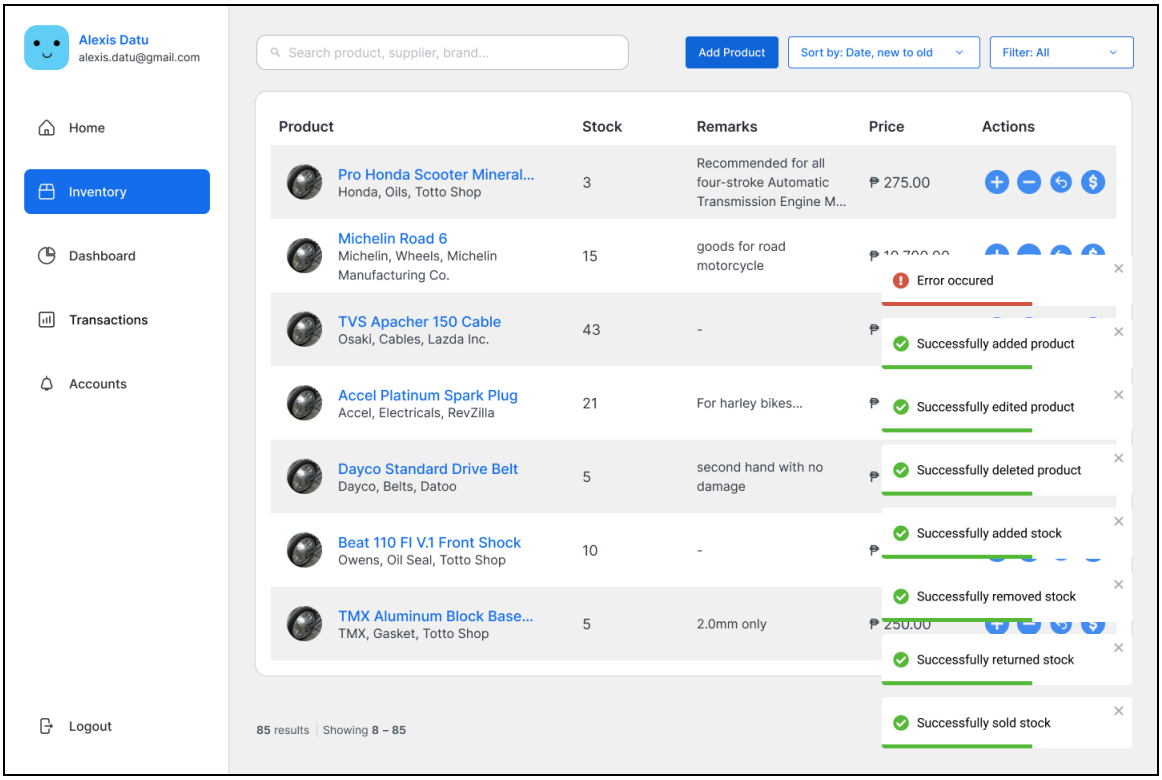

Figure 5.18 Inventory Screen (Toast Messages)

