Wholesale Store Management System

(MegaStore)

Database class project, academic year 2024 /2023

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Student Name in English | Student Name in Arabic | Student ID | Section time as shown in zajel | Work percentage |
| Mohammad Munir | محمد منير شديد | 12113028 | 8 - 9:30 | 50% |
| Nassar Mohamad | نصار محمد هرشة | 12144079 | 8 - 9:30 | 50% |
|  |  |  |  |  |

Date/time

---------------------------------------------This section is intended for the Instructor---------------------------------------

|  |  |
| --- | --- |
| **Topic** | **Mark** |
| Project Requirements and Modeling |  |
| Correctness of Database mapping |  |
| Functional Dependency and Normalization |  |
| Project Tools |  |
| Project Discussion |  |
| Project Completeness |  |
| Project Output Results or reporting (JasperReport, charts, graphs, etc.) |  |
| Project Administration and Management |  |
| Project Report |  |
| Project Idea |  |
| Project Complexity |  |
| Team work |  |
|  |  |
|  |  |
|  |  |

**Abstract:**

The Wholesale Store (MegaStore) project represents a sophisticated solution tailored for the effective management of a wholesale store, aiming to enhance operational efficiency and elevate the overall store experience. Developed with the real-world needs of a wholesale business in mind, the application seamlessly combines intuitive user interfaces with robust backend functionalities.

At its core, the project is designed to simplify inventory control, providing a dynamic platform for users to effortlessly monitor and manage product stock. The system empowers store owners with the tools needed to maintain accurate inventory levels, track product performance, and make informed decisions regarding stock replenishment.

Employee administration is another integral feature, allowing for the streamlined management of staff roles and responsibilities. The application ensures a secure and organized working environment, facilitating optimal collaboration among team members. Sales tracking functionalities offer valuable insights into product movement and revenue generation, enabling store owners to adapt strategies and meet evolving customer demands. The project's user-friendly interface, developed using JavaFX, enhances the overall user experience, making day-to-day store operations more intuitive and efficient.

**Table of Content**

|  |  |
| --- | --- |
| **Title** | **Page** |
| Introduction | 4 |
| Project requirement | 5 |
| Functional Dependency | 6 |
| Project UML | 9 |
| Normalization | 10 |
| Tools used in the project | 14 |
| Interfaces | 15 |
| Conclusion | 36 |
| References | 36 |

**Introduction:**

In this project, we have built an application that uses a database to store information about the Wholesale Store (MegaStore), It is like a digital assistant for managing big retail stores. Imagine a smart helper that makes sure products are in the right place, employees know what to do, and the store keeps track of what people are buying. We have used version 42.7.0 of the PostgreSQL JDBC driver to connect to PostgreSQL databases. and provide all the necessary functionalities for the users, such as search, delete, insert, update, and view tables information from database. In the PgAdmin 4, we created tables to represent key entities like employees, products, warehouses, and transactions… , Each table was carefully designed to store relevant information, ensuring data integrity and efficient retrieval.

Developed using JavaFX, the project not only boasts robust backend functionalities but also prioritizes user experience. The user-friendly interface ensures that day-to-day store operations are not only efficiently managed but also intuitive and accessible.

Our focus is on making sure the store runs smoothly. It helps keep track of all the products, manages the employees, and even shows how well the store is doing in terms of sales. Think of it as a virtual manager that handles the nitty-gritty details, leaving the real managers with more time to make smart decisions.

Our goal is not just about showing off fancy technology but about solving real problems faced by big stores today. As we walk you through our Wholesale Store (MegaStore) project, we'll share how it simplifies the complexities of running a successful retail business.

Overall, this application provides a comprehensive and user-friendly solution for managing warehouses and its inventory in the Whole sale stores.

**Wholesale Store Database Requirements**

* The wholesale store organized into Warehouses, each wherehouse has a unique WarehouseID, a name, and an employee work as manager, a location which expressed by the name of the street and city, and Inventory Capacity which it’s the area of the warehouse place in m2.
* For each employee we will store its name(first name and last name), unique EmployeeID, address(city,street), salary in local currency, E-mail(It is supposed to be unique), and password So that he can enter the system through the login interface, and role of this employee Which could be an **Admin** who can access all parts of the system, make all forms of modifications, and review all data, Or he may be a **Manager**, as he is allowed to see the data of the warehouse attached to him only, as he is his management, where he sees the data of the employees in the warehouse and edits it, and sees the inventory records of his warehouse and can modify them, and he can also modify the data of his own warehouse. Or he can be an **Accountant**, as it allows him to see and modify data related to purchase and sale orders, as well as financial transaction, Or he is a **Worker** as he is not allowed to enter the system. We also will store phone number, Data of birth, Gender, Hire Date(The date when the employee was started work in wholesale), and WarehouseID where employee works, Bank Account Information.
* In the store there is products, for each one we will track ProductID, Product Name, price, Manufacturer(The country that manufactured the product and it`s may be one of these (local, Egypt, Jordan, Turky, Saudi arabia)), the brand of the product, and the shelf life (The period of time during which the product remains fit for consumption).
* There is inventory system in which the quantity of each product in each store and the production dates for each inventory are stored, The changeing on the quantities of the products in inventory It is done through buying and selling operations. in selling case, we store order information which are OrderID , CustomerID, Order Date, quantities of each product in order and from which Warehouse and production date, and total price, OrderNotes and the customer. In buying case we store Purchase order information which are PurchaseOrderID, SupplierID, Order Date, quantities of each product in order and for which Warehouse and production date, and total price and the supplier.
* And also we need to record for financial transactions ,transactionID,transaction Type, Amount, transactiondate, description.

• For the Customers, we will store CustomerID, Name , Contact Information , and address.

• For the Suppliers, we will store SupplierID , Company Name , Contact Information , and Location ( location where the supplier operates ) .

**Functional Dependencies:**

**Employee:**

* **Employee ID, email\_address → first\_name**
* **Employee ID, email\_address → last\_name**
* **Employee ID, email\_address → city**
* **Employee ID, email\_address → street**
* **Employee ID, email\_address → email\_address**
* **Employee ID, email\_address → phone\_number**
* **Employee ID, email\_address → gender**
* **Employee ID, email\_address → salary**
* **Employee ID, email\_address → hire\_date**
* **Employee ID, email\_address → bank\_account\_information**
* **Employee ID, email\_address → birthdate**
* **Employee ID, email\_address → warehouseid**
* **Employee ID, email\_address → role**
* **Employee ID, email\_address → password**

**Customers:**

* **Customer ID → customer name**
* **Customer ID → contact\_information**
* **Customer ID → address**

**Supplier:**

* **Supplier ID → company\_name**
* **Supplier ID → contact\_information**
* **Supplier ID → supplier\_location**

**Warehouse:**

* **Warehouse ID → inventory capacity**
* **Warehouse ID → Warehosue Name**
* **Warehouse ID → city**
* **Warehouse ID → street**

**Product:**

* **Product ID → price**
* **Product ID → manufacturer**
* **Product ID → product name**
* **Product ID → shelf life**
* **Product ID → brand**

**Order:**

* **Order ID→customer id**
* **Order ID →order date**
* **Order ID →order notes**
* **Order ID →total**

**Purchase Order:**

* **Purchaseorder ID →supplierid**
* **Purchaseorder ID →totalprice**
* **Purchaseorder** **ID →order\_date**

**Order Details:**

* **orderdetailid→orderid**
* **orderdetailid→productid**
* **orderdetailid→quantity**
* **orderdetailid→productiondate**
* **orderdetailid→warehouseid**

**P Order Details:**

* **purchaseorderdetail ID→purchaseorder ID**
* **purchaseorderdetail ID→warehouse ID**
* **purchaseorderdetail ID→productiondate**
* **purchaseorderdetail ID→product ID**
* **purchaseorderdetail ID→quantity**

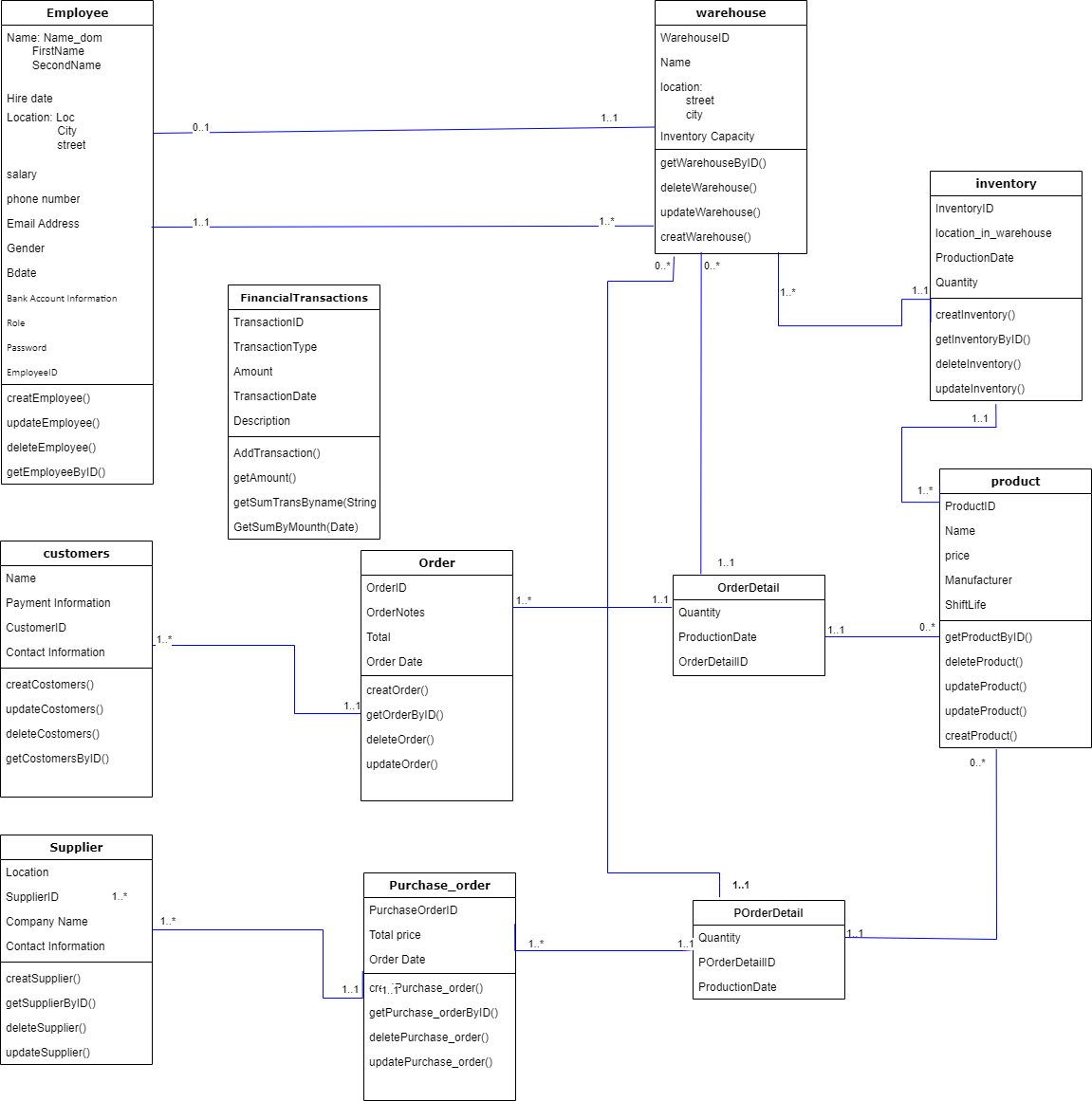
**Financial Transaction:**

* **Transaction ID→ transactiontype**
* **Transaction ID → amount**
* **Transaction ID →** **transaction date**
* **Transaction ID → description**

**Inventory:**

* **inventory ID→product ID**
* **inventory ID→quantity**
* **inventory ID→productiondate**
* **inventory ID→warehouse ID**
* **inventory ID→location\_in\_warehouse**

**UML Diagram:**



**Normalization:**

**Employee:**

* **first normal form (1NF):** There is no multivalued attribute in this table, so **Employee** table in 1NF.
* **second normal form (2NF)**: There is no composite key in this table, so **Employee** table is in 2NF.
* **Third normal form (3NF)**:No non-primary-key column is transitively dependent on the primary key, so **Employee** table in 3NF.
* **Boyce-Codd normal form (BCNF)**:In this table every determinant is a candidate key, so **Employee** table in BCNF.

