# ITE442 Final Report

Toy Shop Management System

Members:

Watsamon Phongwanit 2010160002

Karnsiree Karnka 2010160003

Phachathakorn Prathumma 2010070003

Krittawit Suriyawong 2010140010

## 

## **1. Project Overview**

Our Toy shop is based in Bangkok where most businesses are raising and central business hubs are. It is located in Thailand. Our shop is a Toy shop where most imported toys are available for any customers such as mystery boxes, figures and robots etc.

Our group constructed a database for Toy Shop and established a database integrated application to implement CRUD queries and manage our staff / stocks / orders,and sales.

## **2. Project Scope**

### **2.1 Application Objectives**

Database integrated store management application project composed of 6 pages.

1. Login page : username and password input and a log in button. The log in will be successful only if that username and password exist in the database.
2. Main menu page: have 4 options which leads to another page including staff,sales,orders and stock.
3. Staff page : provide a data input of staff which allows the user to add/delete/update the details of staff table within the database. A table of current detail of staff is also presented.
4. Stock page : presents a table of stocks and the out of stocks table from the database is shown. This is where a trigger is , if stocks quantity turned into 0, the item should appeared on the out of stocks table within the database.
5. Orders page: a table of orders showing a order details and status of delivered/not delivered. The user can also update the delivery status with the update button.
6. Sales page : page showing total sales

### **2.2 Business Rules**

*User:*

Business Rule: Only registered users with valid login credentials (username and password) can access the application.

Usernames and passwords must match the records stored in the database for successful login.

*Employee:*

Business Rule: Staff members are employees of the store.

Staff details should be entered accurately, and any changes to staff information should be reflected in the database.

*Stock:*

Business Rule: Stock represents the available quantity of products in the store.

Stock quantities should be updated whenever there is a change in product availability, such as receiving new stock or selling products.

*Order:*

Business Rule: Orders are placed by customers for purchasing products.

Validation Rule: Order details should be recorded accurately, including the customer ID, product ID, and delivery status.

*OrderItem:*

Business Rule: OrderItem are items available for sale in the store.

Product details should be accurate and up-to-date, including the product name, price, quantity and supplier information.

*Out of Stock:*

Business Rule: Out of Stock represents products that have reached a quantity of 0 in the store.

Products with a quantity of 0 should be automatically added to the Out of Stock table.

*Supplier:*

Business Rule: Suppliers provide the store with products.

Supplier details should be accurately recorded and associated with the relevant products.

## **3. Platform**

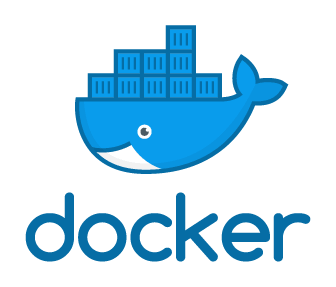
### Database System Architecture: Centralized

### **3.1 Operating System**

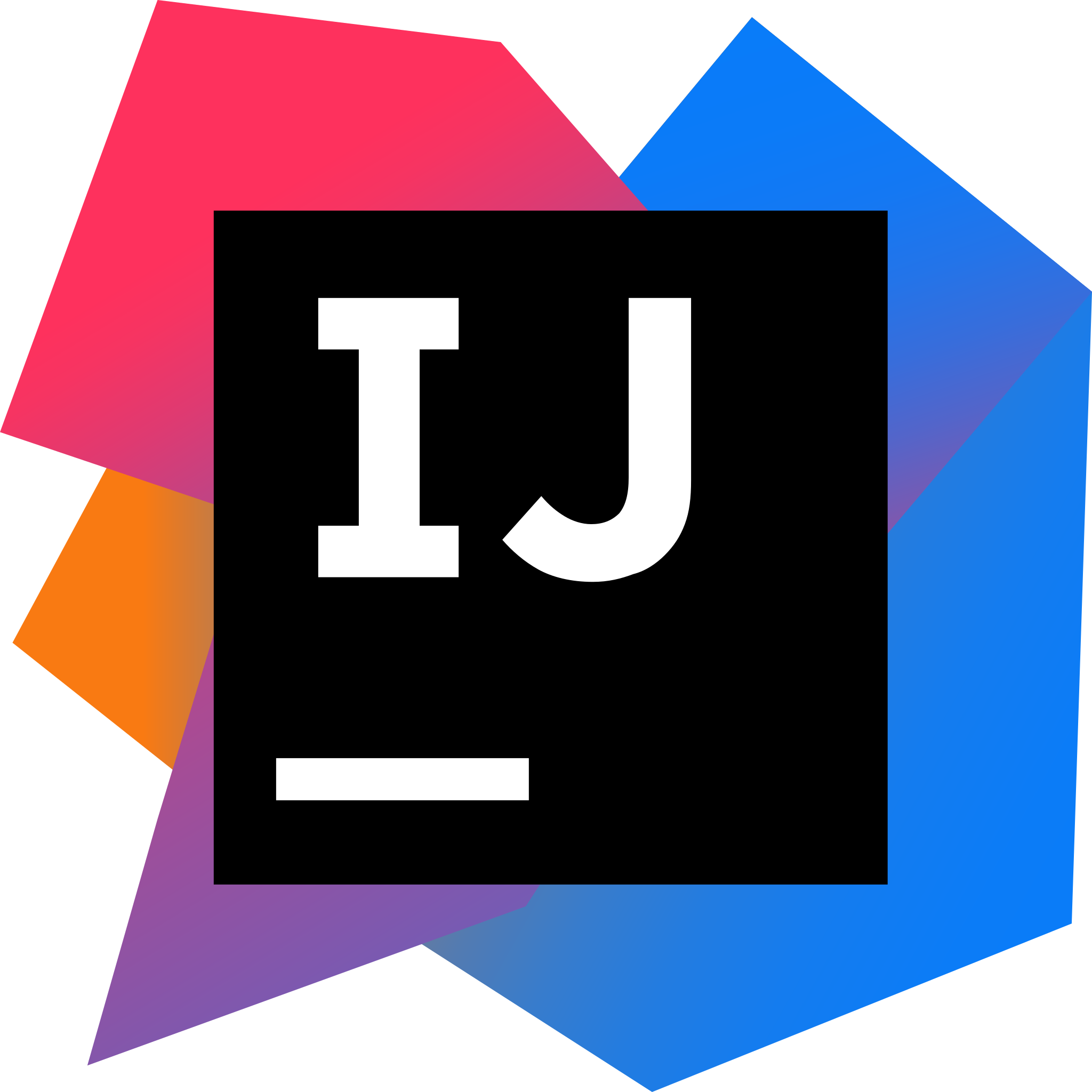
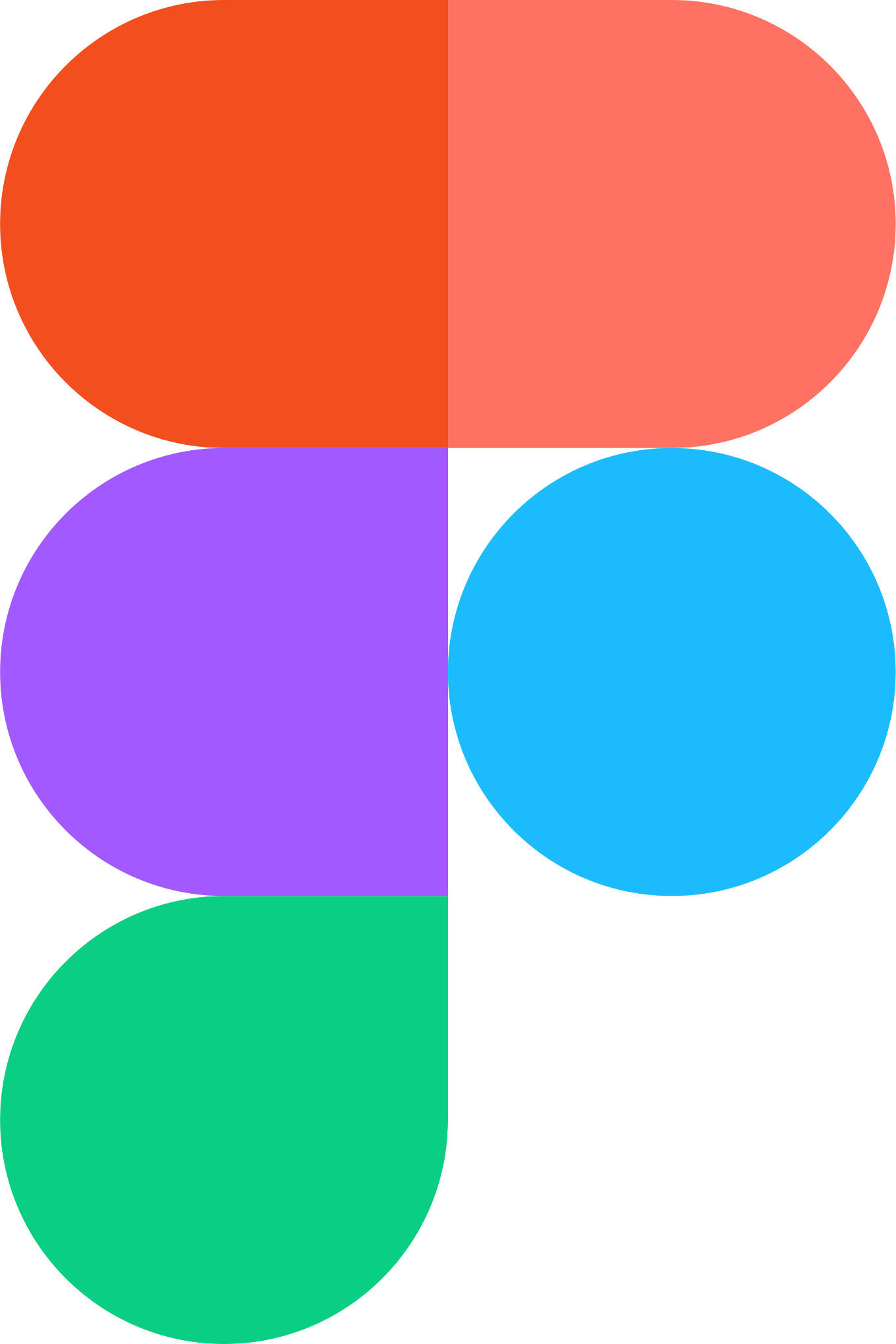
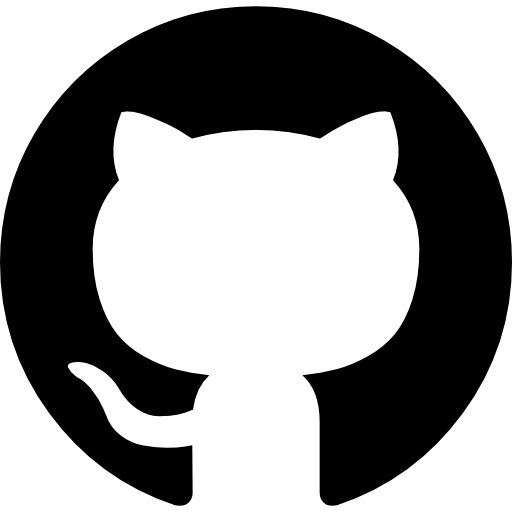
Our operating system is Windows. The latest version of Windows is Windows 11. Therefore, every application is windows-based.