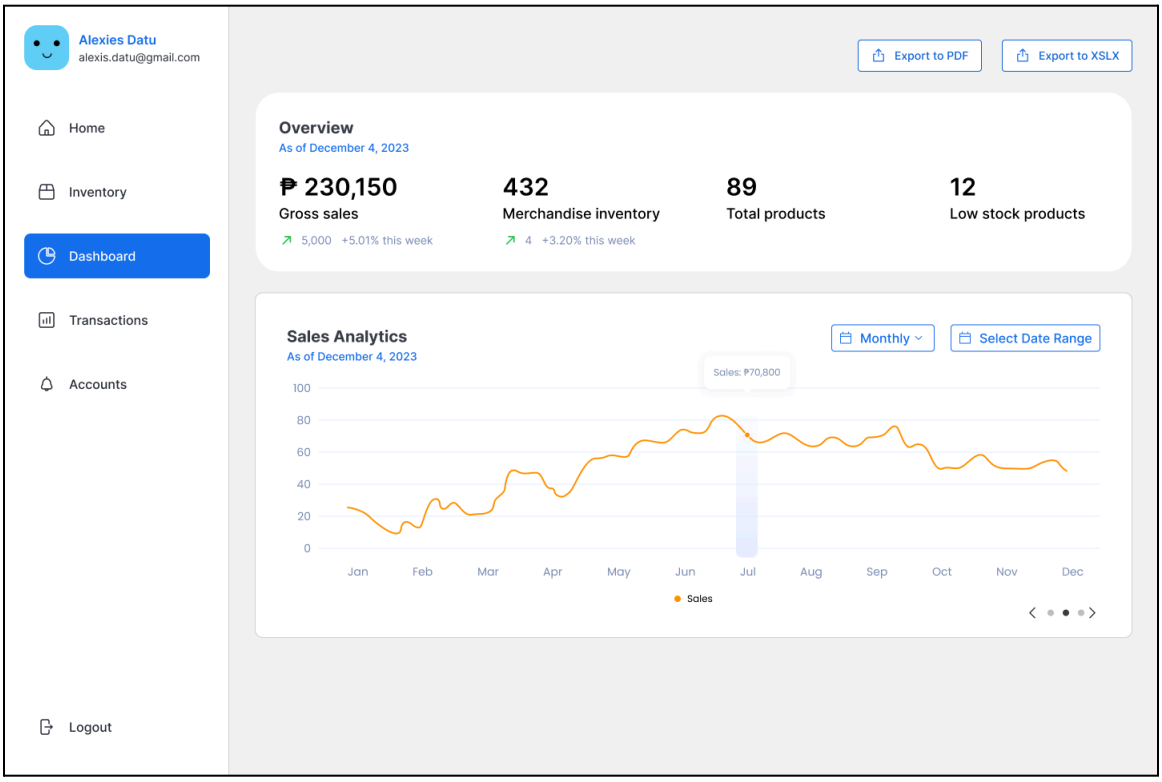

Figure 5.19 Dashboard

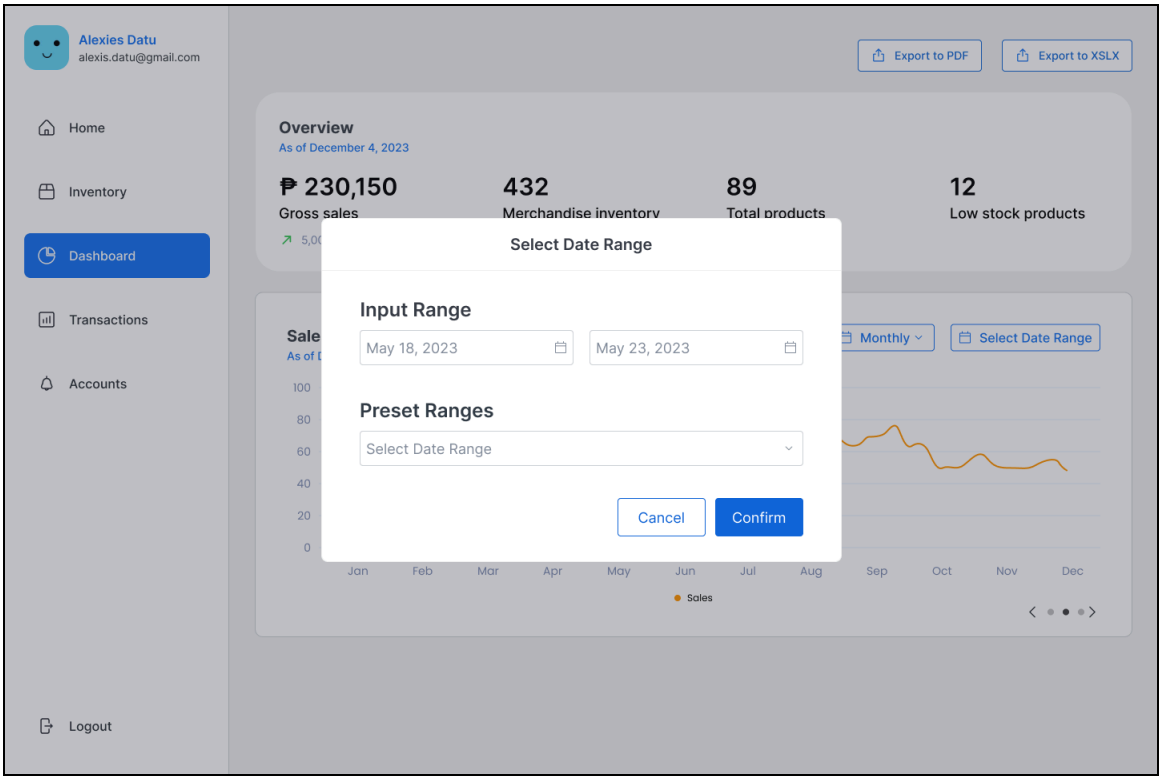

Figure 5.20 Select Date Range Modal

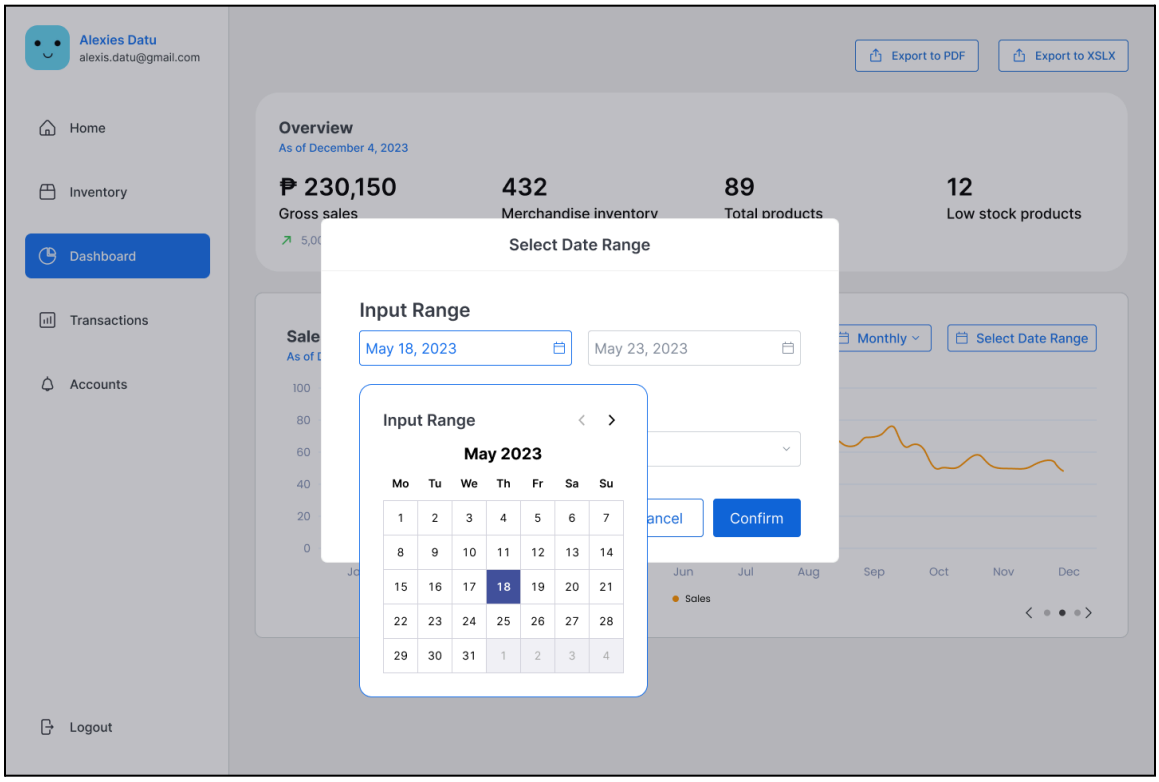

Figure 5.21 Calendar Input (Date Range Modal)

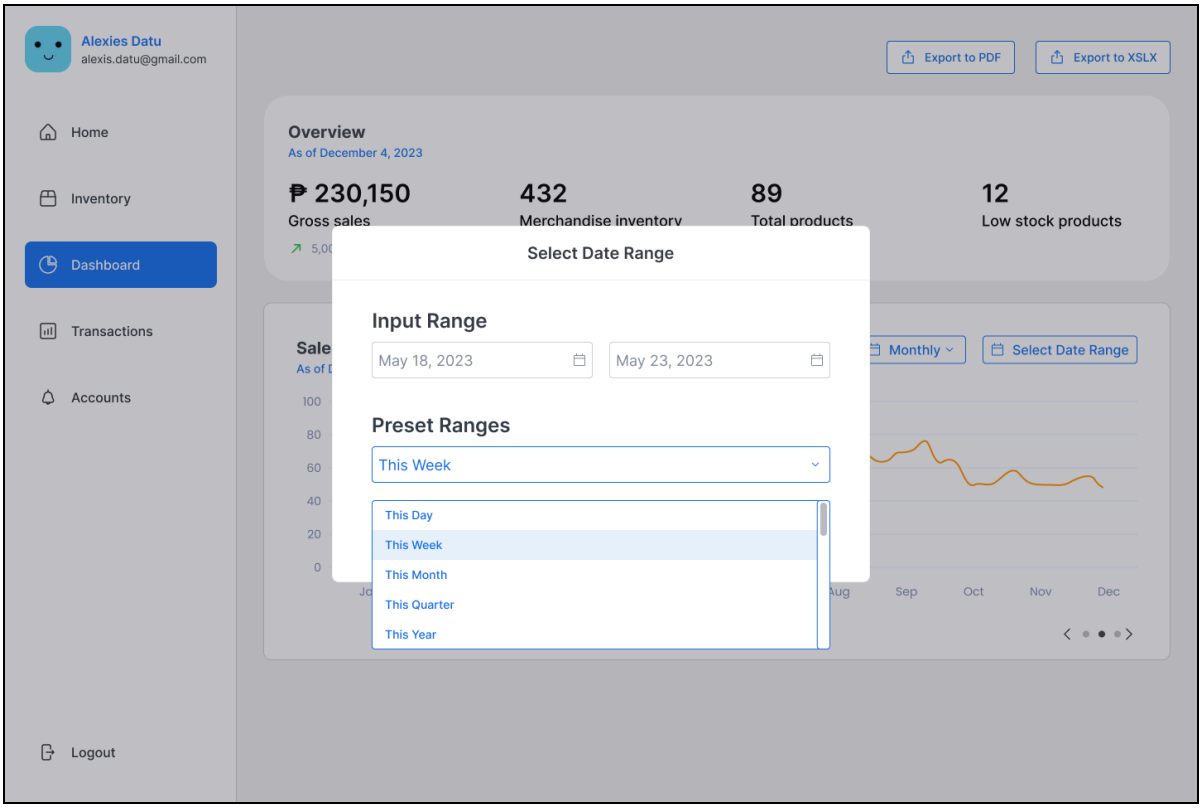

Figure 5.22 Preset Date Ranges (Date Range Modal)