**Customers:**

* **first normal form (1NF):** There is no multivalued attribute in this table, so **Customers** table in 1NF.
* **second normal form (2NF)**: There is no composite key in this table, so **Customers** table is in 2NF.
* **Third normal form (3NF)**:No non-primary-key column is transitively dependent on the primary key, so **Customers** table in 3NF.
* **Boyce-Codd normal form (BCNF)**:In this table every determinant is a candidate key, so **Customers** table in BCNF.

**Supplier:**

* **first normal form (1NF):** There is no multivalued attribute in this table, so **Supplier** table in 1NF.
* **second normal form (2NF)**: There is no composite key in this table, so **Supplier** table is in 2NF.
* **Third normal form (3NF)**:No non-primary-key column is transitively dependent on the primary key, so **Supplier** table in 3NF.
* **Boyce-Codd normal form (BCNF)**:In this table every determinant is a candidate key, so **Supplier** table in BCNF.

**Warehouse:**

* **first normal form (1NF):** There is no multivalued attribute in this table, so **Warehouse** table in 1NF.
* **second normal form (2NF)**: There is no composite key in this table, so **Warehouse** table is in 2NF.
* **Third normal form (3NF)**:No non-primary-key column is transitively dependent on the primary key, so **Warehouse** table in 3NF.
* **Boyce-Codd normal form (BCNF)**:In this table every determinant is a candidate key, so **Warehouse** table in BCNF.

**Product:**

* **first normal form (1NF):** There is no multivalued attribute in this table, so **Product** table in 1NF.
* **second normal form (2NF)**: There is no composite key in this table, so **Product** table is in 2NF.
* **Third normal form (3NF)**:No non-primary-key column is transitively dependent on the primary key, so **Product** table in 3NF.
* **Boyce-Codd normal form (BCNF)**:In this table every determinant is a candidate key, so **Product** table in BCNF.

**Order:**

* **first normal form (1NF):** There is no multivalued attribute in this table, so **Order** table in 1NF.
* **second normal form (2NF)**: There is no composite key in this table, so **Order** table is in 2NF.
* **Third normal form (3NF)**:No non-primary-key column is transitively dependent on the primary key, so **Order** table in 3NF.
* **Boyce-Codd normal form (BCNF)**:In this table every determinant is a candidate key, so **Order** table in BCNF.

**Order Details:**

* **first normal form (1NF):** There is no multivalued attribute in this table, so **Order Details** table in 1NF.
* **second normal form (2NF)**: There is no composite key in this table, so **Order Details** table is in 2NF.
* **Third normal form (3NF)**:No non-primary-key column is transitively dependent on the primary key, so **Order Details** table in 3NF.
* **Boyce-Codd normal form (BCNF)**:In this table every determinant is a candidate key, so **Order Details** table in BCNF.

**Purchase Order:**

* **first normal form (1NF):** There is no multivalued attribute in this table, so **Purchase Order** table in 1NF.
* **second normal form (2NF)**: There is no composite key in this table, so **Purchase Order** table is in 2NF.
* **Third normal form (3NF)**:No non-primary-key column is transitively dependent on the primary key, so **Purchase Order** table in 3NF.
* **Boyce-Codd normal form (BCNF)**:In this table every determinant is a candidate key, so **Purchase Order** table in BCNF.

**P Order Details:**

* **first normal form (1NF):** There is no multivalued attribute in this table, so **P Order Details** table in 1NF.
* **second normal form (2NF)**: There is no composite key in this table, so **P Order Details** table is in 2NF.
* **Third normal form (3NF)**:No non-primary-key column is transitively dependent on the primary key, so **P Order Details** table in 3NF.
* **Boyce-Codd normal form (BCNF)**:In this table every determinant is a candidate key, so **P Order Details** table in BCNF.

**Financial Transaction:**

* **first normal form (1NF):** There is no multivalued attribute in this table, so **Financial Transaction** table in 1NF.
* **second normal form (2NF)**: There is no composite key in this table, so **Financial Transaction** table is in 2NF.
* **Third normal form (3NF)**:No non-primary-key column is transitively dependent on the primary key, so **Financial Transaction** table in 3NF.
* **Boyce-Codd normal form (BCNF)**:In this table every determinant is a candidate key, so **Financial Transaction** table in BCNF.

**Inventory:**

* **first normal form (1NF):** There is no multivalued attribute in this table, so **Inventory** table in 1NF.
* **second normal form (2NF)**: There is no composite key in this table, so **Inventory** table is in 2NF.
* **Third normal form (3NF)**:No non-primary-key column is transitively dependent on the primary key, so **Inventory** table in 3NF.
* **Boyce-Codd normal form (BCNF)**:In this table every determinant is a candidate key, so **Inventory** table in BCNF.

**Tools and software used in the project:**

* IntelliJ ultimate.
* JavaFX & Scene Builder.
* PgAdmin 4 SQL Developer.
* PostgreSQl JDBC 42.7.0.
* Jaspersoft Studio.
* Just color picker.
* GitHub.
* Canva.
* Icon8
* Youtube
* Adobe Photoshop 2022
* Draw.io
* Color Fitter
* PostgreSQL 16.1 Documentation
* PostgreSQL Tutorial
* discord

**Interfaces:**

**We have 3 types of users who can access the system:**

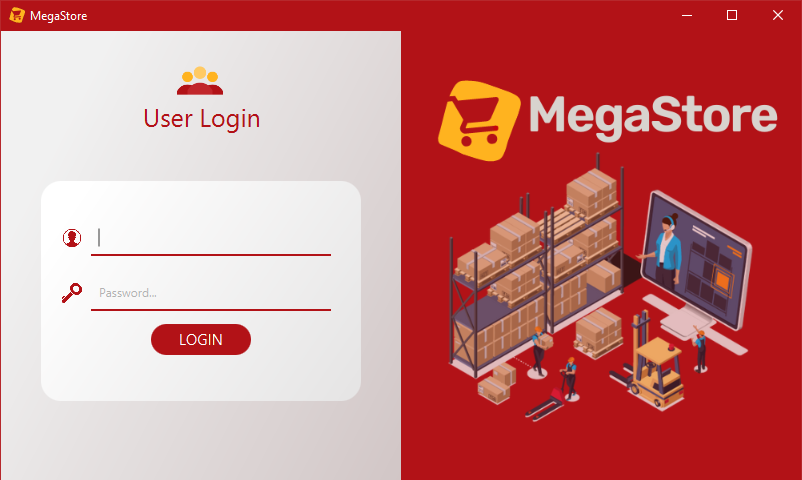
* **Admin**
* **Manager (has privileges over a one Warehouse)**
* **Accountant**

First, there is a splash interface, after which a login interface appears, and from the entered registration data, it is determined who can enter and their permissions are determined. **Admin** user has full acsess and modify to all interfaces except (Financial Transaction interface, The accountant is distinguished by his access to it only), **Manager** user can Access and modify to Employee and inventory interfaces(Data that related to the warehouse that he manages) and to Warehouse settings interface, **Accountanat** user can access and modify in Order and Purchase Order interfaces and Order detail interface and Financial Transaction interfaceThose interfaces being either a main interface accessed from the navigation panel or pop-ups accessed from other interfaces.

*\*We don't have a sign-up window because it is a closed system. Admins and managers can add users and give them permissions*.

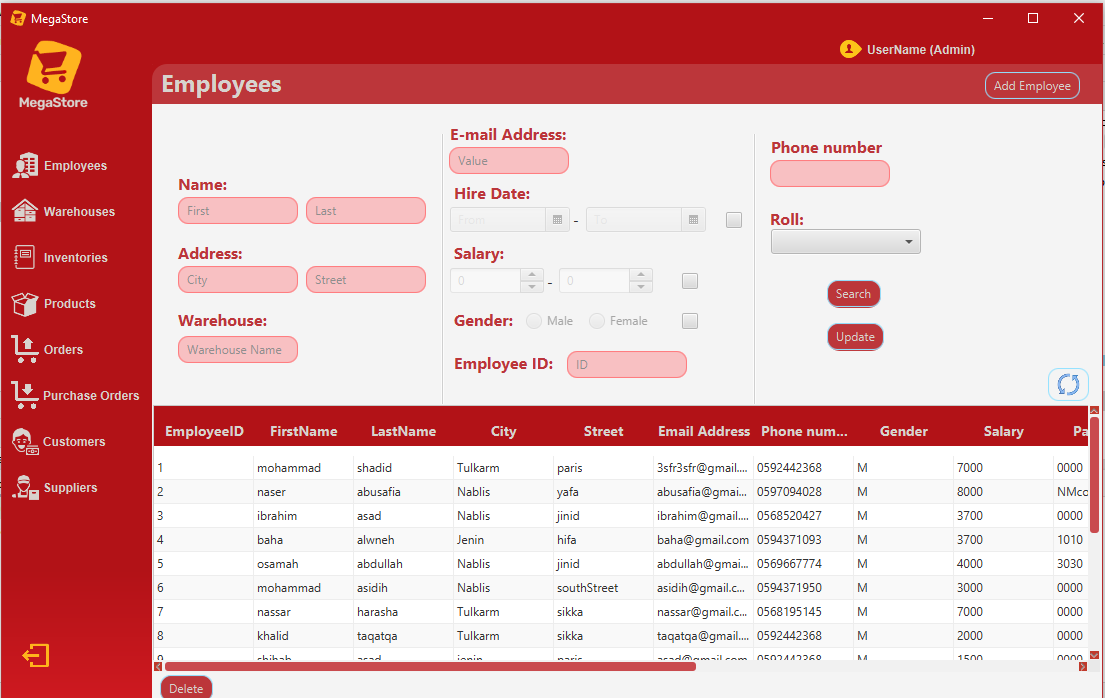
**SplashScreen**:

**Login Screen**



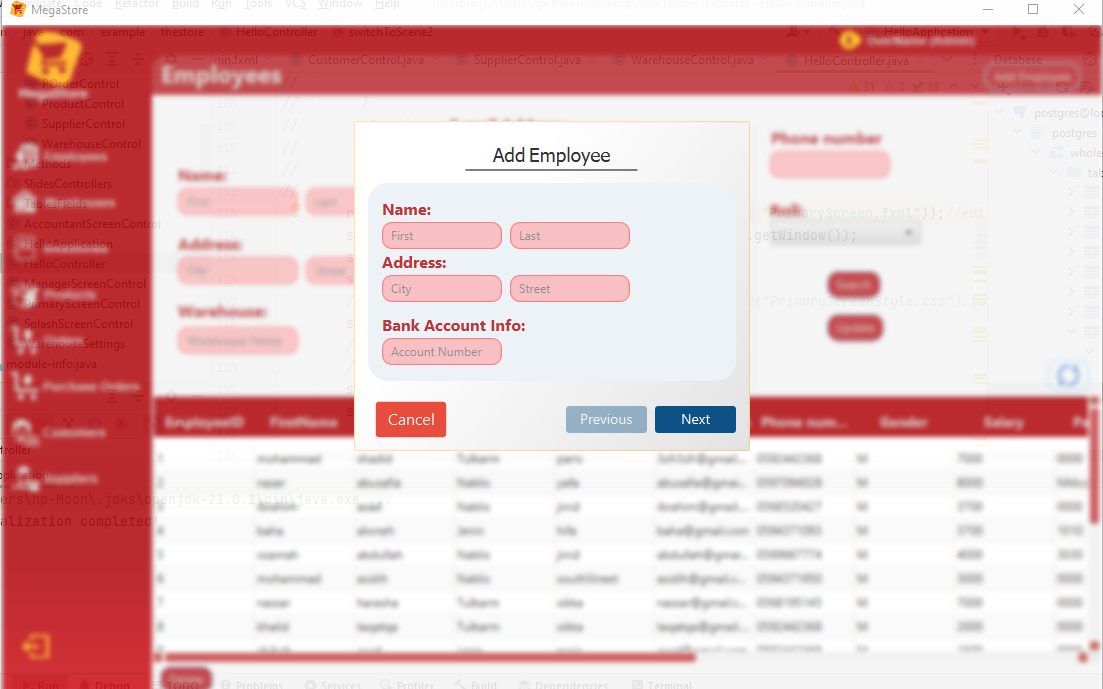
In this interface, the user enter his email and password and click on the button **LOGIN**, if email and password exist for a record in Employee table then there is four cases, first if the user is **Admin** then the Admin view will show(see it below), if the user is **Manager** the Manager View will show for containing information of its warehouse, if the user is **Accountant** accountant view will show, and if the user is **worker** no access for him and there is no view will apear.

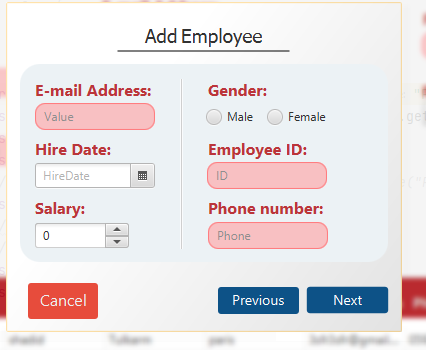
If email does not exist or password wrong, then Warning red text will appear.

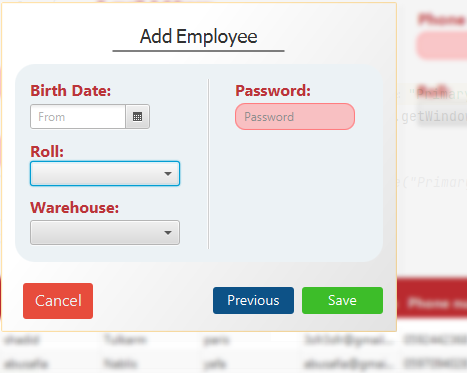
**Admin Interfaces and Adding pop ups:-**

**Employee Interface:**

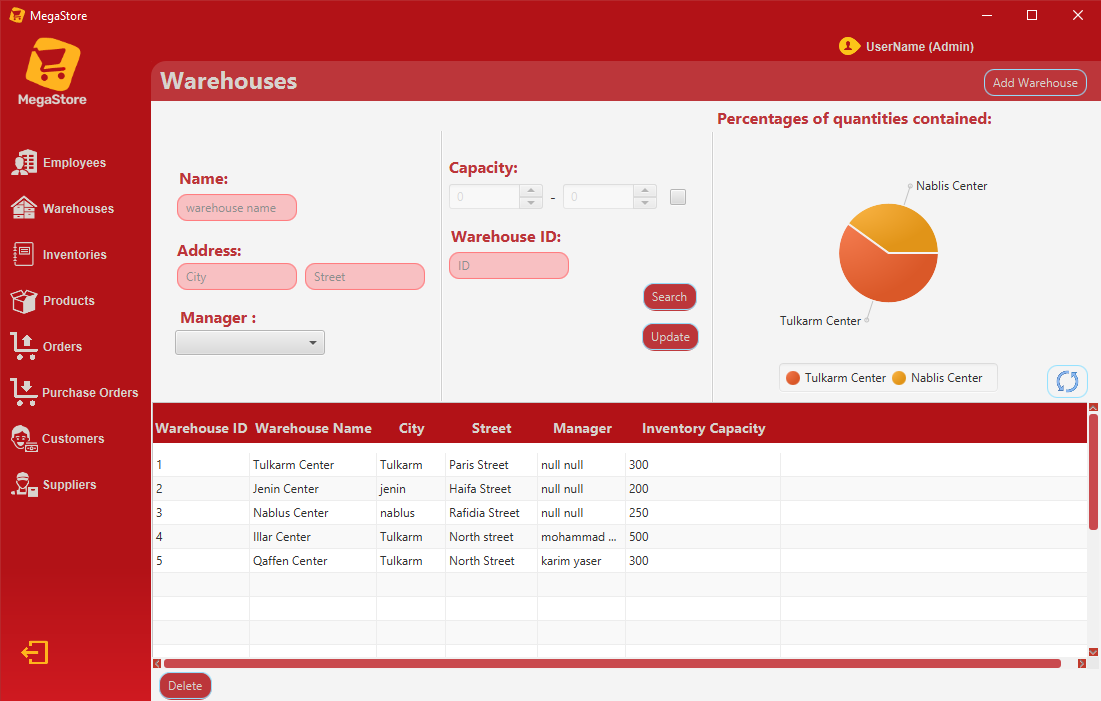
This interface contain Table view will show the records for all employees(In Admin view), and contain search part, when user enter value for one field or more and click on search button then result search will show on table , and if user click on refreash button the table will return to contain all records,and when selecting row and click delete then this row will delete , and if click update then the adding popup(see it below) will appear and contain the current values of the row with the ability to modify it and then save. When click Add Employee then adding popup will apear.

**Adding Employee Popup:**  
 Enter data of new employee and click next until save.



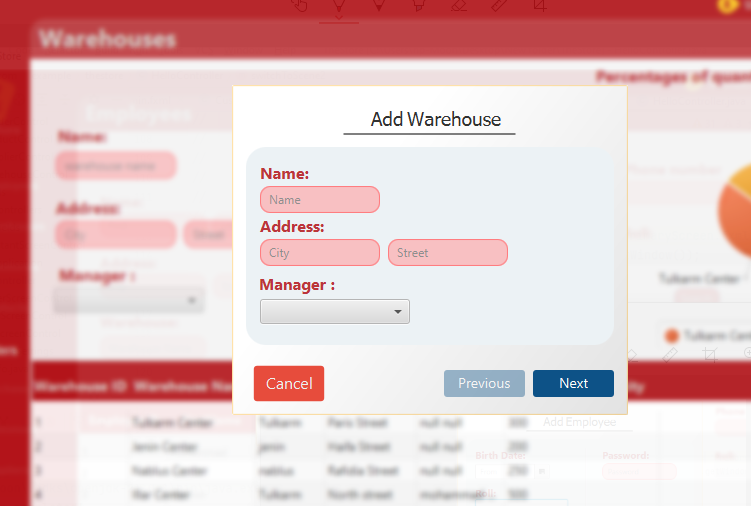
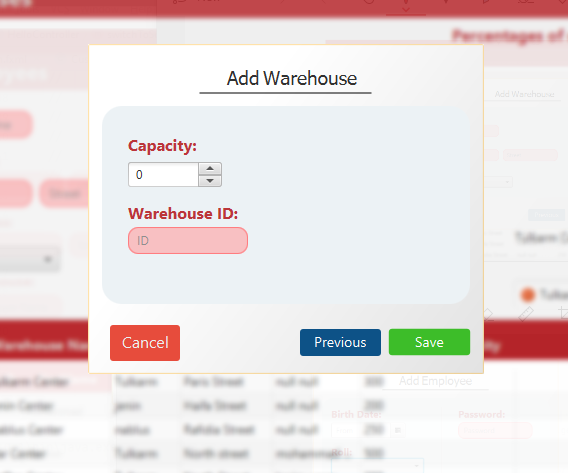
****

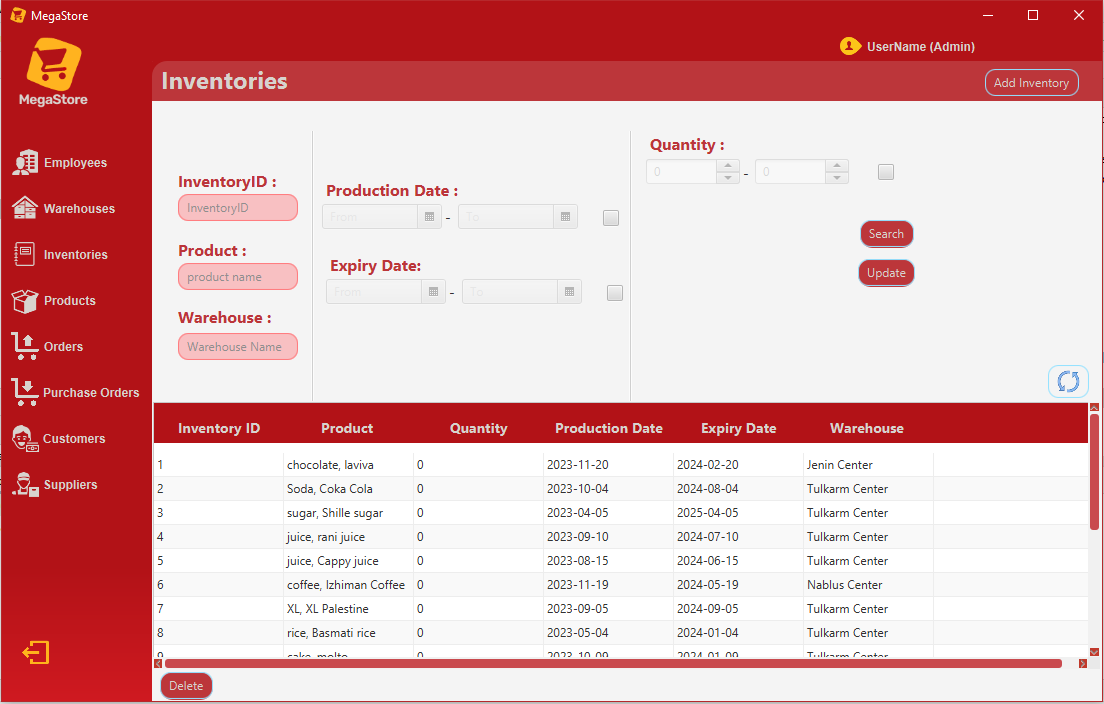
After adding data for final and previous slides, click save and then the employee wil add.