**3.2 Database Management System**

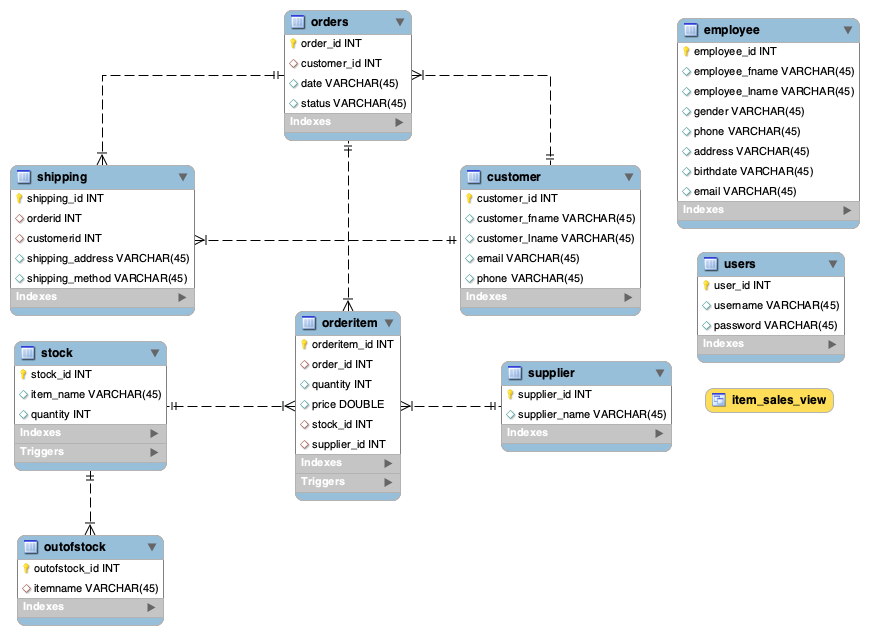
For database management systems, we use Docker Desktop, SQL workbench. Docker desktop's purpose is to provide an constant environment for creating an SQL database image. We use Docker Desktop to create a container for our database to maintain a constant environment for each member in a group and avoid difficulties which often caused by each member individually download different versions dependencies leading to unexpected errors and conflicts in the queries. As for SQL workbench, MySQL Workbench allows you to create and manipulate data, create models, reverse/forward engineer a live database , and create and edit tables and insert data. We run queries to manipulate data into SQL.

### **3.3 Application Development Tools**

For Application Development Tools, we use Intellij (JavaFX), Github, Scene Builder, and Figma. First, we use Figma to design our application and create prototypes then we use an IntelliJ IDEA IDE to develop our application and implement codes with java programming language to connect the applciation to database. GitHub is a platform for hosting code that allows version control and collaboration among team members. It allows each member to collaborate on projects from anywhere by commit , push and pull to update the recent changes. Scene Builder were used to assist the process of creating a layout for the actual user interface .

## **4. Database Design**

### **4.1 Entity Relationship Diagram (ER-D)**



### **4.2 Relational Schema**

### 

### **4.3 Normalization Process**

1. Table customer:

Identify functional dependencies:

customer\_id → customer\_fname, customer\_lname, email, phone

Normalize into 2NF:

Create a new table customer\_info with columns:

customer\_id, customer\_fname, customer\_lname

Create another table customer\_contact with columns:

customer\_id, email, phone

Use customer\_id as the primary key in both tables.

Normalize into 3NF:

No further normalization is required as there are no transitive dependencies.

1. Table users:

Identify functional dependencies:

user\_id → username, password

Normalize into 2NF:

No further normalization is required as the table is already in 2NF.

Normalize into 3NF:

No further normalization is required as there are no transitive dependencies.

1. Table employee:

Identify functional dependencies:

employee\_id → employee\_fname, employee\_lname, gender, phone, address, birthdate, email

Normalize into 2NF:

Create a new table employee\_info with columns:

employee\_id, employee\_fname, employee\_lname, gender, birthdate

Create another table employee\_contact with columns:

employee\_id, phone, address, email

Use employee\_id as the primary key in both tables.

Normalize into 3NF:

No further normalization is required as there are no transitive dependencies.

1. Table orders:

Identify functional dependencies:

order\_id → customer\_id, date, status

Normalize into 2NF:

No further normalization is required as the table is already in 2NF.

Normalize into 3NF:

No further normalization is required as there are no transitive dependencies.

1. Table supplier:

Identify functional dependencies:

supplier\_id → supplier\_name

Normalize into 2NF:

No further normalization is required as the table is already in 2NF.

Normalize into 3NF:

No further normalization is required as there are no transitive dependencies.

1. Table stock:

Identify functional dependencies:

stock\_id → item\_name, quantity

Normalize into 2NF:

No further normalization is required as the table is already in 2NF.

Normalize into 3NF:

No further normalization is required as there are no transitive dependencies.

1. Table orderitem:

Identify functional dependencies:

orderitem\_id → order\_id, quantity, price, stock\_id, supplier\_id

Normalize into 2NF:

Create a new table order\_item\_details with columns:

orderitem\_id, quantity, price

Use orderitem\_id as the primary key in the new table.

Modify the orderitem table to have columns:

orderitem\_id, order\_id, stock\_id, supplier\_id

Use orderitem\_id as the primary key in the modified table.

Normalize into 3NF:

No further normalization is required as there are no transitive dependencies.