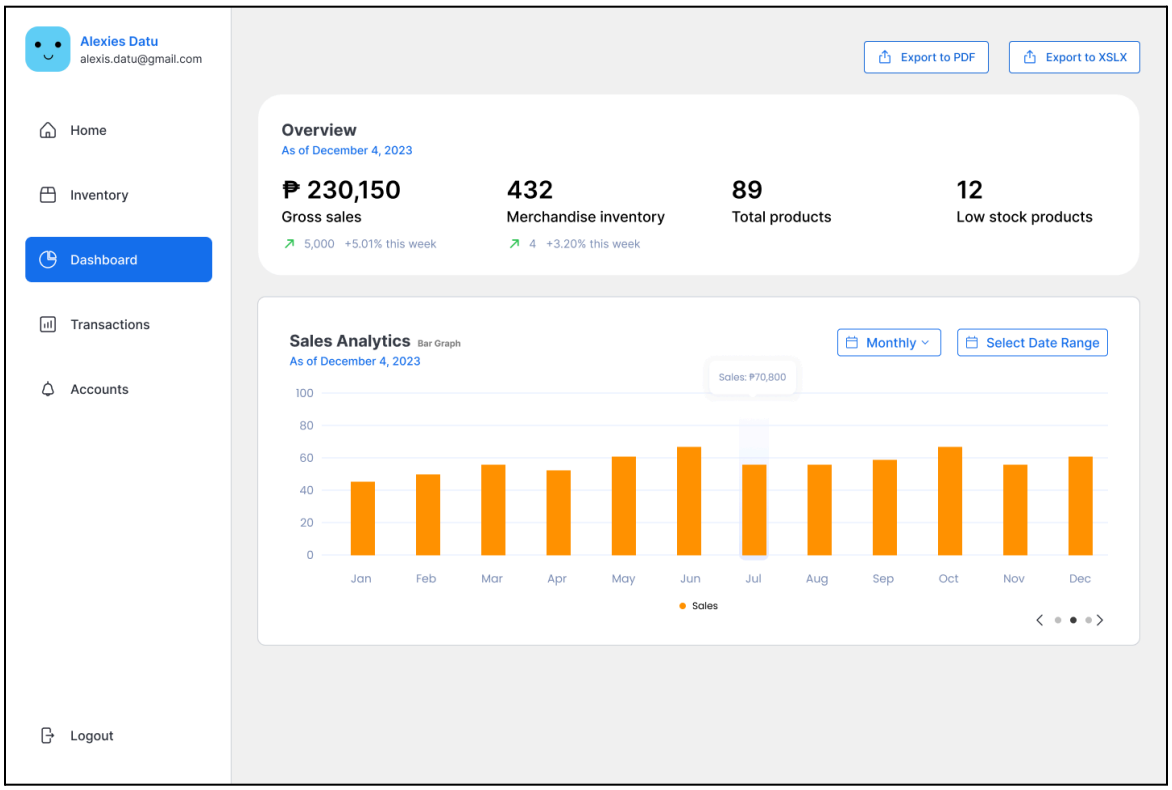

Figure 5.23 Bar Graph Setting for Dashboard

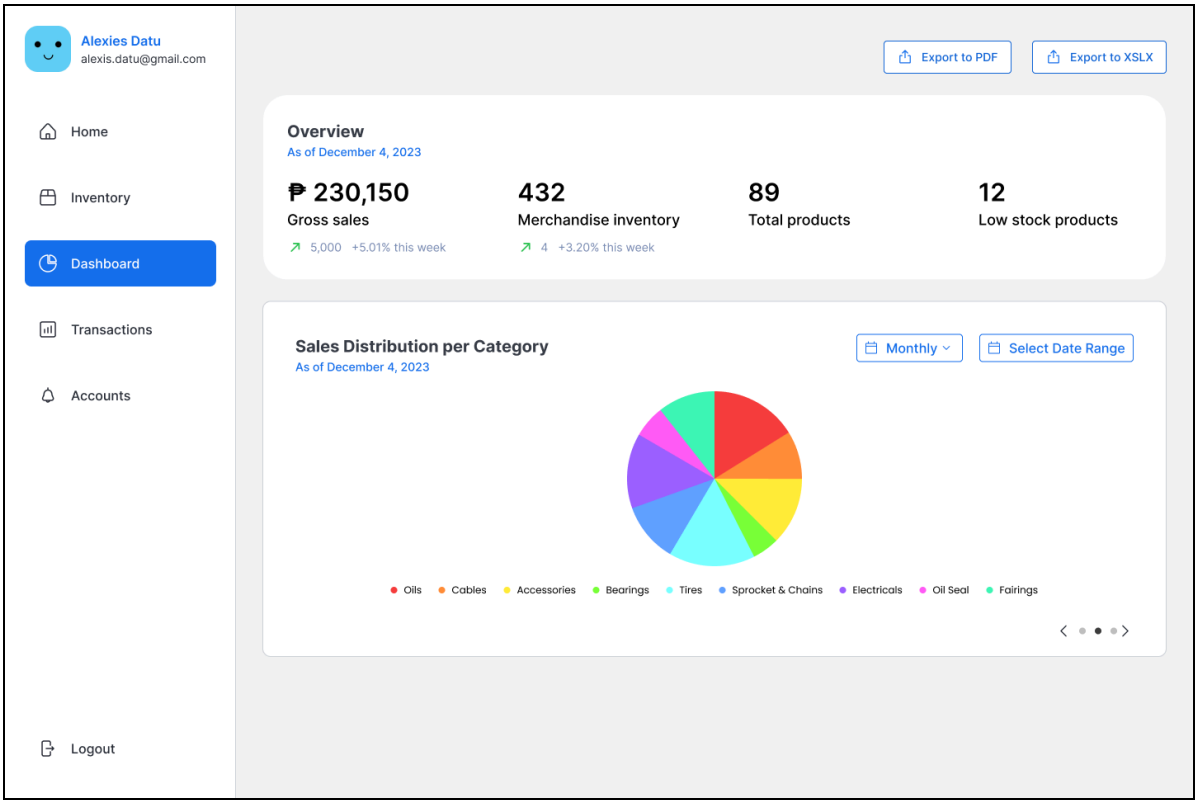

Figure 5.24 Pie Graph Setting for Dashboard