**Warehouse Interface :**

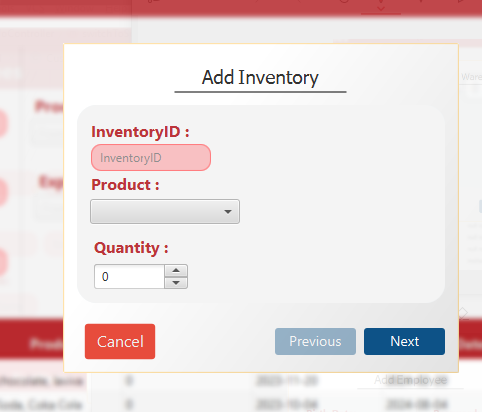
Like employee, warehouse interface contain table and (Add,Search,update,delete) operations, and contain PieChart representation showing the quantities contained in the Warehouses.

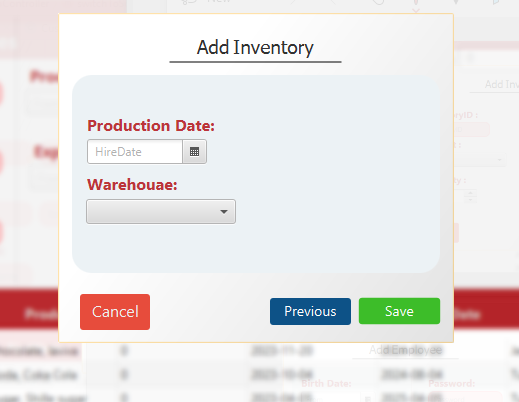
**Adding Warehouse popup:-**

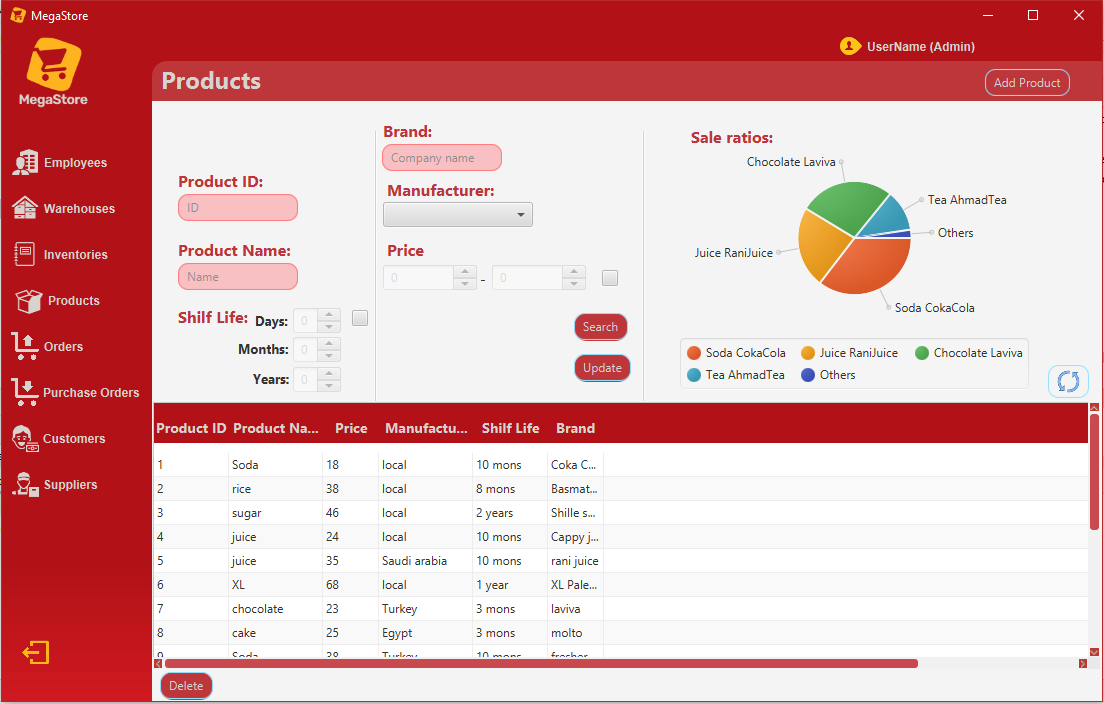
 

**Inventory Interface:**

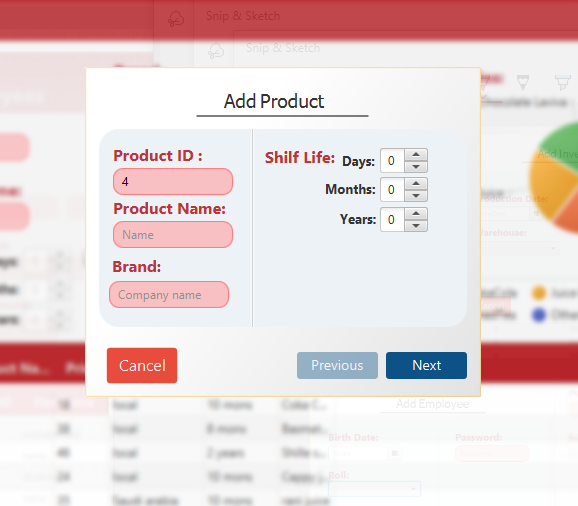
Contain the same previous operations for editing and showing data of inventory ..

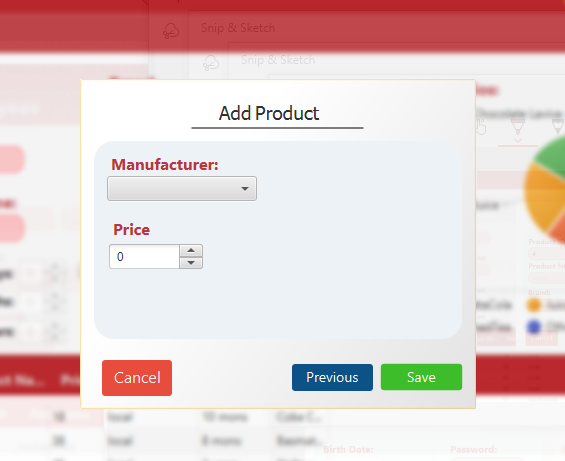
 Add Inventory Popup:

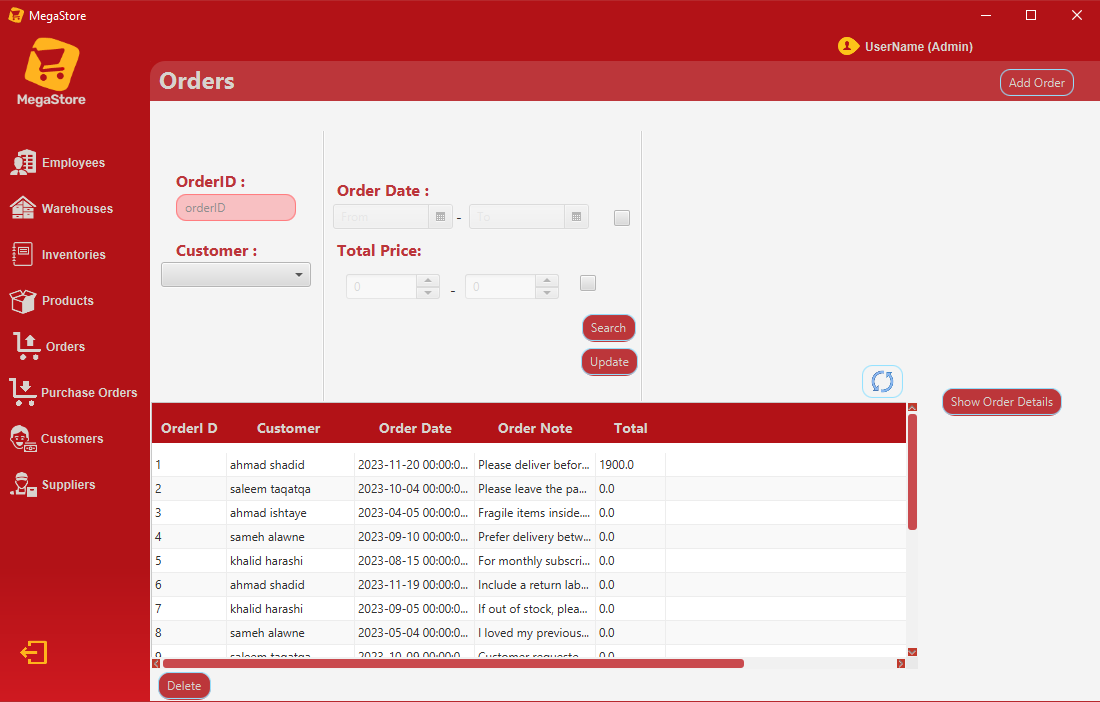


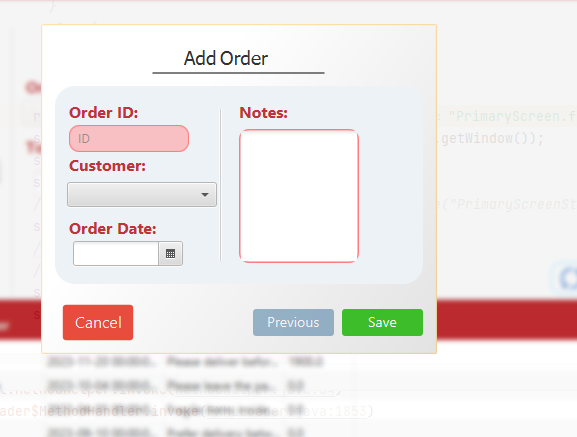
**Product Interface and its adding popup:**

Product interface contain PieChart Represents sales of products.