### **4.4 Data Dictionary**

| TABLE NAME | Orders | | | | | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| General Characteristics | | | | | | | | | | | Constraints | | |
| No. | Attribute Name | Data Type | Data Size | Descriptions | | | Attribute Domain | Example | Primary Key | Nullable | Foreign Key | | |
| Format | Possible Values | Yes/No | Yes/No | Yes/No | Reference Table | Reference Attribute |
| 1 | order\_id | int | - | ID of order | | | None | 1 | Yes | No | No | - | - |
| 2 | customer\_id | int | - | ID of customer | | | None | 1 | No | Yes | Yes | Customer | customer\_id |
| 3 | date | varchar | 45 | date of each order | | | YYYY-MM-DD | 2023-05-06 | No | Yes | No | - | - |
| 4 | status | varchar | 45 | status of order | | | None | delivered | No | Yes | No | - | - |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| TABLE NAME | Stock | | | | | | | | | | | | |
| General Characteristics | | | | | | | | | | | Constraints | | |
| No. | Attribute Name | Data Type | Data Size | Descriptions | | | Attribute Domain | Example | Primary Key | Nullable | Foreign Key | | |
| Format | Possible Values | Yes/No | Yes/No | Yes/No | Reference Table | Reference Attribute |
| 1 | stocker\_id | int | - | ID of stock | | | None | 1 | Yes | No | No | - | - |
| 2 | item\_name | varchar | 45 | name of item | | | None | Cosmic Galaxy Bearbrick | No | Yes | No | - | - |
| 3 | quantity | int | - | quantity of item | | | None | 100 | No | Yes | No | - | - |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| TABLE NAME | Supplier | | | | | | | | | | | | |
| General Characteristics | | | | | | | | | | | Constraints | | |
| No. | Attribute Name | Data Type | Data Size | Descriptions | | | Attribute Domain | Example | Primary Key | Nullable | Foreign Key | | |
| Format | Possible Values | Yes/No | Yes/No | Yes/No | Reference Table | Reference Attribute |
| 1 | supplier\_id | int | - | ID of supplier | | | None | 1 | Yes | No | No | - | - |
| 2 | supplier\_name | varchar | 45 | name of supplier | | | None | Toyland Express | No | Yes | No | - | - |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| TABLE NAME | Users | | | | | | | | | | | | |
| General Characteristics | | | | | | | | | | | Constraints | | |
| No. | Attribute Name | Data Type | Data Size | Descriptions | | | Attribute Domain | Example | Primary Key | Nullable | Foreign Key | | |
| Format | Possible Values | Yes/No | Yes/No | Yes/No | Reference Table | Reference Attribute |
| 1 | user\_id | int | - | ID of user | | | None | 1 | Yes | No | No | - | - |
| 2 | username | varchar | 45 | user's name | | | None | bob | No | Yes | No | - | - |
| 3 | password | varchar | 45 | user's password | | | None | 4 | No | Yes | No | - | - |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| TABLE NAME | Customer | | | | | | | | | | | | |
| General Characteristics | | | | | | | | | | | Constraints | | |
| No. | Attribute Name | Data Type | Data Size | Descriptions | | | Attribute Domain | Example | Primary Key | Nullable | Foreign Key | | |
| Format | Possible Values | Yes/No | Yes/No | Yes/No | Reference Table | Reference Attribute |
| 1 | customer\_id | int | - | ID of customer | | | None | 1 | Yes | No | No | - | - |
| 2 | customer\_fname | varchar | 45 | customer first name | | | None | Emily | No | No | No | - | - |
| 3 | customer\_lname | varchar | 45 | customer last name | | | None | Johnson | No | No | No | - | - |
| 4 | email | varchar | 45 | customer email | | | None | emily.johnson@hotmail.com | No | No | No | - | - |
| 5 | phone | varchar | 45 | custome phone | | | None | 12345678 | No | No | No | - | - |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| TABLE NAME | Employee | | | | | | | | | | | | |
| General Characteristics | | | | | | | | | | | Constraints | | |
| No. | Attribute Name | Data Type | Data Size | Descriptions | | | Attribute Domain | Example | Primary Key | Nullable | Foreign Key | | |
| Format | Possible Values | Yes/No | Yes/No | Yes/No | Reference Table | Reference Attribute |
| 1 | employee\_id | int | - | ID of employee | | | None | 1 | Yes | No | No | - | - |
| 2 | employee\_fname | varchar | 45 | Employee First name | | | None | Harper | No | No | No | - | - |
| 3 | employee\_lname | varchar | 45 | Employee Last name | | | None | Bennett | No | No | No | - | - |
| 4 | gender | varchar | 45 | Gender of employee | | | None | Female | No | No | No | - | - |
| 5 | phone | varchar | 45 | Employee Phone number | | | None | 126479318 | No | No | No | - | - |
| 6 | address | varchar | 45 | Employee address | | | None | Chiang Mai | No | No | No | - | - |
| 7 | brithdate | varchar | 45 | Employee Brithdate | | | YYYY-MM-DD | 2001-05-02 | No | No | No | - | - |
| 8 | email | varchar | 45 | Employee Email | | | None | Harper@hotmail.com | No | No | No | - | - |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| TABLE NAME | Shipping | | | | | | | | | | | | |
| General Characteristics | | | | | | | | | | | Constraints | | |
| No. | Attribute Name | Data Type | Data Size | Descriptions | | | Attribute Domain | Example | Primary Key | Nullable | Foreign Key | | |
| Format | Possible Values | Yes/No | Yes/No | Yes/No | Reference Table | Reference Attribute |
| 1 | shipping\_id | int | - | ID of shipping | | | None | 1 | Yes | No | No | - | - |
| 2 | orderid | int | - | ID of order | | | None | 1 | No | No | Yes | Orders | order\_id |
| 3 | customerid | int | - | ID of customer | | | None | 1 | No | No | Yes | Customer | customer\_id |
| 4 | shipping\_address | varchar | 45 | shipping address | | | None | kolkata | No | No | No | - | - |
| 5 | shipping\_method | varchar | 45 | shipping method | | | None | plane | No | No | No | - | - |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| TABLE NAME | orderItem | | | | | | | | | | | | |
| General Characteristics | | | | | | | | | | | Constraints | | |
| No. | Attribute Name | Data Type | Data Size | Descriptions | | | Attribute Domain | Example | Primary Key | Nullable | Foreign Key | | |
| Format | Possible Values | Yes/No | Yes/No | Yes/No | Reference Table | Reference Attribute |
| 1 | orderitem\_id | int | - | ID of OrderItem | | | None | 1 | Yes | No | No | - | - |
| 2 | orderid | int | - | ID of order | | | None | 1 | No | No | Yes | Orders | order\_id |
| 3 | quantity | int | - | order quantity | | | None | 11 | No | No | No | - | - |
| 4 | price | double | - | price | | | None | 2500 | No | No | No | - | - |
| 5 | stock\_id | int | - | ID of stock | | | None | 1 | No | No | Yes | Stock | stock\_id |
| 5 | supplier\_id | int | - | ID of supplier | | | None | 1 | No | No | Yes | Supplier | supplier\_id |

### 

### 

### **4.5 Structured Query Language**

### **4.5.1 Data Definition Language (DDL) of all tables and views**

* Create schema ‘storemanagement’

-- -----------------------------------------------------

-- Schema storemanagement

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `storemanagement` DEFAULT CHARACTER SET utf8 ;

USE `storemanagement` ;

* Create table ‘customer’

-- -----------------------------------------------------

-- Table `storemanagement`.`customer`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `storemanagement`.`customer` (

`customer\_id` INT NOT NULL,

`customer\_fname` VARCHAR(45) NULL,

`customer\_lname` VARCHAR(45) NULL,

`email` VARCHAR(45) NULL,

`phone` VARCHAR(45) NULL,

PRIMARY KEY (`customer\_id`))

ENGINE = InnoDB;

* Create table ‘user’

-- -----------------------------------------------------

-- Table `storemanagement`.`user`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `storemanagement`.`users` (

`user\_id` INT NOT NULL,

`username` VARCHAR(45) NULL,

`password` VARCHAR(45) NULL,

PRIMARY KEY (`user\_id`))

ENGINE = InnoDB;

* Create table ‘employee’

-- -----------------------------------------------------

-- Table `storemanagement`.`employee`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `storemanagement`.`employee` (

`employee\_id` INT NOT NULL,

`employee\_fname` VARCHAR(45) NULL,

`employee\_lname` VARCHAR(45) NULL,

`gender` VARCHAR(45) NULL,

`phone` VARCHAR(45) NULL,

`address` VARCHAR(45) NULL,

`birthdate` VARCHAR(45) NULL,

`email` VARCHAR(45) NULL,

PRIMARY KEY (`employee\_id`))

ENGINE = InnoDB;

* Create table ‘orders’

-- -----------------------------------------------------

-- Table `storemanagement`.`order`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `storemanagement`.`orders` (

`order\_id` INT NOT NULL,

`customer\_id` INT NULL,

`date` VARCHAR(45) NULL,

`status` VARCHAR(45) NULL,

PRIMARY KEY (`order\_id`),

INDEX `customer\_id\_idx` (`customer\_id` ASC) VISIBLE,

CONSTRAINT `customer\_id`

FOREIGN KEY (`customer\_id`)

REFERENCES `storemanagement`.`customer` (`customer\_id`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

* Create table ‘supplier’

-- -----------------------------------------------------

-- Table `storemanagement`.`supplier`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `storemanagement`.`supplier` (

`supplier\_id` INT NOT NULL,

`supplier\_name` VARCHAR(45) NULL,

PRIMARY KEY (`supplier\_id`))

ENGINE = InnoDB;

* Create table ‘stock’

-- -----------------------------------------------------

-- Table `storemanagement`.`stock`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `storemanagement`.`stock` (

`stock\_id` INT NOT NULL,

`item\_name` VARCHAR(45) NULL,

`quantity` INT NULL,

PRIMARY KEY (`stock\_id`),

UNIQUE INDEX `item\_name\_UNIQUE` (`item\_name` ASC) VISIBLE)

ENGINE = InnoDB;

* Create table ‘orderitem’

-- -----------------------------------------------------

-- Table `storemanagement`.`orderitem`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `storemanagement`.`orderitem` (

`orderitem\_id` INT NOT NULL,

`order\_id` INT NULL,

`quantity` INT NULL,

`price` DOUBLE NULL,

`stock\_id` INT NULL,

`supplier\_id` INT NULL,

PRIMARY KEY (`orderitem\_id`),

INDEX `order\_id\_idx` (`order\_id` ASC) VISIBLE,

INDEX `supplier\_id\_idx` (`supplier\_id` ASC) VISIBLE,

INDEX `stock\_id\_idx` (`stock\_id` ASC) VISIBLE,

CONSTRAINT `order\_id`

FOREIGN KEY (`order\_id`)

REFERENCES `storemanagement`.`orders` (`order\_id`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `supplier\_id`

FOREIGN KEY (`supplier\_id`)

REFERENCES `storemanagement`.`supplier` (`supplier\_id`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `stock\_id`

FOREIGN KEY (`stock\_id`)

REFERENCES `storemanagement`.`stock` (`stock\_id`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

* Create table ‘outofstock’

-- -----------------------------------------------------

-- Table `storemanagement`.`outofstock`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `storemanagement`.`outofstock` (

`outofstock\_id` INT NOT NULL,

`itemname` VARCHAR(45) NULL,

PRIMARY KEY (`outofstock\_id`),

UNIQUE INDEX `item\_name\_UNIQUE` (`itemname` ASC) VISIBLE,

CONSTRAINT `itemname`

FOREIGN KEY (`itemname`)

REFERENCES `storemanagement`.`stock` (`item\_name`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

* Create table ‘shipping’

-- -----------------------------------------------------

-- Table `storemanagement`.`shipping`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `storemanagement`.`shipping` (

`shipping\_id` INT NOT NULL,

`orderid` INT NULL,

`customerid` INT NULL,

`shipping\_address` VARCHAR(45) NULL,

`shipping\_method` VARCHAR(45) NULL,

PRIMARY KEY (`shipping\_id`),

UNIQUE INDEX `shipping\_address\_UNIQUE` (`shipping\_address` ASC) VISIBLE,

INDEX `order\_id\_idx` (`orderid` ASC) VISIBLE,

INDEX `customer\_id\_idx` (`customerid` ASC) VISIBLE,

CONSTRAINT `orderid`

FOREIGN KEY (`orderid`)

REFERENCES `storemanagement`.`orders` (`order\_id`)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT `customerid`

FOREIGN KEY (`customerid`)

REFERENCES `storemanagement`.`customer` (`customer\_id`)

ON DELETE CASCADE

ON UPDATE CASCADE)

ENGINE = InnoDB;

* Create view ‘item\_sales\_view’

-- -----------------------------------------------------

-- View `item\_sales\_view`.