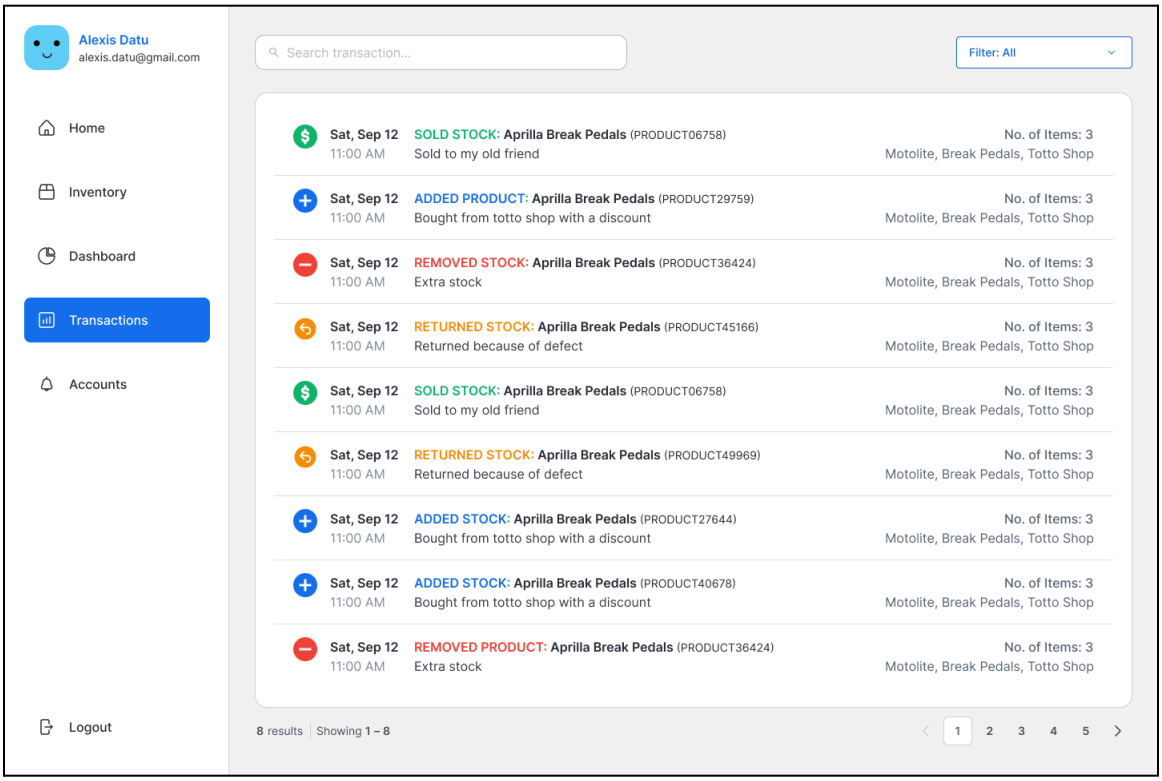

Figure 5.25 Transactions History Screen

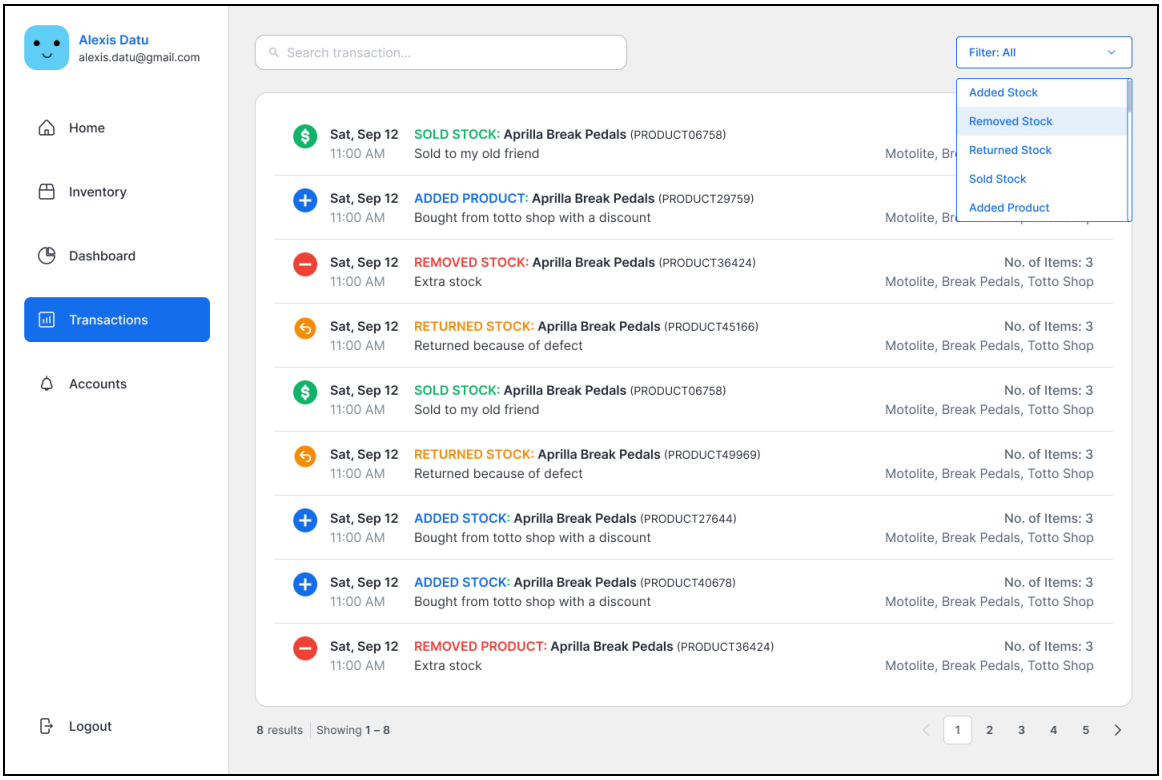

Figure 5.26 Transactions History Screen (Filter)