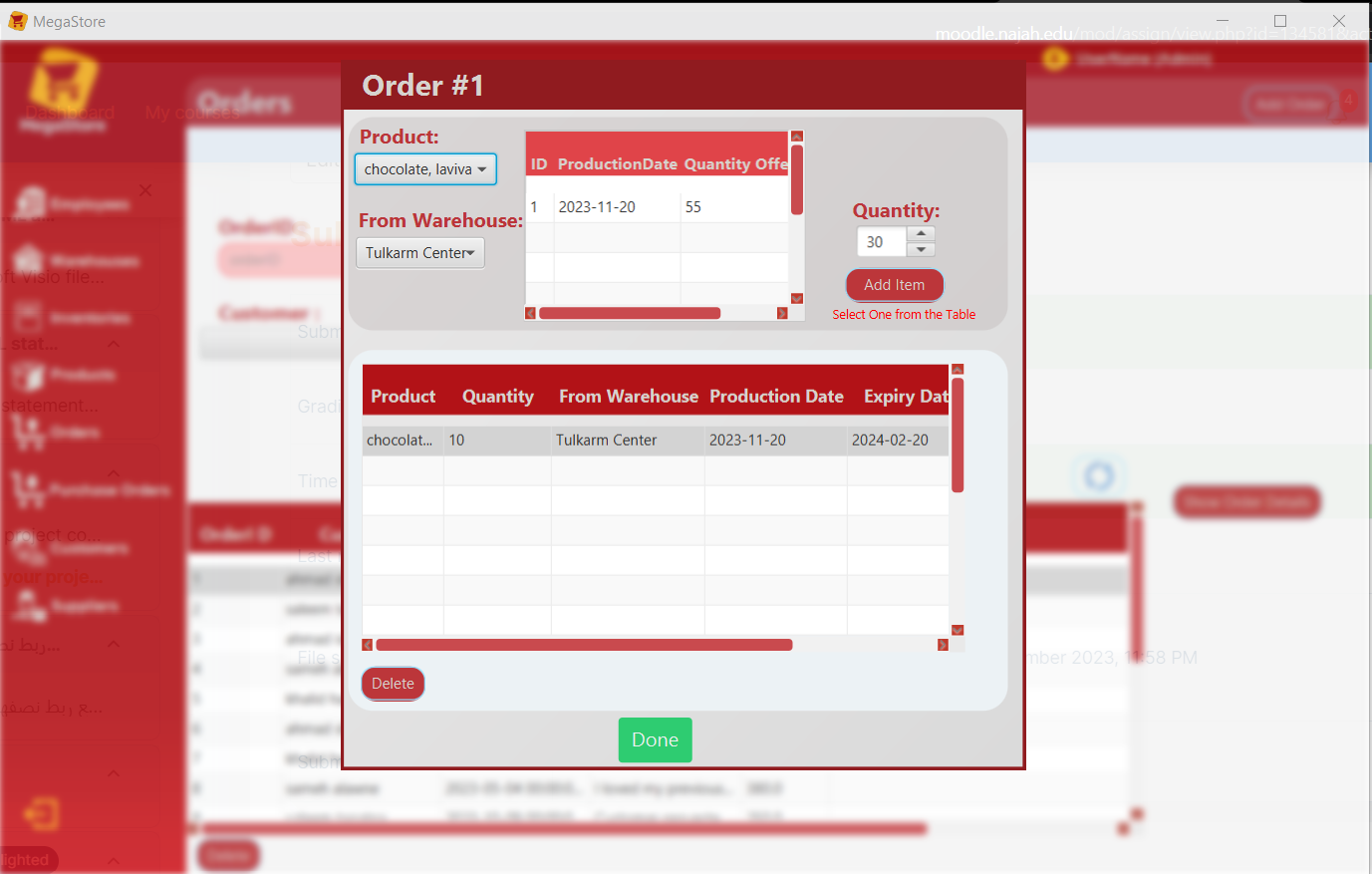


****

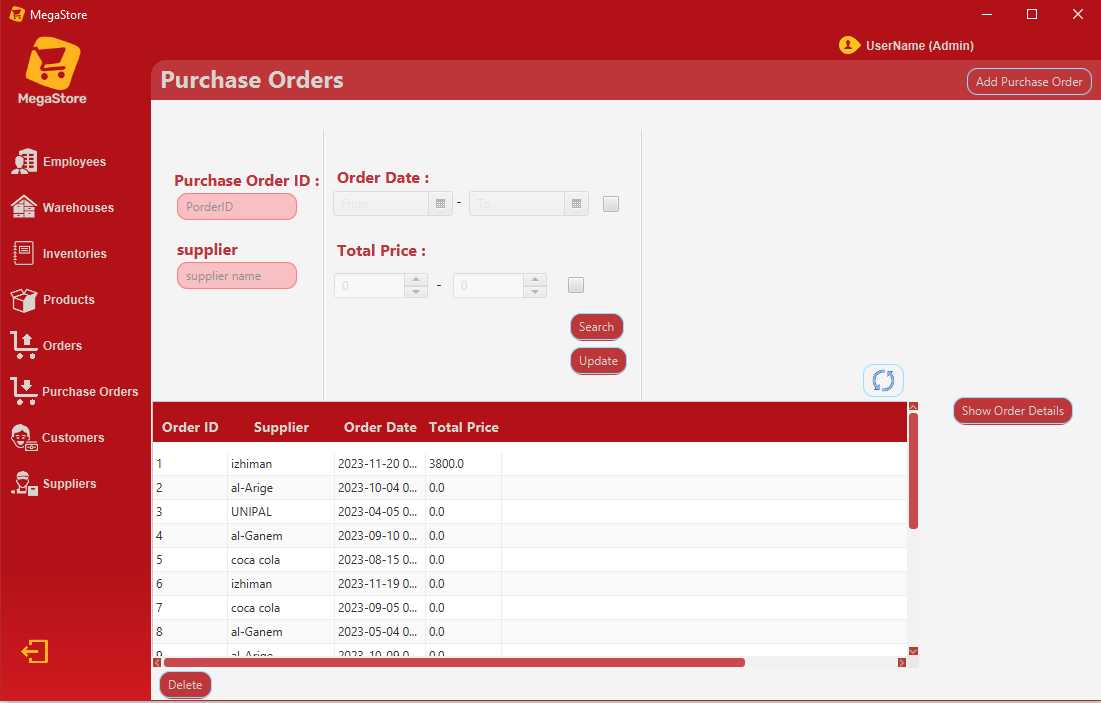
**Order Interface:**In addition to containing the operations found in previous interfaces, the Order &Purchase\_Order Interface has a new button and Popup(Show Order Details).

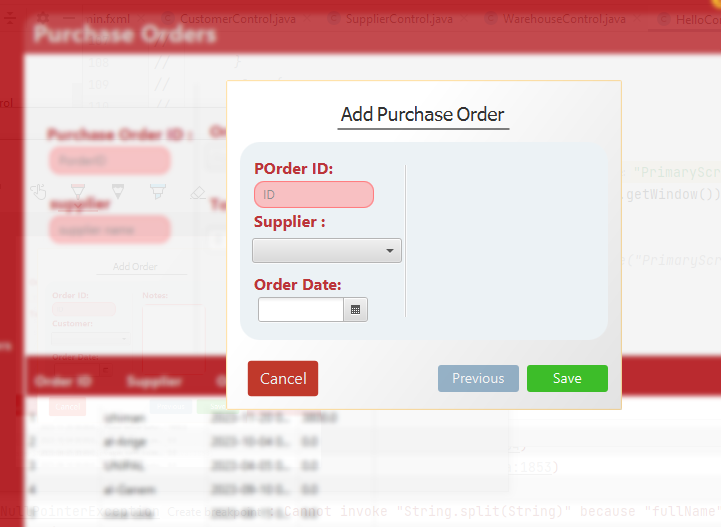
****

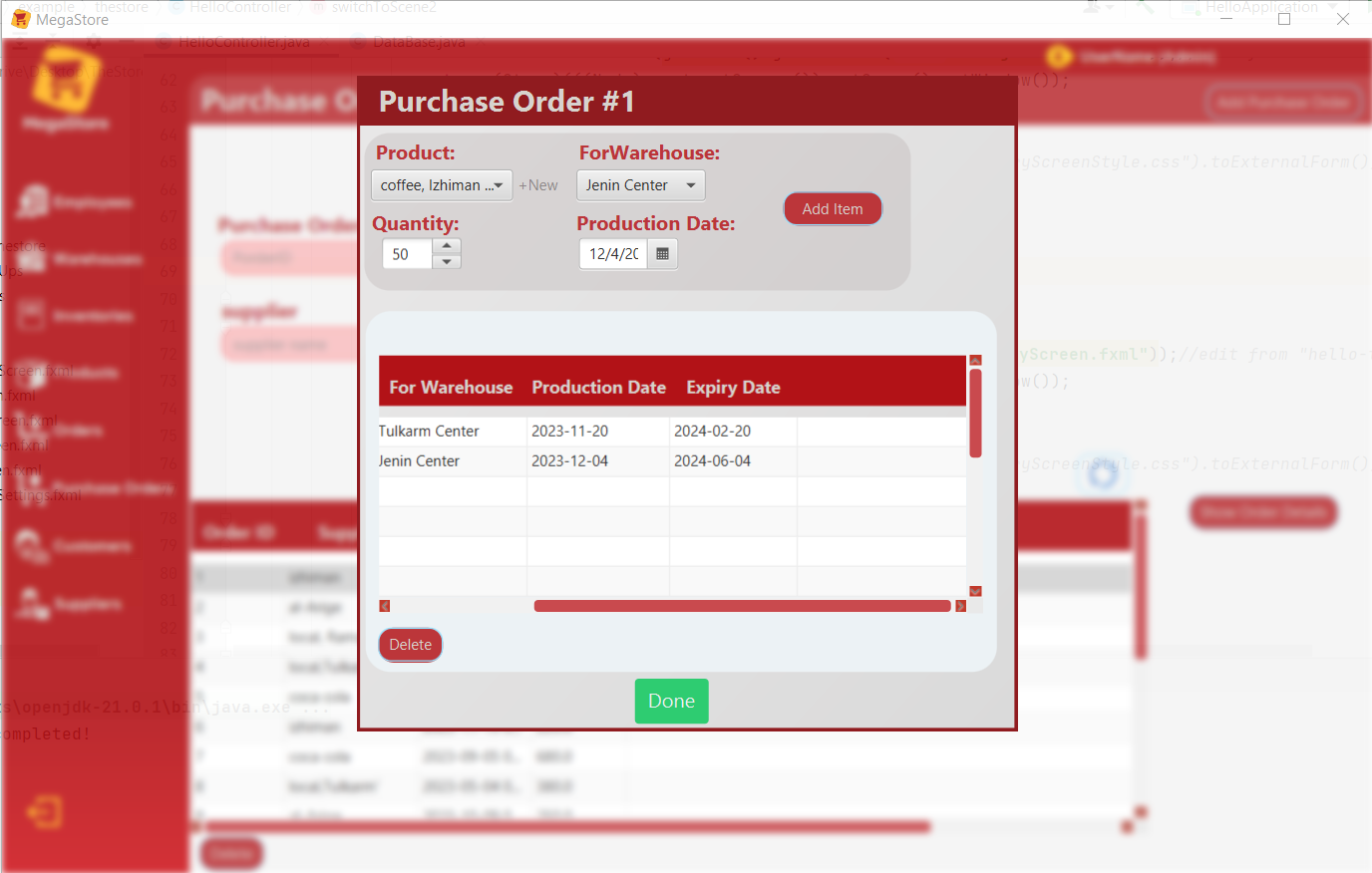
When the user select order and click show Order details button then a pop up will appear contains the order number, and order details(products in order and their quanities and production date for each one)  
 **Order Details popup:**



Through this interface, the user can modify the order details, where he can add new products according to what is available in the Inventory , or delete existing products in the order.