-- -----------------------------------------------------

CREATE VIEW item\_sales\_view AS

SELECT a.stock\_id,b.item\_name,a.price,a.quantity, SUM(a.quantity \* a.price) AS total\_sales

FROM orderitem AS a,stock AS b

WHERE a.stock\_id = b.stock\_id

GROUP BY stock\_id,item\_name,price,quantity

ORDER BY total\_sales DESC;

SELECT \* FROM item\_sales\_view;

### **4.5.2 Data Manipulation Language (DML)**

* Insert information

-- -----------------------------------------------------

-- Table `storemanagement`.`users`

-- -----------------------------------------------------

INSERT INTO `storemanagement`.`users` (`user\_id`, `username`, `password`) VALUES ('1', 'arm', '1234');

INSERT INTO `storemanagement`.`users` (`user\_id`, `username`, `password`) VALUES ('2', 'phufah', '123');

INSERT INTO `storemanagement`.`users` (`user\_id`, `username`, `password`) VALUES ('3', 'kidcat', '4');

INSERT INTO `storemanagement`.`users` (`user\_id`, `username`, `password`) VALUES ('4', 'pear', '15235');

INSERT INTO `storemanagement`.`users` (`user\_id`, `username`, `password`) VALUES ('5', 'dodo', '151dsd');

INSERT INTO `storemanagement`.`users` (`user\_id`, `username`, `password`) VALUES ('6', 'albert', '231231');

INSERT INTO `storemanagement`.`users` (`user\_id`, `username`, `password`) VALUES ('7', 'bob', '1999');

INSERT INTO `storemanagement`.`users` (`user\_id`, `username`, `password`) VALUES ('8', 'tedison', '45858');

INSERT INTO `storemanagement`.`users` (`user\_id`, `username`, `password`) VALUES ('9', 'josh', '4451');

INSERT INTO `storemanagement`.`users` (`user\_id`, `username`, `password`) VALUES ('10', 'ted', '187');

-- -----------------------------------------------------

-- Table `storemanagement`.`supplier`

-- -----------------------------------------------------

INSERT INTO `storemanagement`.`supplier` (`supplier\_id`, `supplier\_name`) VALUES ('1', 'FunZone Toys');

INSERT INTO `storemanagement`.`supplier` (`supplier\_id`, `supplier\_name`) VALUES ('2', 'Playful Pals Toy Co.');

INSERT INTO `storemanagement`.`supplier` (`supplier\_id`, `supplier\_name`) VALUES ('3', 'Toyland Express');

INSERT INTO `storemanagement`.`supplier` (`supplier\_id`, `supplier\_name`) VALUES ('4', 'Happy Kids Toys');

INSERT INTO `storemanagement`.`supplier` (`supplier\_id`, `supplier\_name`) VALUES ('5', 'MagicToy World');

INSERT INTO `storemanagement`.`supplier` (`supplier\_id`, `supplier\_name`) VALUES ('6', 'Toy Universe');

INSERT INTO `storemanagement`.`supplier` (`supplier\_id`, `supplier\_name`) VALUES ('7', 'WonderToy Company');

INSERT INTO `storemanagement`.`supplier` (`supplier\_id`, `supplier\_name`) VALUES ('8', 'Dreamy Dolls and Toys');

INSERT INTO `storemanagement`.`supplier` (`supplier\_id`, `supplier\_name`) VALUES ('9', 'Adventure Playtime Supplies');

INSERT INTO `storemanagement`.`supplier` (`supplier\_id`, `supplier\_name`) VALUES ('10', 'Creative Kids Toy Co.');

-- -----------------------------------------------------

-- Table `storemanagement`.`customer`

-- -----------------------------------------------------

INSERT INTO `storemanagement`.`customer` (`customer\_id`, `customer\_fname`, `customer\_lname`, `email`, `phone`) VALUES ('1', 'Emily', 'Johnson', 'emily.johnson@hotmail.com', '12345678');

INSERT INTO `storemanagement`.`customer` (`customer\_id`, `customer\_fname`, `customer\_lname`, `email`, `phone`) VALUES ('2', 'Liam', 'Smith', 'liam.smith@hotmail.com', '98765432');

INSERT INTO `storemanagement`.`customer` (`customer\_id`, `customer\_fname`, `customer\_lname`, `email`, `phone`) VALUES ('3', 'Olivia', 'Brown', 'olivia.brown@hotmail.com', '53697415');

INSERT INTO `storemanagement`.`customer` (`customer\_id`, `customer\_fname`, `customer\_lname`, `email`, `phone`) VALUES ('4', 'Noah', 'Davis', 'noah.davis@hotmail.com', '73169842');

INSERT INTO `storemanagement`.`customer` (`customer\_id`, `customer\_fname`, `customer\_lname`, `email`, `phone`) VALUES ('5', 'Ava', 'Wilson', 'ava.wilson@hotmail.com', '23697815');

INSERT INTO `storemanagement`.`customer` (`customer\_id`, `customer\_fname`, `customer\_lname`, `email`, `phone`) VALUES ('6', 'Ethan', 'Martinez', 'ethan.martinez@hotmail.com', '59731589');

INSERT INTO `storemanagement`.`customer` (`customer\_id`, `customer\_fname`, `customer\_lname`, `email`, `phone`) VALUES ('7', 'Sophia', 'Thompson ', 'sophia.thompson@hotmail.com', '23549813');

INSERT INTO `storemanagement`.`customer` (`customer\_id`, `customer\_fname`, `customer\_lname`, `email`, `phone`) VALUES ('8', 'Mason', 'Taylor', 'mason.taylor@hotmail.com', '26879512');

INSERT INTO `storemanagement`.`customer` (`customer\_id`, `customer\_fname`, `customer\_lname`, `email`, `phone`) VALUES ('9', 'Mia', 'Anderson', 'mia.anderson@hotmail.com', '12135498');

INSERT INTO `storemanagement`.`customer` (`customer\_id`, `customer\_fname`, `customer\_lname`, `email`, `phone`) VALUES ('10', 'Lucas', 'Clark', 'lucas.clark@hotmail.com', '85274196');

-- -----------------------------------------------------

-- Table `storemanagement`.`orders`

-- -----------------------------------------------------

INSERT INTO `storemanagement`.`orders` (`order\_id`, `customer\_id`, `date`, `status`) VALUES ('1', '1', '2023-05-06', 'delivered');

INSERT INTO `storemanagement`.`orders` (`order\_id`, `customer\_id`, `date`, `status`) VALUES ('2', '2', '2023-05-09', 'delivered');

INSERT INTO `storemanagement`.`orders` (`order\_id`, `customer\_id`, `date`, `status`) VALUES ('3', '3', '2023-05-11', 'delivered');

INSERT INTO `storemanagement`.`orders` (`order\_id`, `customer\_id`, `date`, `status`) VALUES ('4', '4', '2023-05-12', 'delivered');

INSERT INTO `storemanagement`.`orders` (`order\_id`, `customer\_id`, `date`, `status`) VALUES ('5', '5', '2023-05-15', 'delivered');

INSERT INTO `storemanagement`.`orders` (`order\_id`, `customer\_id`, `date`, `status`) VALUES ('6', '6', '2023-05-17', 'delivered');

INSERT INTO `storemanagement`.`orders` (`order\_id`, `customer\_id`, `date`, `status`) VALUES ('7', '7', '2023-05-19', 'delivered');

INSERT INTO `storemanagement`.`orders` (`order\_id`, `customer\_id`, `date`, `status`) VALUES ('8', '8', '2023-05-20', 'delivered');

INSERT INTO `storemanagement`.`orders` (`order\_id`, `customer\_id`, `date`, `status`) VALUES ('9', '9', '2023-05-22', 'delivered');

INSERT INTO `storemanagement`.`orders` (`order\_id`, `customer\_id`, `date`, `status`) VALUES ('10', '10', '2023-05-25', 'not delivered');

-- -----------------------------------------------------

-- Table `storemanagement`.`stock`

-- -----------------------------------------------------

INSERT INTO `storemanagement`.`stock` (`stock\_id`, `item\_name`, `quantity`) VALUES ('1', 'Marvel Legends Avengers Assemble', '100');

INSERT INTO `storemanagement`.`stock` (`stock\_id`, `item\_name`, `quantity`) VALUES ('2', 'Mighty Mjolnir Thor Statue', '100');

INSERT INTO `storemanagement`.`stock` (`stock\_id`, `item\_name`, `quantity`) VALUES ('3', 'Iron Man Mark XLII Limited Edition Figure', '100');

INSERT INTO `storemanagement`.`stock` (`stock\_id`, `item\_name`, `quantity`) VALUES ('4', 'Captain America Shield Collection', '100');

INSERT INTO `storemanagement`.`stock` (`stock\_id`, `item\_name`, `quantity`) VALUES ('5', 'Black Widow Deadly Origins Statue', '100');