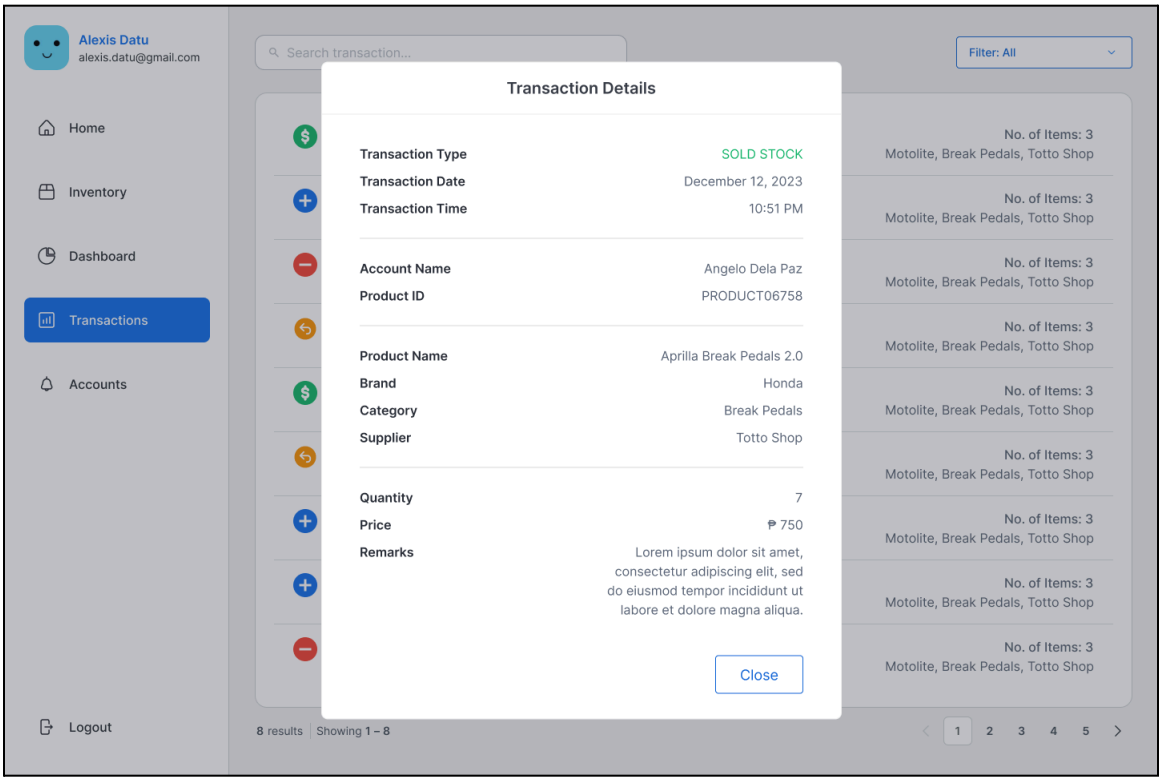

Figure 5.27 Transactions History Screen (View Details)

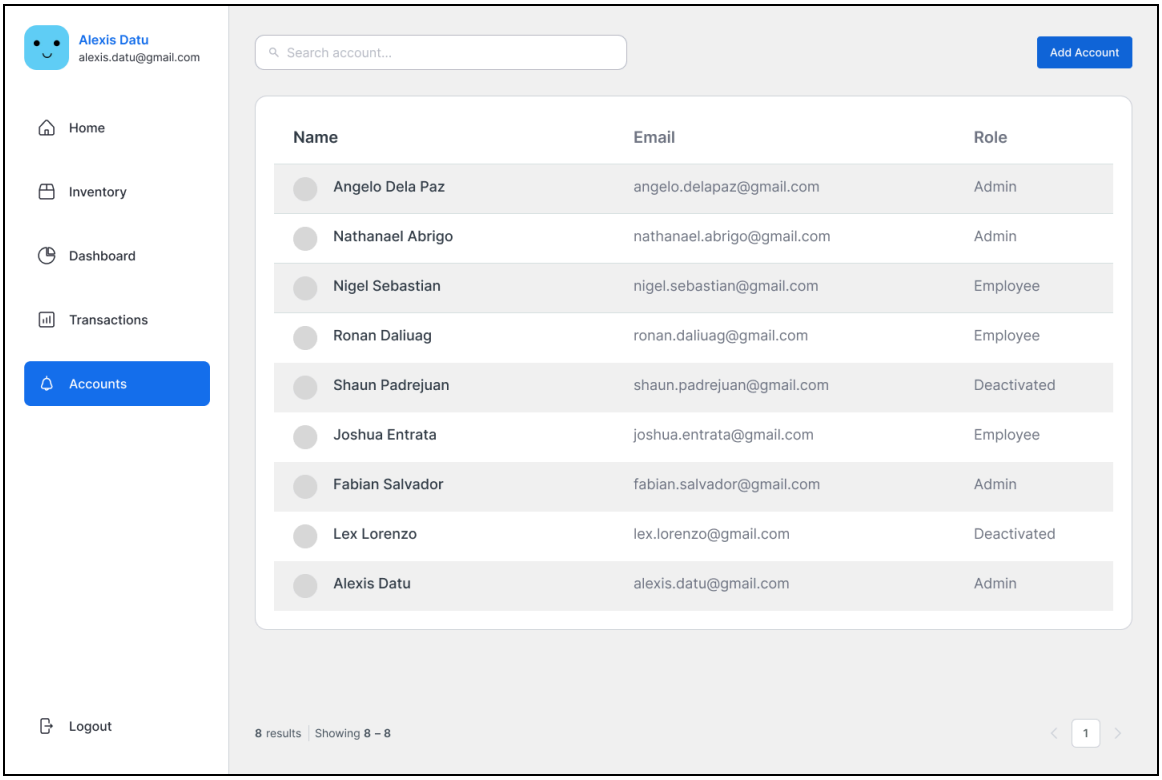

Figure 5.28 Account Management Screen

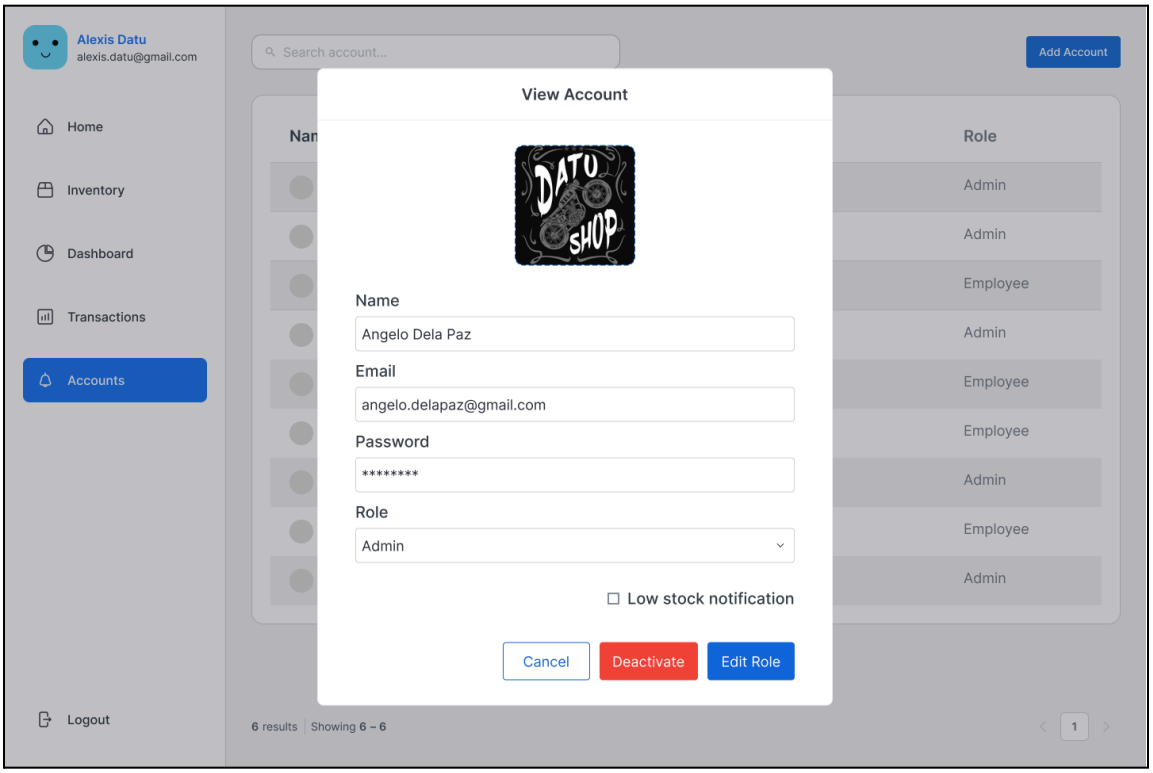

Figure 5.29 Account Management Screen (View Account)