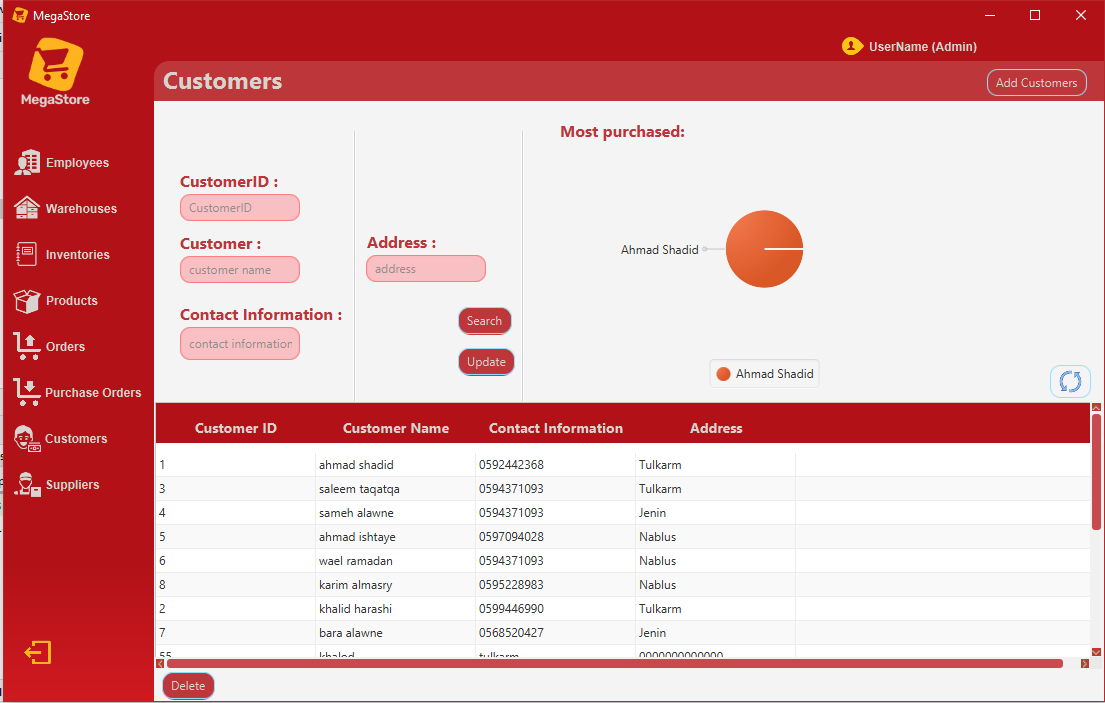
**Purchase Order Interface and its Adding popup and Show Order details popup :**



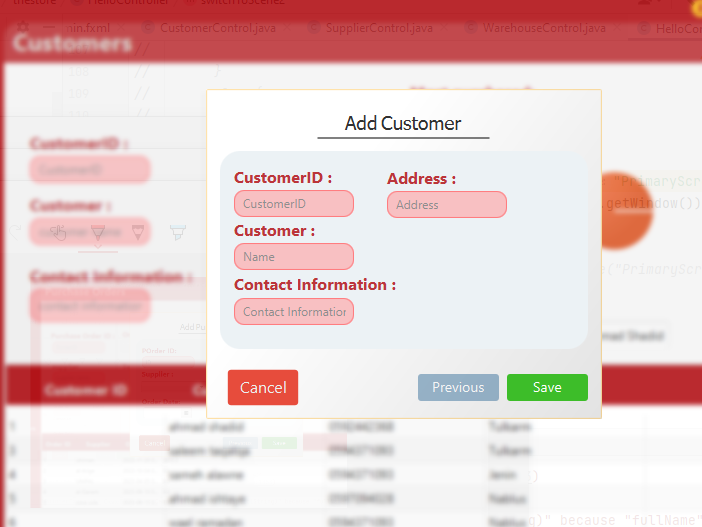


The same as Order, in show purchase order details the user can modify the purchase order details.

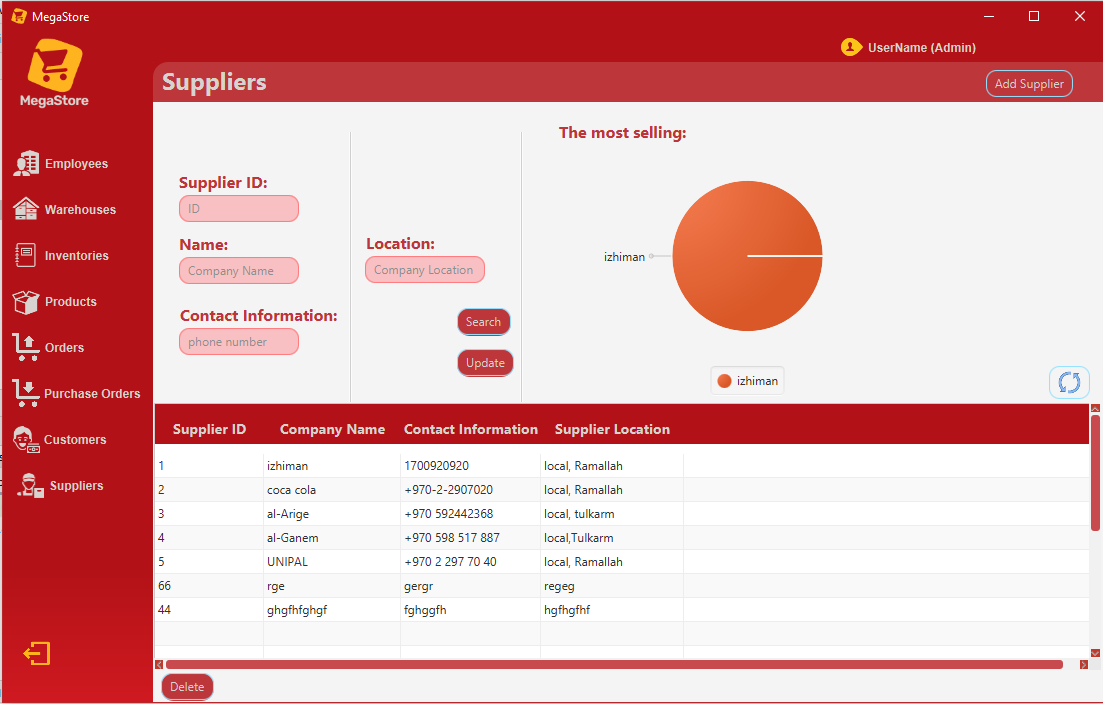
**Customer Interface and its Adding popup:**

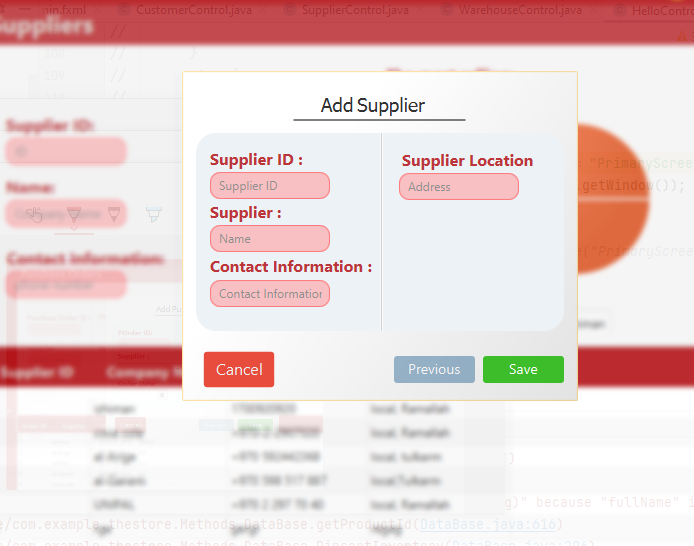


Customer interface contains Piechart Represents Purchasing ratios for each of them.