INSERT INTO `storemanagement`.`stock` (`stock\_id`, `item\_name`, `quantity`) VALUES ('6', 'Hulk Smash Battle Set', '100');

INSERT INTO `storemanagement`.`stock` (`stock\_id`, `item\_name`, `quantity`) VALUES ('7', 'Golden Luxe Bearbrick', '100');

INSERT INTO `storemanagement`.`stock` (`stock\_id`, `item\_name`, `quantity`) VALUES ('8', 'Cosmic Galaxy Bearbrick', '100');

INSERT INTO `storemanagement`.`stock` (`stock\_id`, `item\_name`, `quantity`) VALUES ('9', 'Superman: Man of Steel Deluxe Figure', '100');

INSERT INTO `storemanagement`.`stock` (`stock\_id`, `item\_name`, `quantity`) VALUES ('10', 'The Flash: Speed Force Showcase', '100');

-- -----------------------------------------------------

-- Table `storemanagement`.`orderitem`

-- -----------------------------------------------------

INSERT INTO `storemanagement`.`orderitem` (`orderitem\_id`, `order\_id`, `quantity`, `price`, `stock\_id`, `supplier\_id`) VALUES ('1', '1', '11', '2500', '1', '1');

INSERT INTO `storemanagement`.`orderitem` (`orderitem\_id`, `order\_id`, `quantity`, `price`, `stock\_id`, `supplier\_id`) VALUES ('2', '2', '8', '2600', '2', '1');

INSERT INTO `storemanagement`.`orderitem` (`orderitem\_id`, `order\_id`, `quantity`, `price`, `stock\_id`, `supplier\_id`) VALUES ('3', '3', '9', '1800', '3', '1');

INSERT INTO `storemanagement`.`orderitem` (`orderitem\_id`, `order\_id`, `quantity`, `price`, `stock\_id`, `supplier\_id`) VALUES ('4', '4', '2', '800', '4', '1');

INSERT INTO `storemanagement`.`orderitem` (`orderitem\_id`, `order\_id`, `quantity`, `price`, `stock\_id`, `supplier\_id`) VALUES ('5', '5', '3', '1500', '5', '1');

INSERT INTO `storemanagement`.`orderitem` (`orderitem\_id`, `order\_id`, `quantity`, `price`, `stock\_id`, `supplier\_id`) VALUES ('6', '6', '5', '4000', '6', '1');

INSERT INTO `storemanagement`.`orderitem` (`orderitem\_id`, `order\_id`, `quantity`, `price`, `stock\_id`, `supplier\_id`) VALUES ('7', '7', '19', '3800', '7', '1');

INSERT INTO `storemanagement`.`orderitem` (`orderitem\_id`, `order\_id`, `quantity`, `price`, `stock\_id`, `supplier\_id`) VALUES ('8', '8', '6', '3600', '8', '1');

INSERT INTO `storemanagement`.`orderitem` (`orderitem\_id`, `order\_id`, `quantity`, `price`, `stock\_id`, `supplier\_id`) VALUES ('9', '9', '1', '5000', '9', '1');

INSERT INTO `storemanagement`.`orderitem` (`orderitem\_id`, `order\_id`, `quantity`, `price`, `stock\_id`, `supplier\_id`) VALUES ('10', '10', '3', '6000', '10', '1');

-- -----------------------------------------------------

-- Table `storemanagement`.`employee`

-- -----------------------------------------------------

INSERT INTO `storemanagement`.`employee` (`employee\_id`, `employee\_fname`, `employee\_lname`, `gender`, `phone`, `address`, `birthdate`, `email`) VALUES ('1', 'Harper', 'Bennett', 'Female', '126479318', 'Chiang Mai', '2001-05-02', 'Harper@hotmail.com');

INSERT INTO `storemanagement`.`employee` (`employee\_id`, `employee\_fname`, `employee\_lname`, `gender`, `phone`, `address`, `birthdate`, `email`) VALUES ('2', 'Samuel', 'Foster ', 'Male', '75631698', 'Chiang Rai', '2000-07-02', 'Samuel@hotmail.com');

INSERT INTO `storemanagement`.`employee` (`employee\_id`, `employee\_fname`, `employee\_lname`, `gender`, `phone`, `address`, `birthdate`, `email`) VALUES ('3', 'Lily', 'Russell', 'Female', '24681358', 'Bangkok', '1999-03-12', 'Lily@hotmail.com');

INSERT INTO `storemanagement`.`employee` (`employee\_id`, `employee\_fname`, `employee\_lname`, `gender`, `phone`, `address`, `birthdate`, `email`) VALUES ('4', 'Owen', 'Peterson', 'Male', '54301895', 'Bangkok', '1999-06-20', 'Owen@hotmail.com');

INSERT INTO `storemanagement`.`employee` (`employee\_id`, `employee\_fname`, `employee\_lname`, `gender`, `phone`, `address`, `birthdate`, `email`) VALUES ('5', 'Grace', 'Kelly ', 'Female', '03597519', 'Bangkok', '1999-08-01', 'Grace@hotmail.com');

INSERT INTO `storemanagement`.`employee` (`employee\_id`, `employee\_fname`, `employee\_lname`, `gender`, `phone`, `address`, `birthdate`, `email`) VALUES ('6', 'Noah', 'Sullivan', 'Male', '01357951', 'Bangkok', '1999-10-23', 'Noah@hotmail.com');

INSERT INTO `storemanagement`.`employee` (`employee\_id`, `employee\_fname`, `employee\_lname`, `gender`, `phone`, `address`, `birthdate`, `email`) VALUES ('7', 'Amelia', 'Richardson', 'Female', '01367981', 'Bangkok', '1998-11-23', 'Amelia@hotmail.com');

INSERT INTO `storemanagement`.`employee` (`employee\_id`, `employee\_fname`, `employee\_lname`, `gender`, `phone`, `address`, `birthdate`, `email`) VALUES ('8', 'Elijah', 'Hayes', 'Male', '54931818', 'Chiang Mai', '1998-12-11', 'Elijah@hotmail.com');

INSERT INTO `storemanagement`.`employee` (`employee\_id`, `employee\_fname`, `employee\_lname`, `gender`, `phone`, `address`, `birthdate`, `email`) VALUES ('9', 'Stella', 'Murphy', 'Female', '66688833', 'Chiang Rai', '1998-12-20', 'Stella@hotmail.com');

INSERT INTO `storemanagement`.`employee` (`employee\_id`, `employee\_fname`, `employee\_lname`, `gender`, `phone`, `address`, `birthdate`, `email`) VALUES ('10', 'Henry', 'Cooper', 'Male', '18479524', 'Chiang Mai', '1997-12-01', 'Henry@hotmail.com');

-- -----------------------------------------------------

-- Table `storemanagement`.`outofstock`

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Table `storemanagement`.`shipping`

-- -----------------------------------------------------

INSERT INTO `storemanagement`.`shipping` (`shipping\_id`, `orderid`, `customerid`, `shipping\_address`, `shipping\_method`) VALUES ('1', '1', '1', 'kolkata', 'plane');

INSERT INTO `storemanagement`.`shipping` (`shipping\_id`, `orderid`, `customerid`, `shipping\_address`, `shipping\_method`) VALUES ('2', '2', '2', 'Phuket', 'car');

INSERT INTO `storemanagement`.`shipping` (`shipping\_id`, `orderid`, `customerid`, `shipping\_address`, `shipping\_method`) VALUES ('3', '3', '3', 'Mumbai', 'plane');

INSERT INTO `storemanagement`.`shipping` (`shipping\_id`, `orderid`, `customerid`, `shipping\_address`, `shipping\_method`) VALUES ('4', '4', '4', 'Bangkok', 'car');

INSERT INTO `storemanagement`.`shipping` (`shipping\_id`, `orderid`, `customerid`, `shipping\_address`, `shipping\_method`) VALUES ('5', '5', '5', 'Chiang mai', 'car');

INSERT INTO `storemanagement`.`shipping` (`shipping\_id`, `orderid`, `customerid`, `shipping\_address`, `shipping\_method`) VALUES ('6', '6', '6', 'Chiang Rai', 'car');

INSERT INTO `storemanagement`.`shipping` (`shipping\_id`, `orderid`, `customerid`, `shipping\_address`, `shipping\_method`) VALUES ('7', '7', '7', 'Osaka', 'cargo ship');

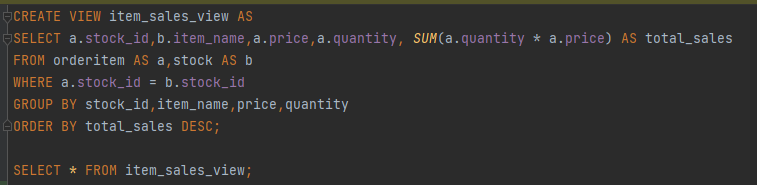
INSERT INTO `storemanagement`.`shipping` (`shipping\_id`, `orderid`, `customerid`, `shipping\_address`, `shipping\_method`) VALUES ('8', '8', '8', 'Tokyo', 'plane');

INSERT INTO `storemanagement`.`shipping` (`shipping\_id`, `orderid`, `customerid`, `shipping\_address`, `shipping\_method`) VALUES ('9', '9', '9', 'Melbroune', 'cargo ship');

INSERT INTO `storemanagement`.`shipping` (`shipping\_id`, `orderid`, `customerid`, `shipping\_address`, `shipping\_method`) VALUES ('10', '10', '10', 'Nonthaburi', 'car')**Data Manipulation Language (DML)**

### **4.5.3 Views**

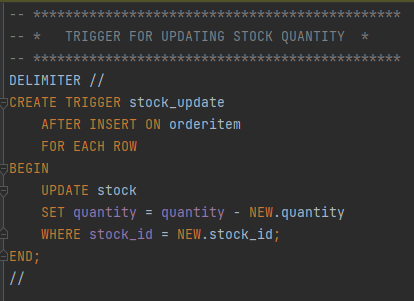
* Create ‘item\_sales\_view’ as an virtual table to calculate total\_sales and display it on the sale page.