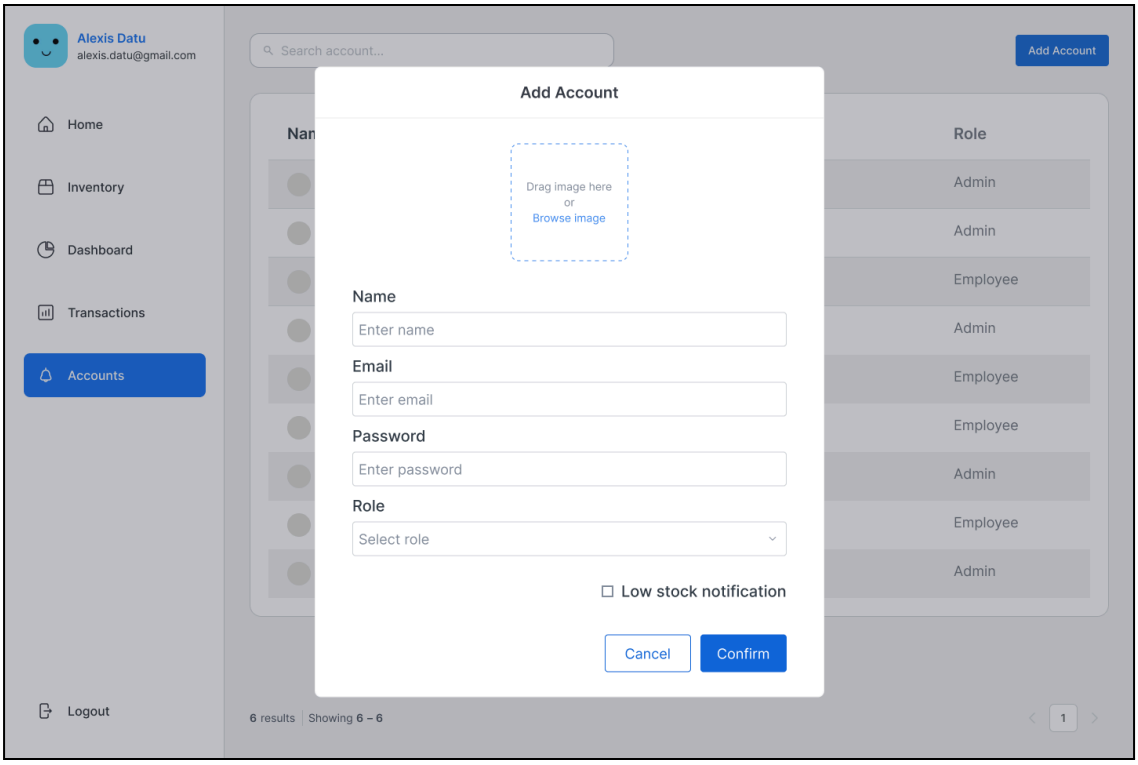

Figure 5.30 Account Management Screen (Add Account)

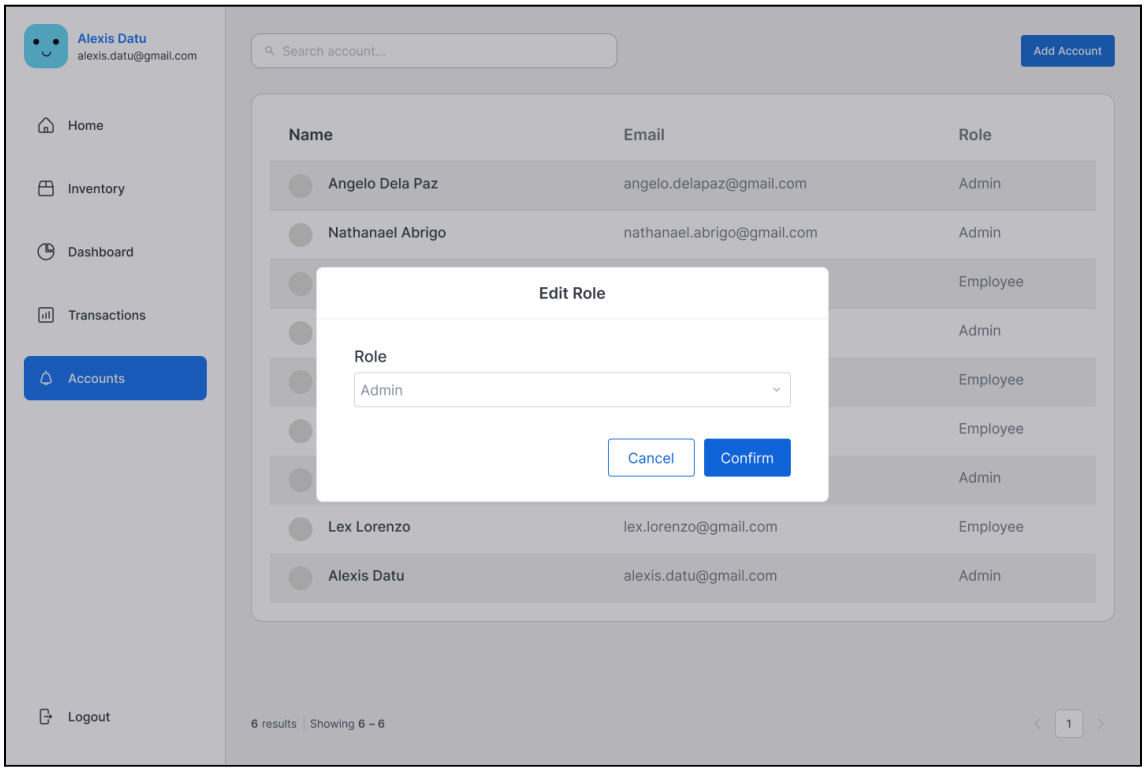

Figure 5.31 Account Management Screen (Edit Role)