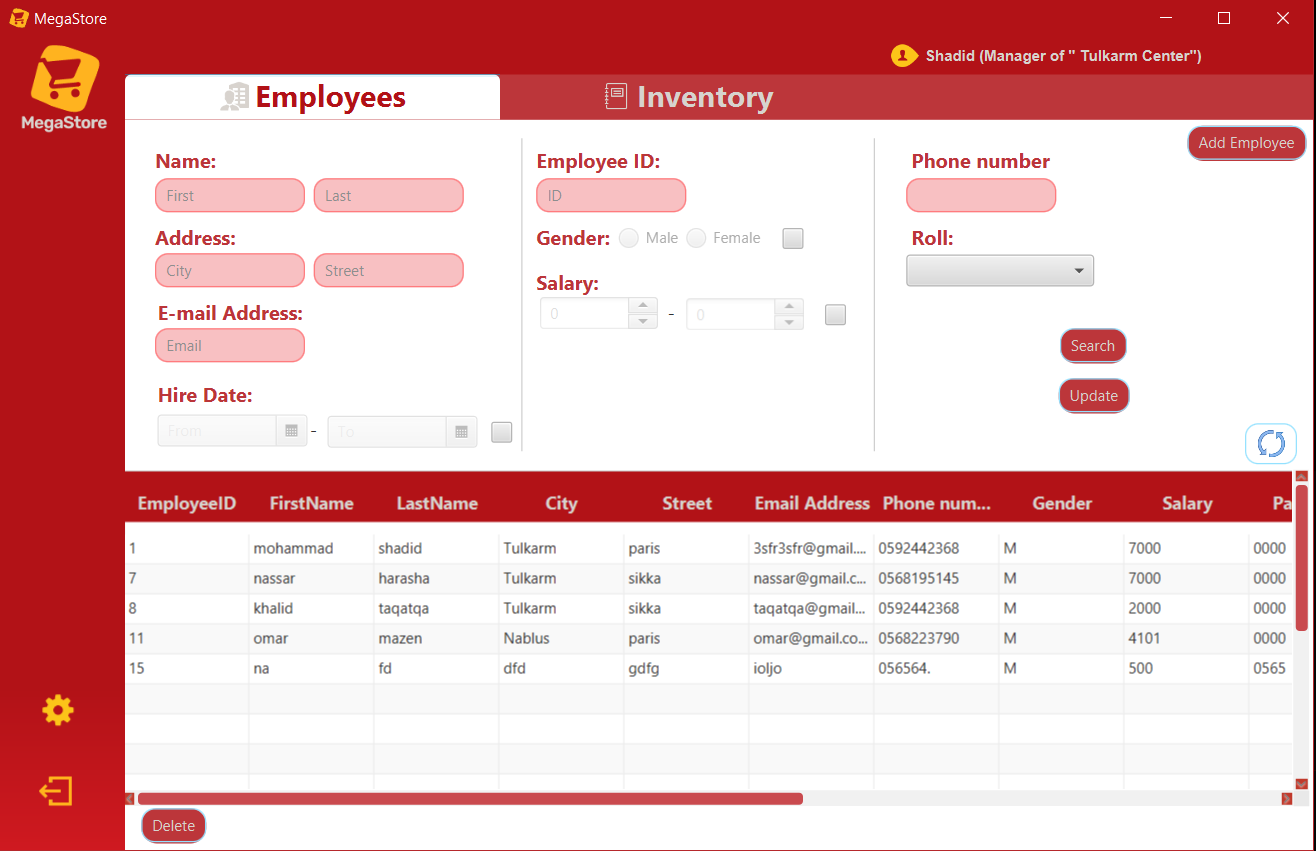
**Supplier Interface and its Adding popup:**

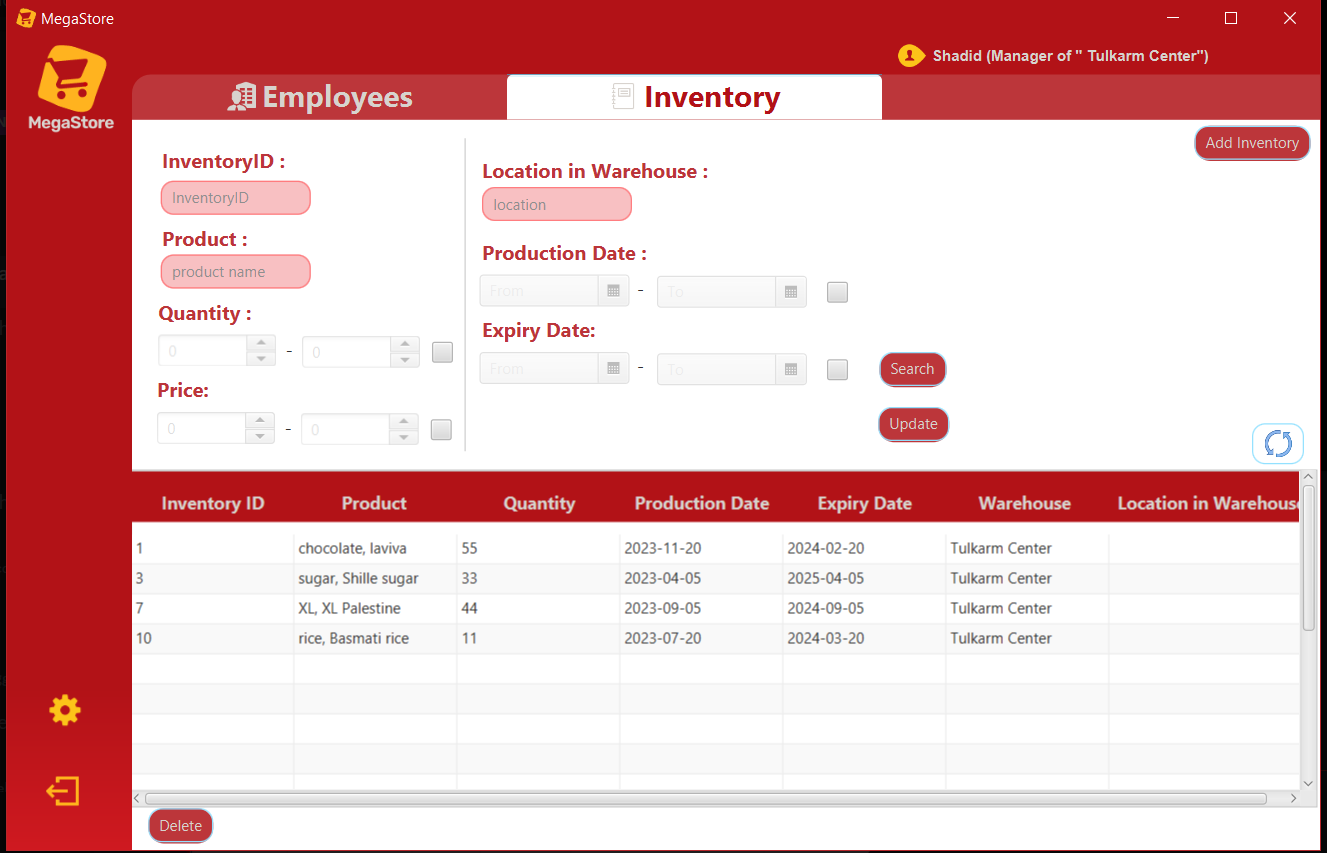


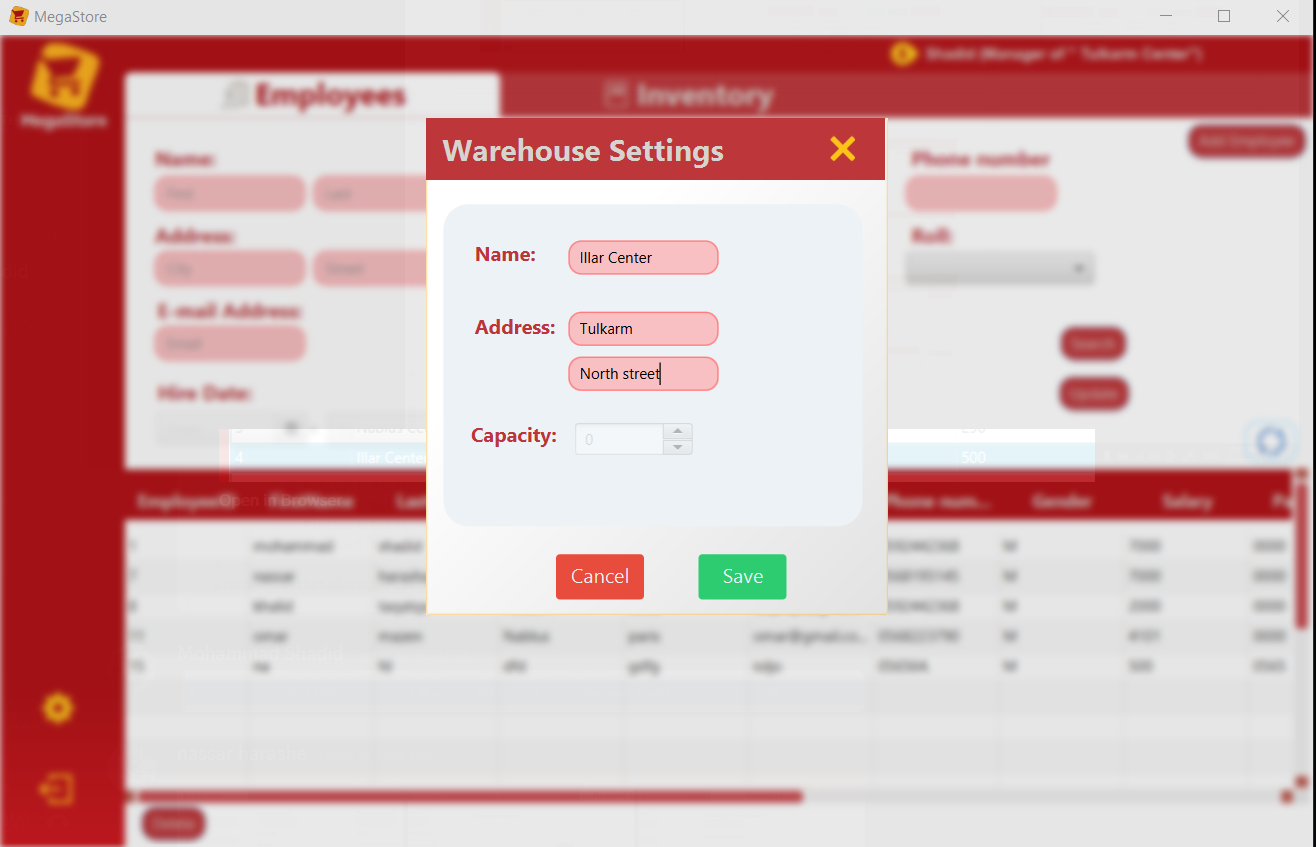


**Manager View :-**

Employee Interface: same as in Admin functionality (And adding popup)but for emplyees in the managing warehouse;