### **4.5.4 Triggers**

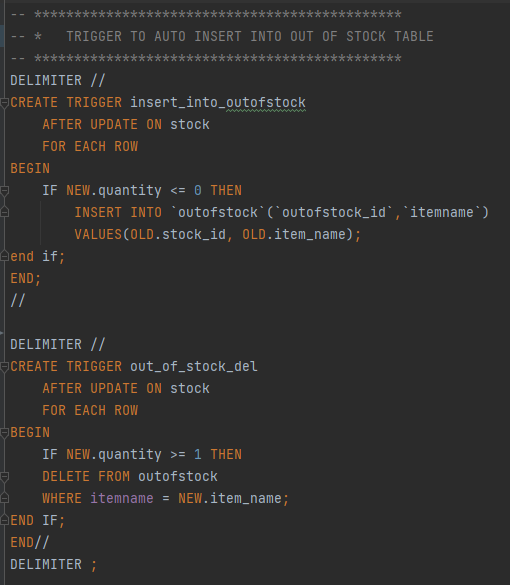
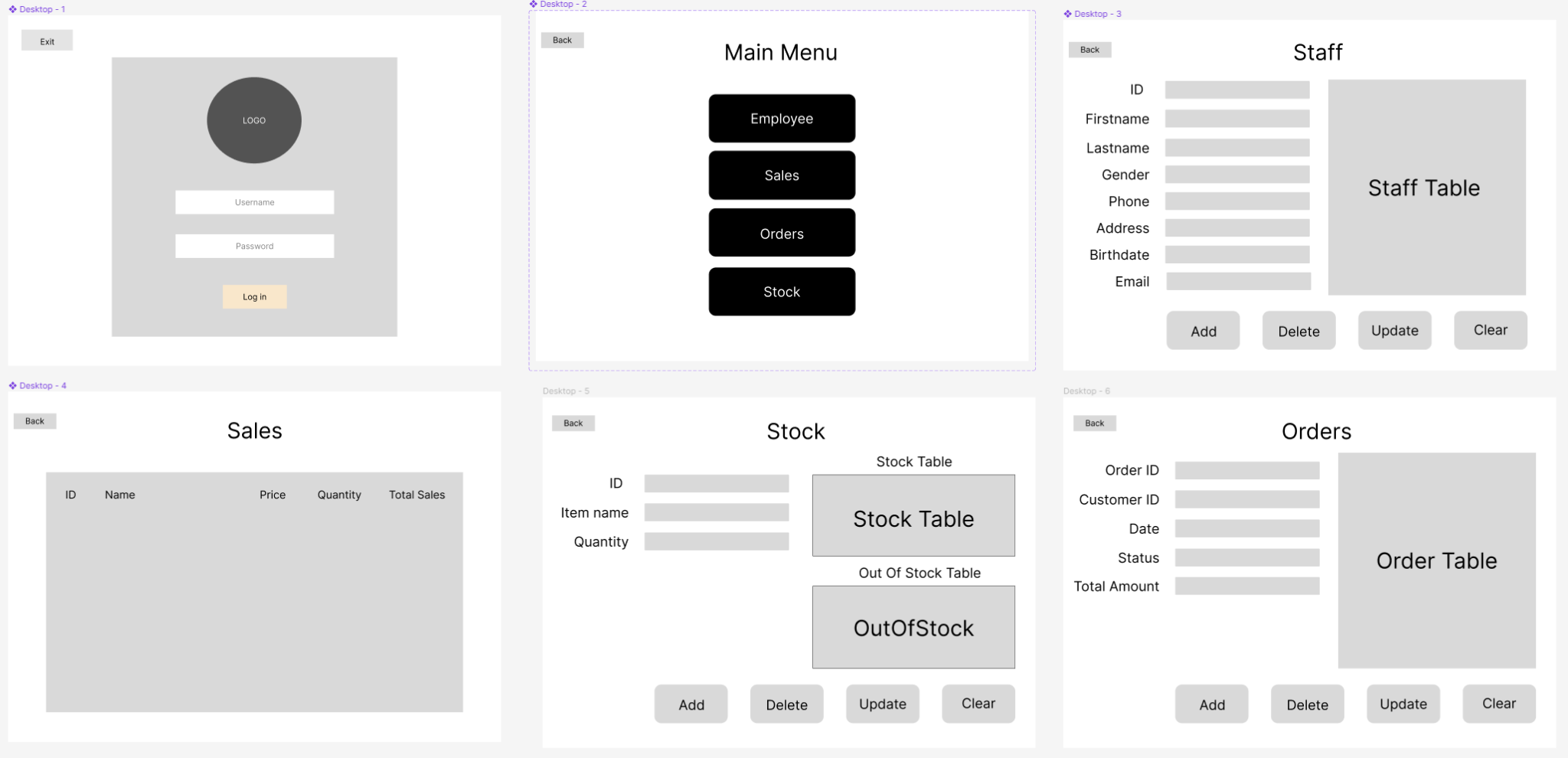
1. Trigger for updating the stock quantity.

* After each insert on the orderitem table , the quantity of the stock will be updated.
* The new stock quantity = stock quantity - orderitem quantity where stock.id is the same.

****

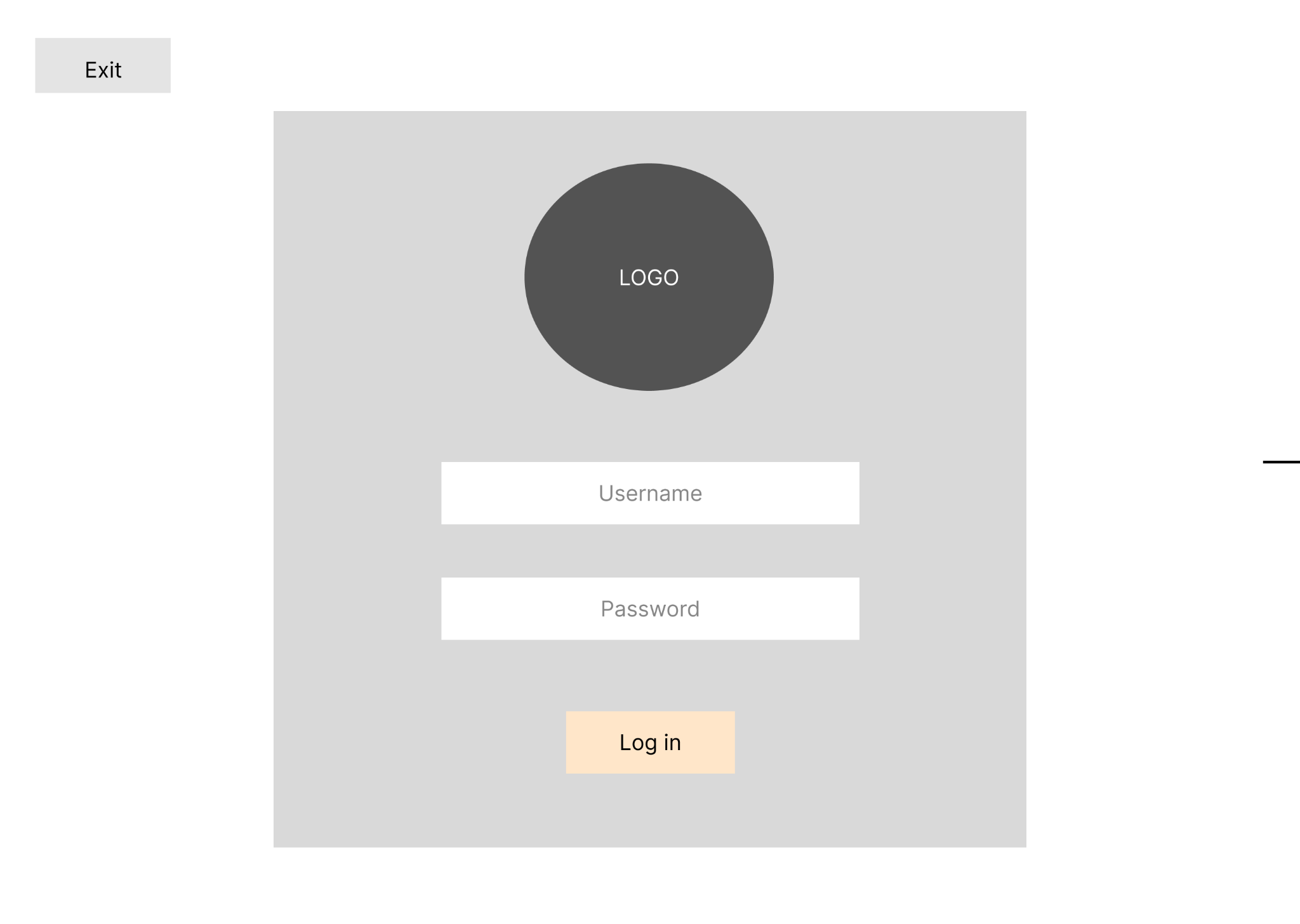
2. Trigger for updating the out of stock table.