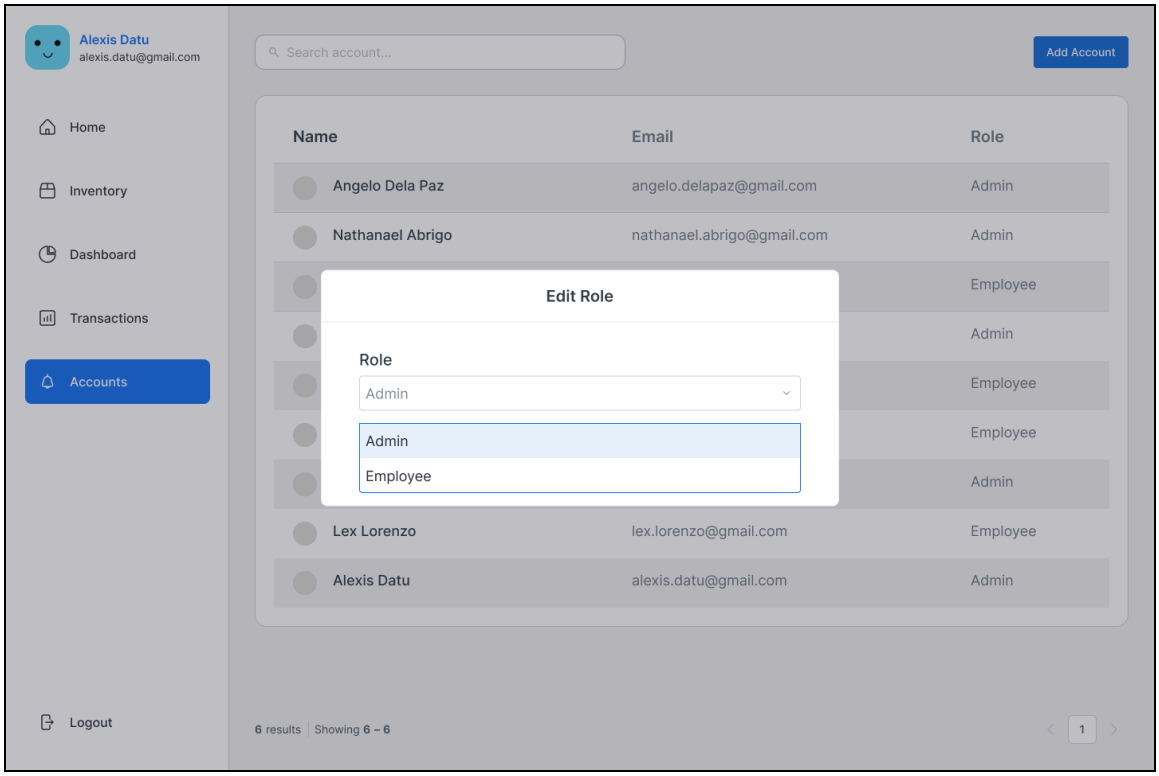

Figure 5.32 Account Management Screen (Roles)

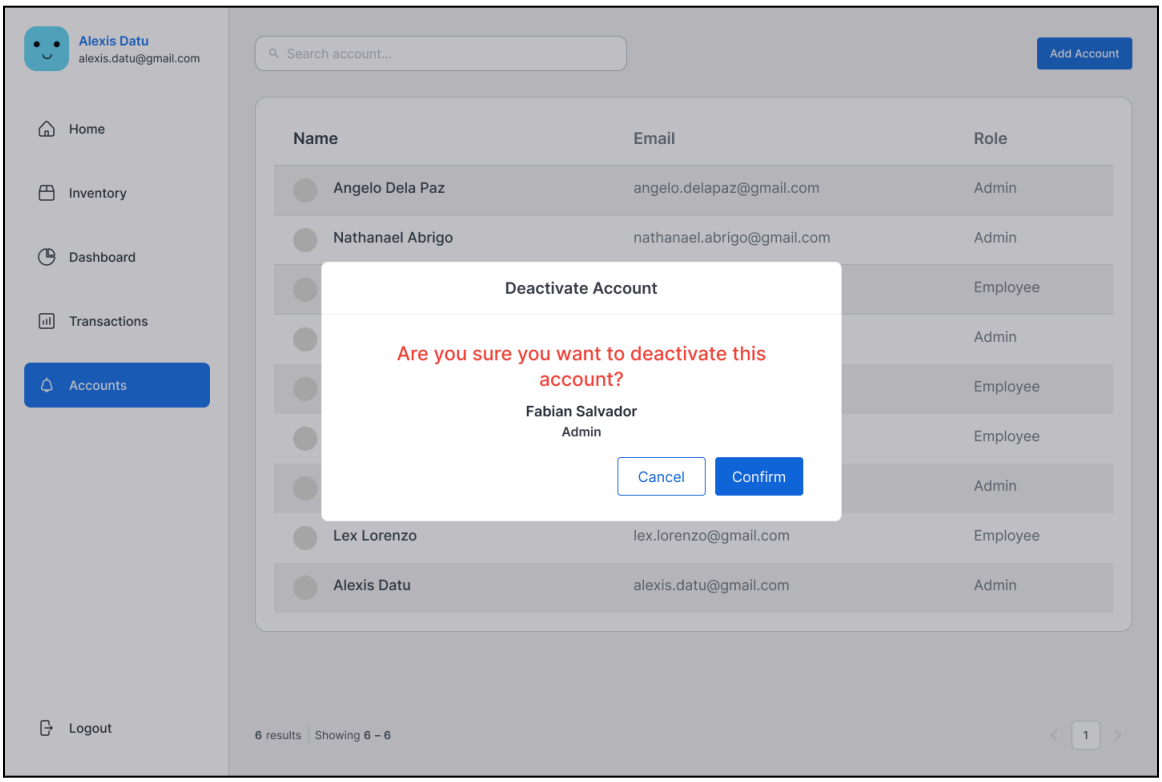

Figure 5.33 Account Management Screen (Deactivate Account)