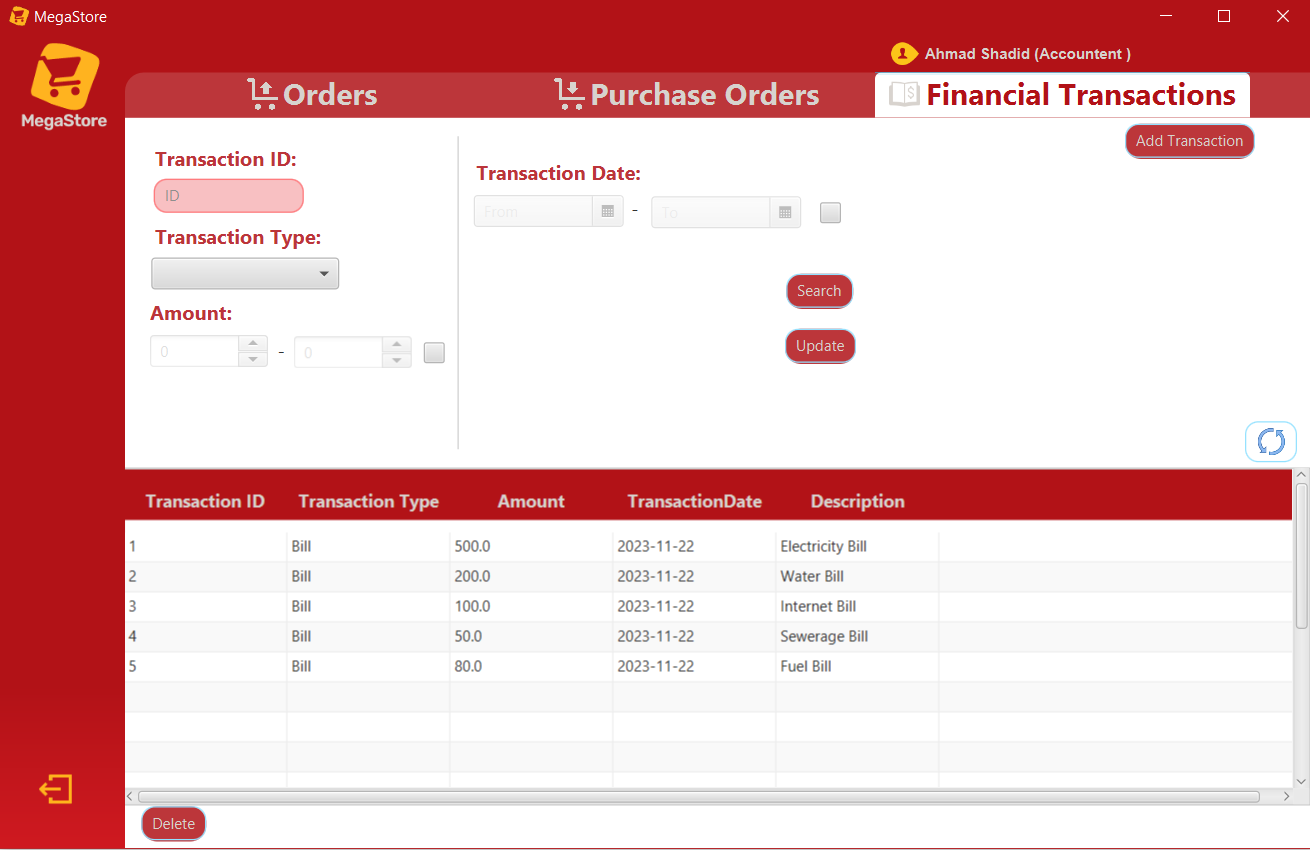
**Inventory Interface:** same as admin functionality(and adding popup) but for the inventories that exist in the managing warehouse.

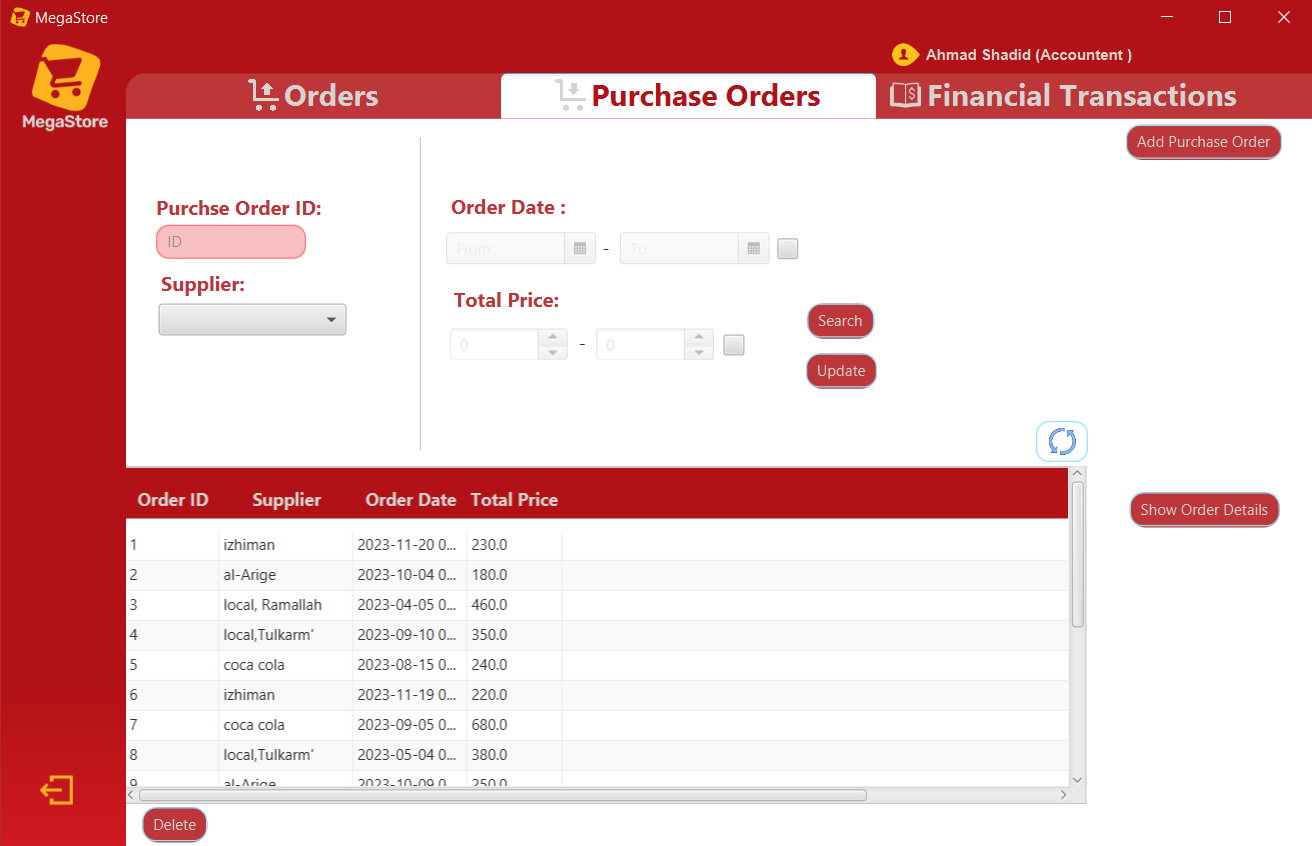


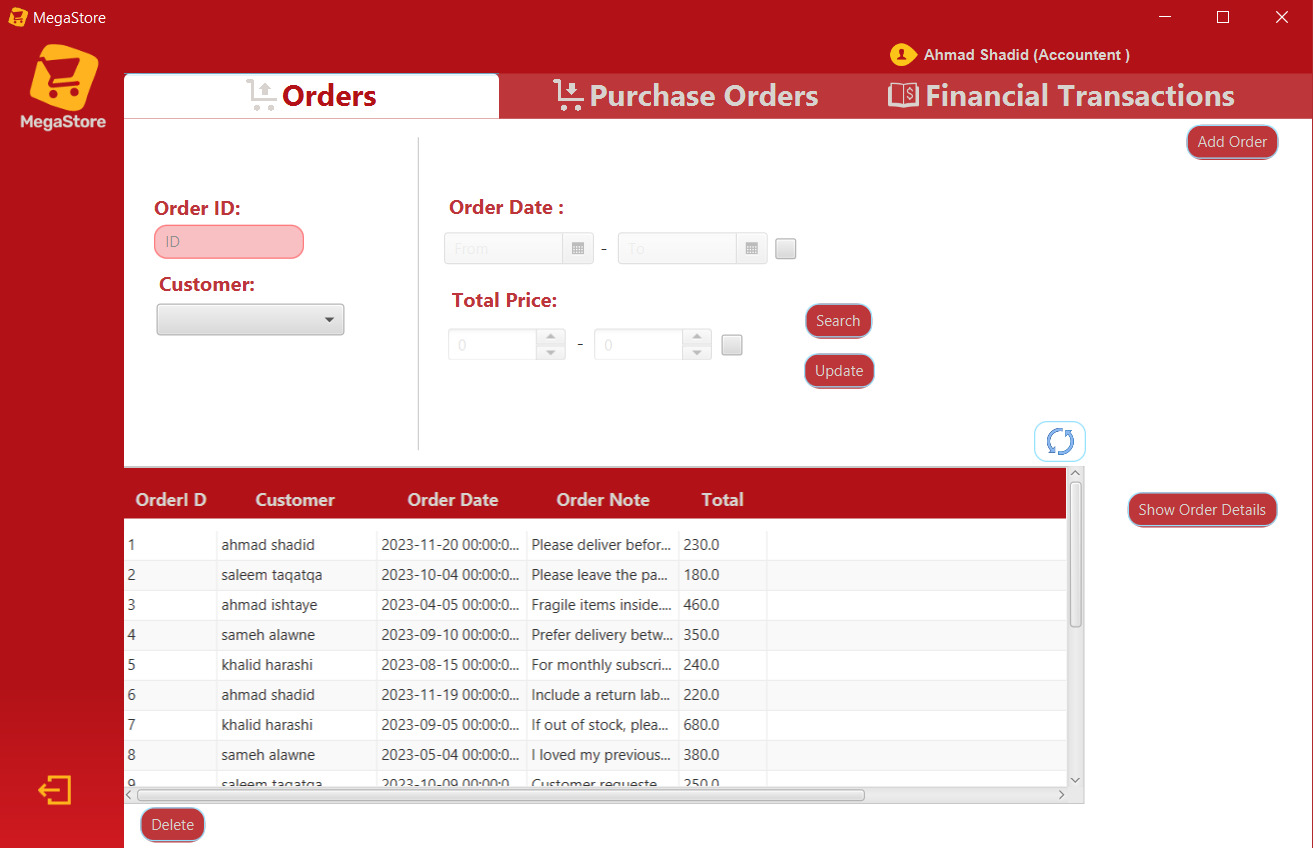
When click on setting button in below left of the view, then a popup warehouse setting   
will appear, using it the manager can edit on it is warehouse data.

**Accountant Interfaces:**

**Financial Transaction interface:** this is an interface especially for the accountant, having same previous functionality.



**purchase order interface:** same as purchase order interface in admin view.

**Order :**same as order interface in admin view.

**Conclusion**

In conclusion, the development of the Wholesale Store (MegaStore) project has been a comprehensive journey encompassing various facets of database management and JavaFX application development. The integration of a PostgreSQL database through pgAdmin has provided a robust infrastructure for storing and retrieving data, mirroring the intricacies of a real-world wholesale store.

Through the creation of tables, establishment of relationships, and implementation of constraints, we have designed a database schema that accurately models the store's organizational structure. This schema, coupled with the intuitive JavaFX user interface, forms a dynamic and efficient system for managing employees, products, warehouses, and transactions.

As we conclude this project, we recognize the potential for further enhancements, expansions, and adaptations to cater to evolving business needs. The Wholesale Store (MegaStore) project not only showcases technical proficiency but also exemplifies the fusion of technology and business acumen to create a comprehensive and impactful solution for store management.

**References:**

* PostgresSql documentation.
* GeeksforGeeks [(click here)](https://www.oracle.com).
* Stack overflow [(click here)](https://stackoverflow.com).
* W3school [(click here)](https://www.w3schools.com).
* YouTube channel: Bro Code.
* Database slides.
* Maven repository [(click here)](https://mvnrepository.com).
* Tutorials Point [(click here)](https://www.tutorialspoint.com/index.htm).