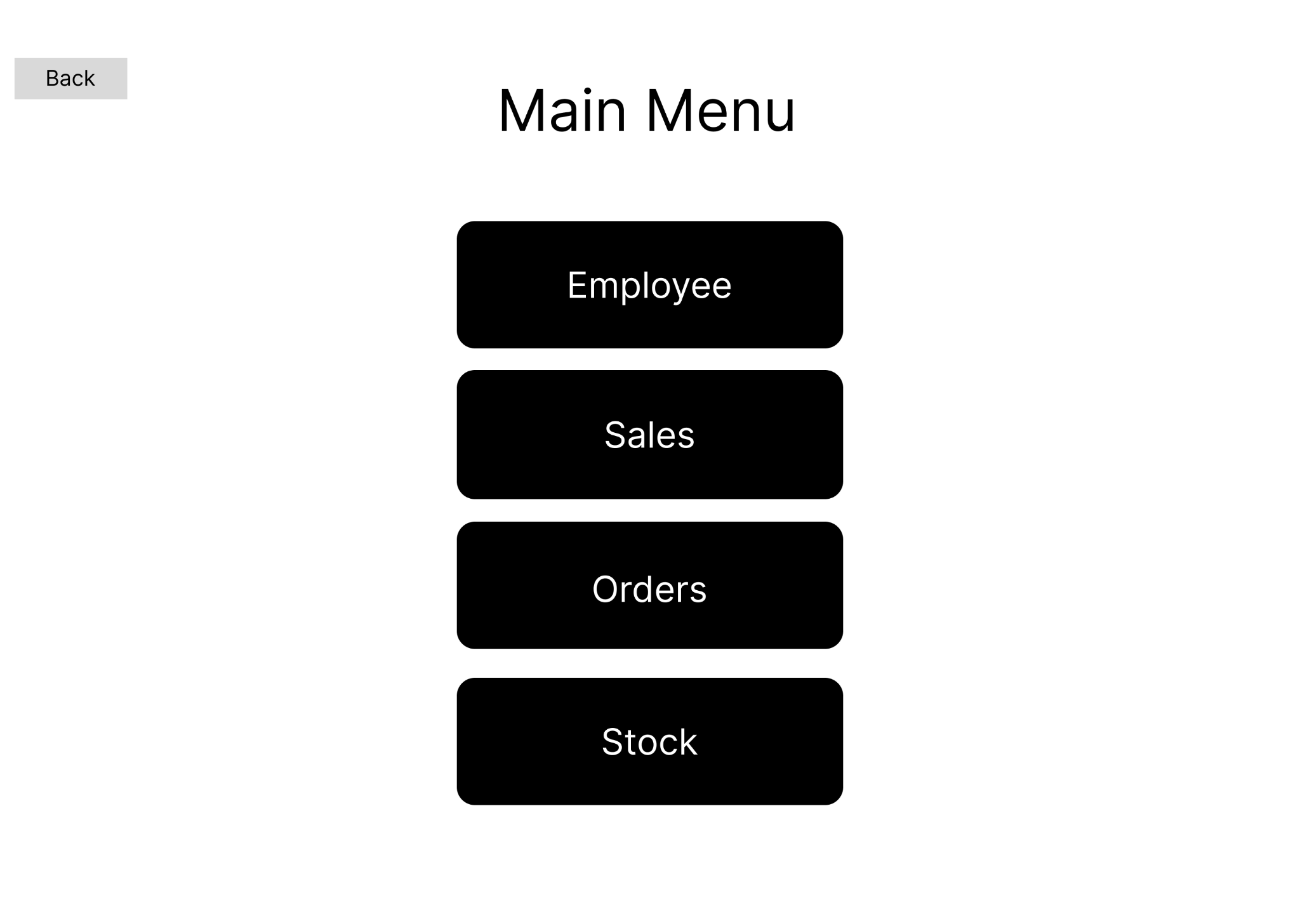
* After each update on the stock table
* if quantity is <= 0 then the item will be inserted into the out of stock table.
* if the quantity is >=0 then that item will be removed from the out of stock table and will be inserted back into the stock table.