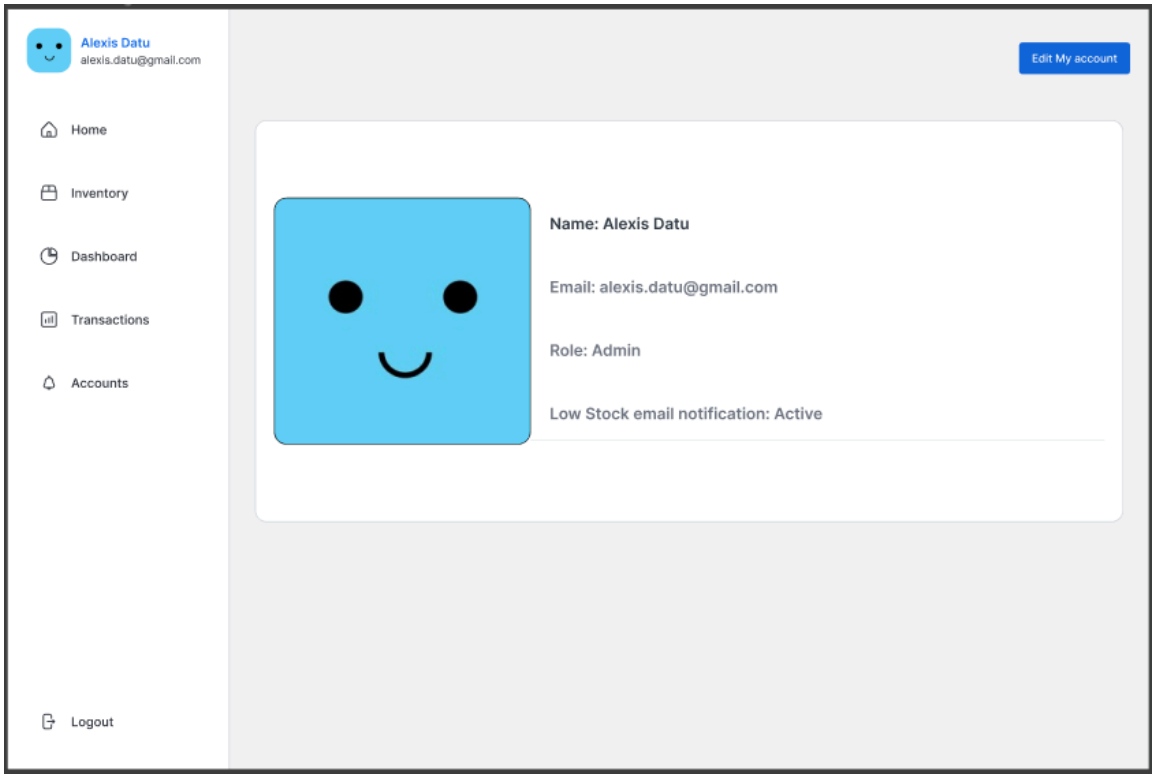

Figure 5.34 Personal Account Screen

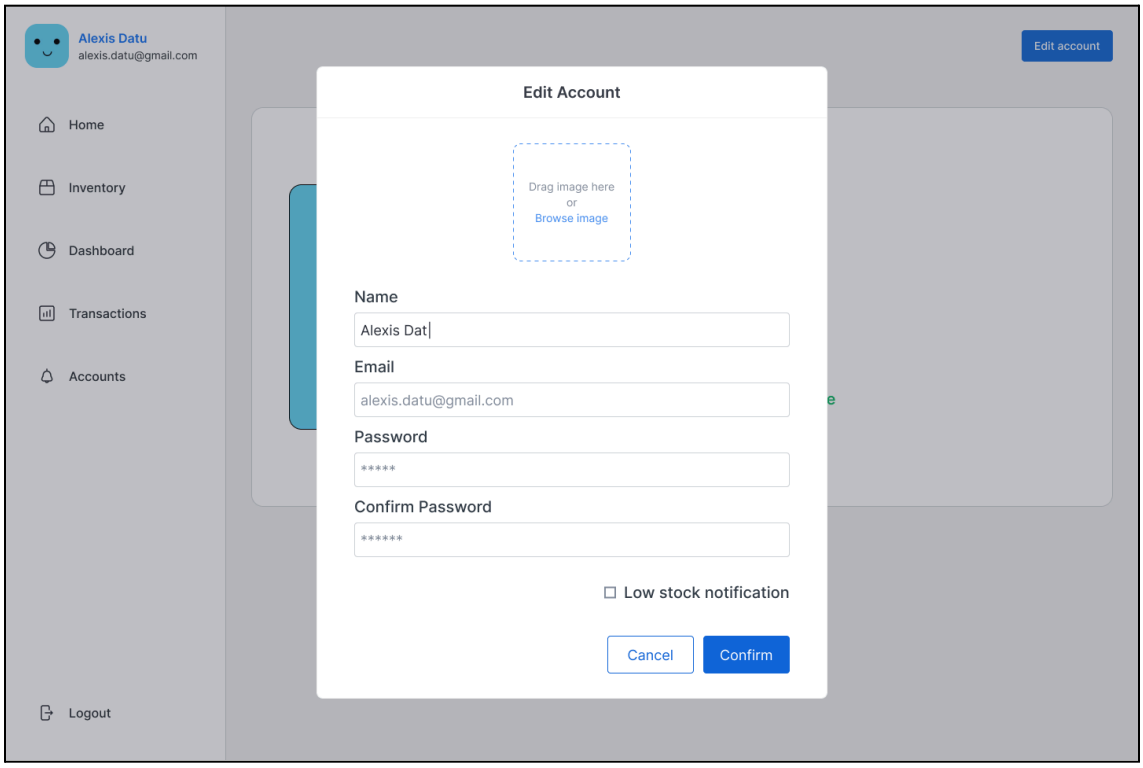

Figure 5.35 Edit Personal Account Screen

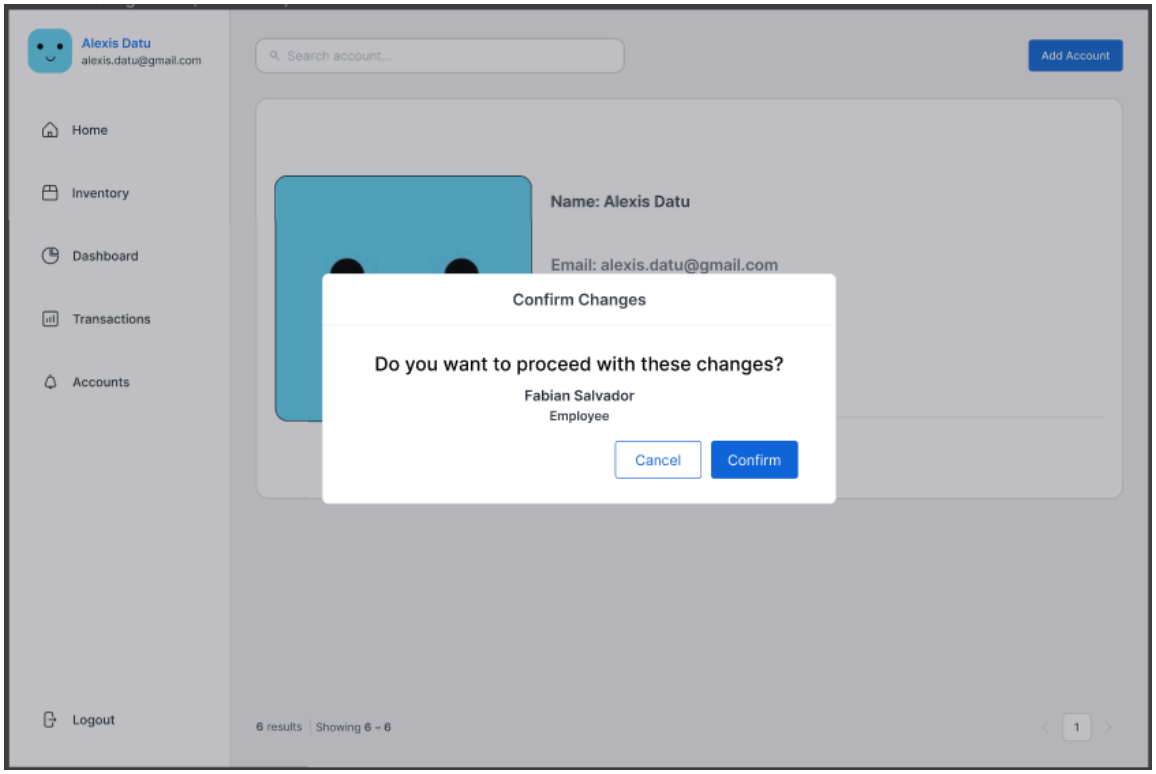

Figure 5.36 Confirm Changes for Personal Account Screen

6. Appendices

6.1. Minutes of the Meeting with Client

Meeting with Client

Date: November 17, 2023

Method: Google Meet Call

Members: Shaun Padrejuan, Nigel Sebastian, Fabian Salvador, Nathanael Abrigo

Guest: Sir Alexies Aries Datu

Agendas

- Demo of Website Mockups.
- Rundown of System Features with the concept of Mockups
- Discussion of Sales Dashboard on sample contents and other possible metrics
- Showcase of Transaction History
- Discussion about first defense
- Clarification about supplier process for products

- Clarification about low stock notifications.
- Clarification about product attributes.

Padrejuan, Shaun Kristoffer C.
Scribe

Alexies Aries Datu
Shop Owner / Interviewee

Meeting with Client

Date: December 4, 2023

Method: Facebook Messenger Call

Members: Shaun Padrejuan

Guest: Sir Alexies Aries Datu

Agendas

- Listing of Categories in the shop
- Listing of limit threshold per category for low stock feature
- Low stock email and dashboard talk about sample use case

Padrejuan, Shaun Kristoffer C.
Scribe

Alexies Aries Datu
Shop Owner / Interviewee