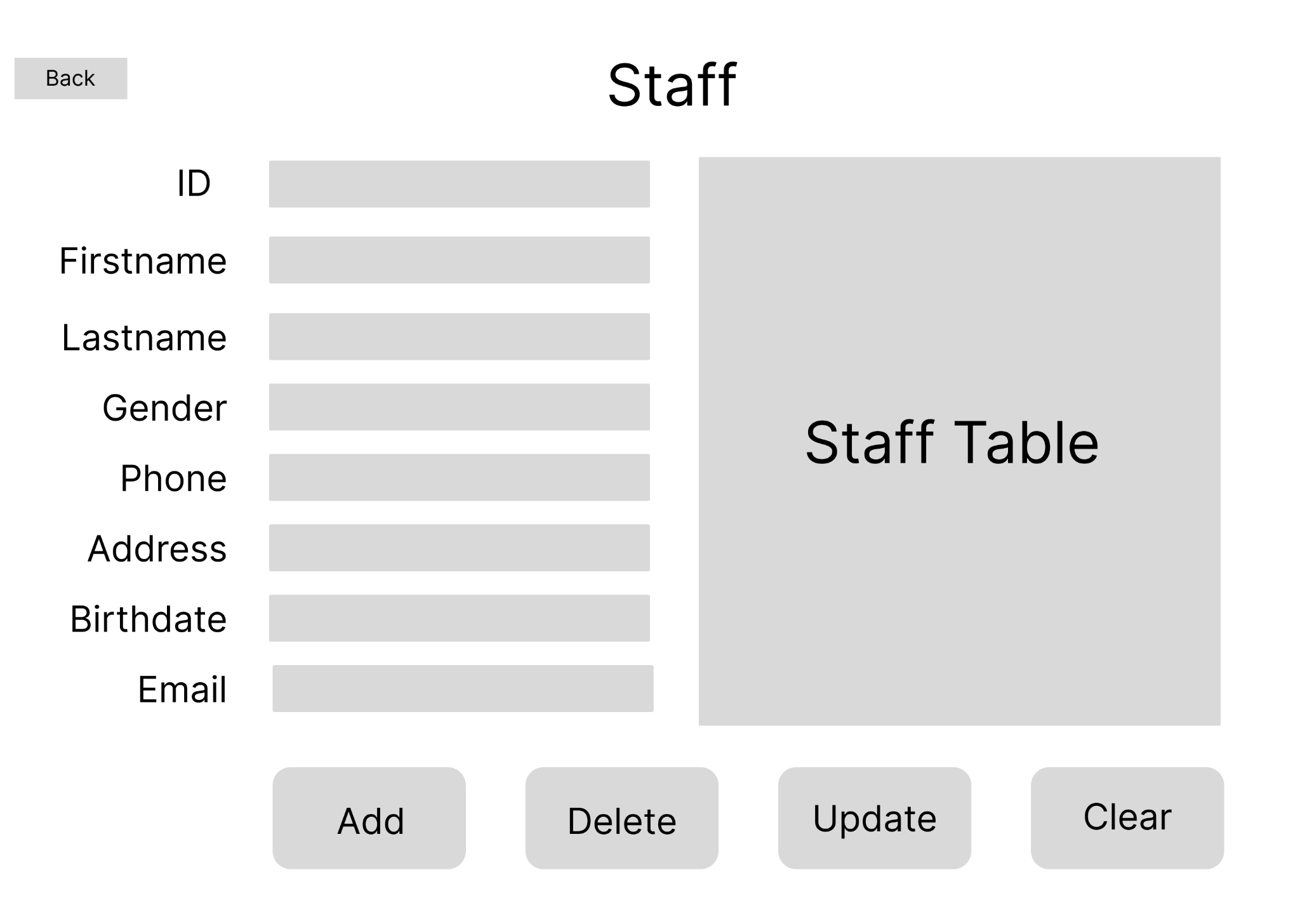
****

## **5. Application Design**

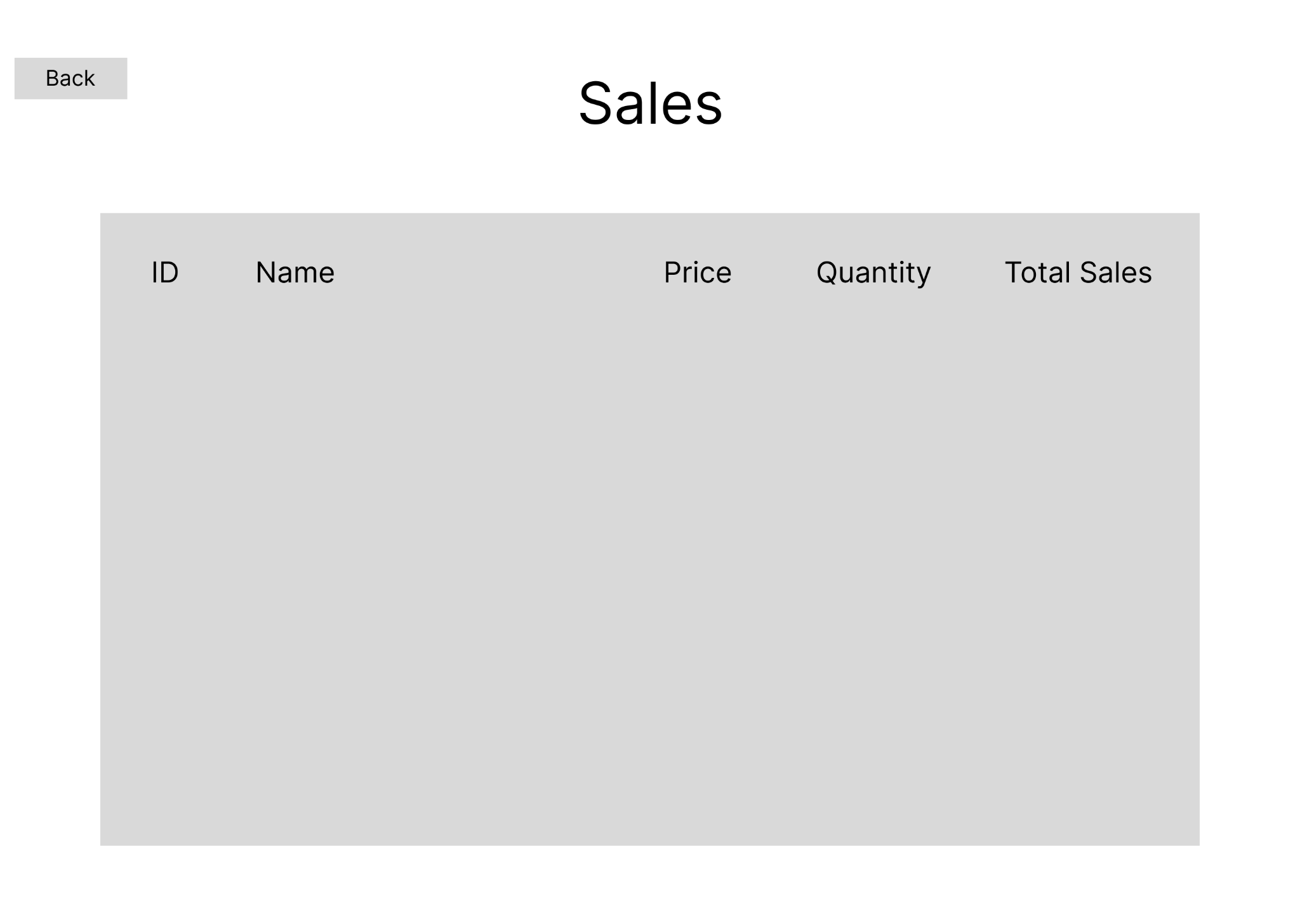
### **5.1 Screen Design**

**5.1.1 Login Page**

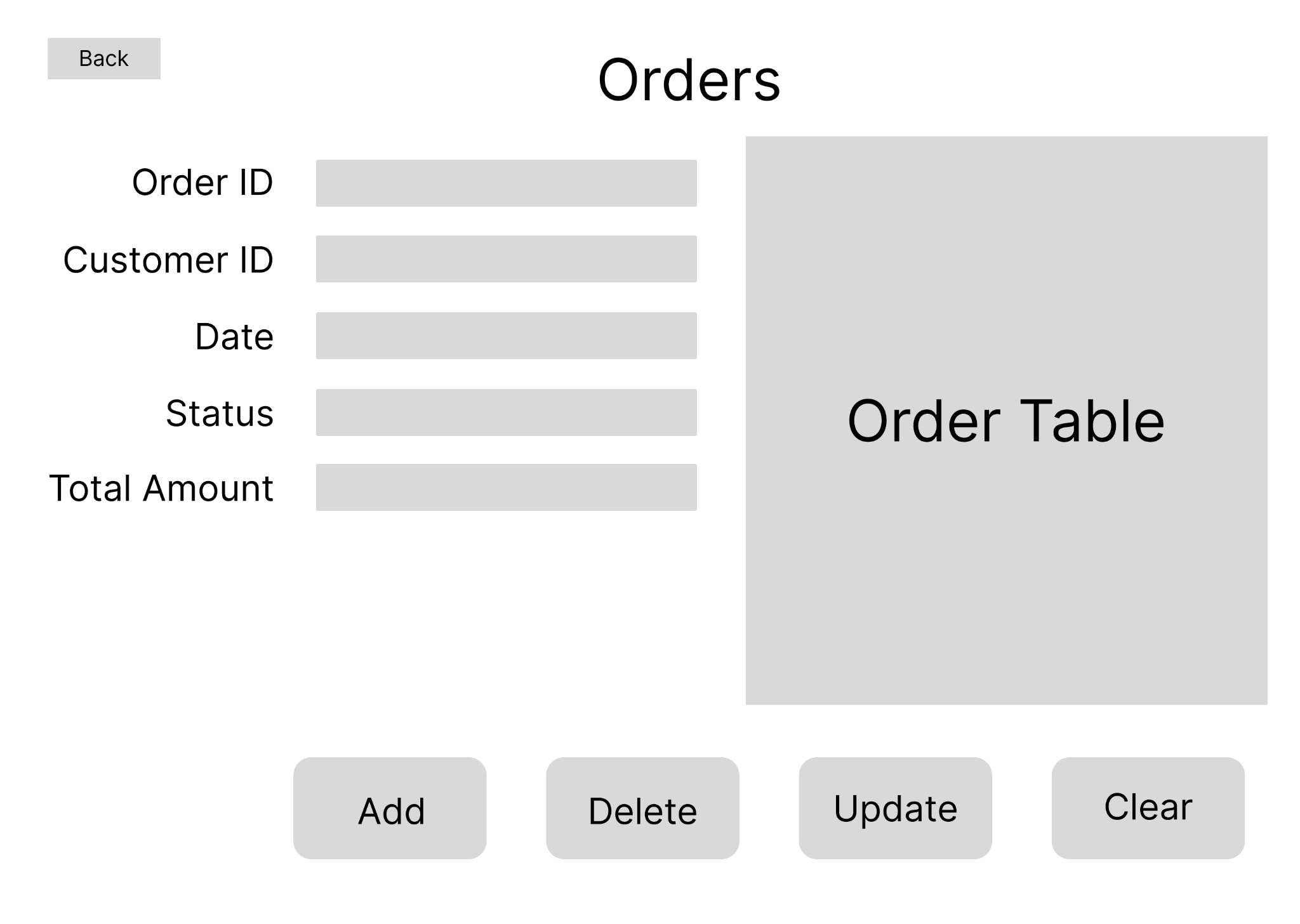
**5.1.2 Main Menu Page**



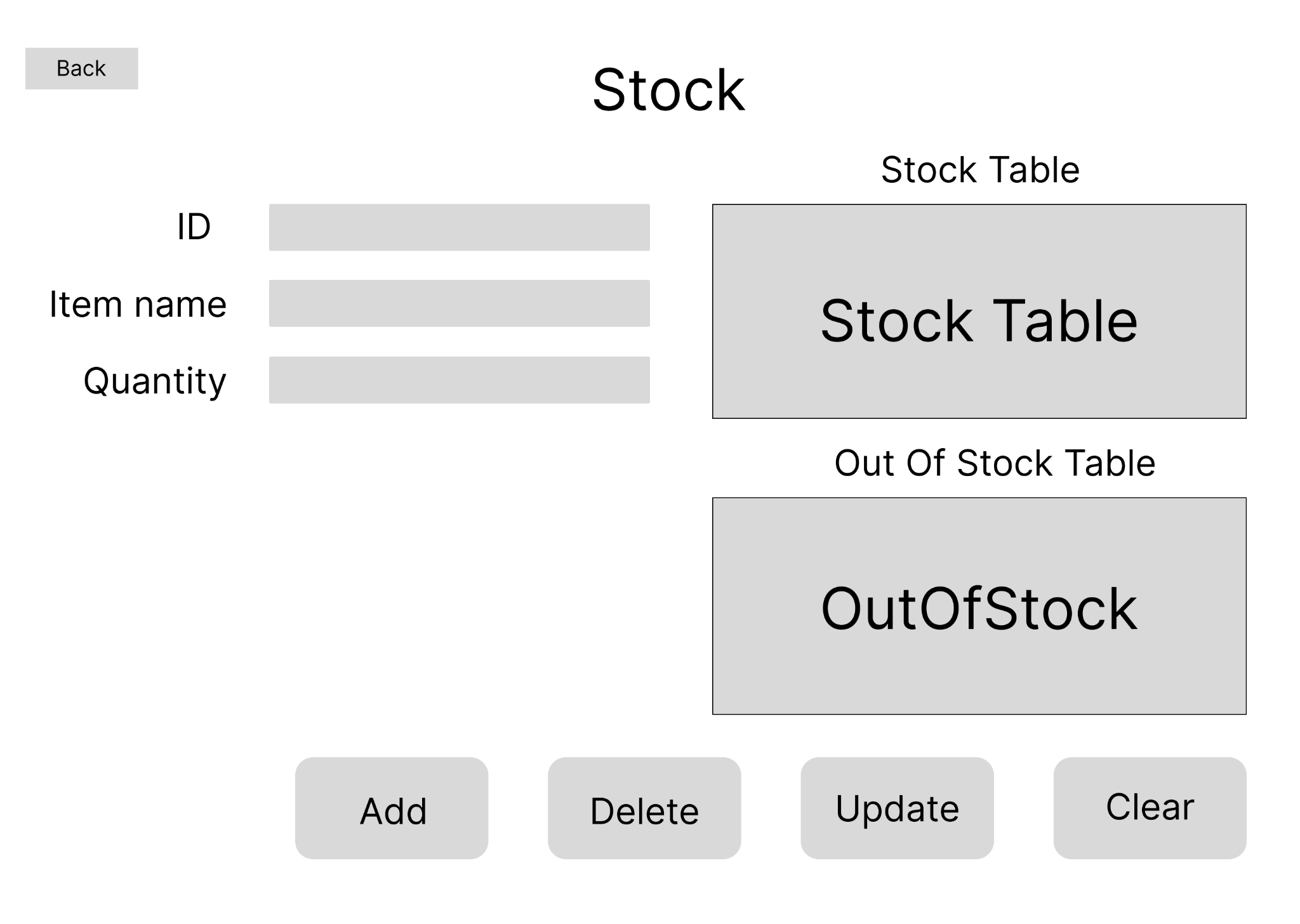
**5.1.3 Staff Page**



**5.1.4 Sales Page**

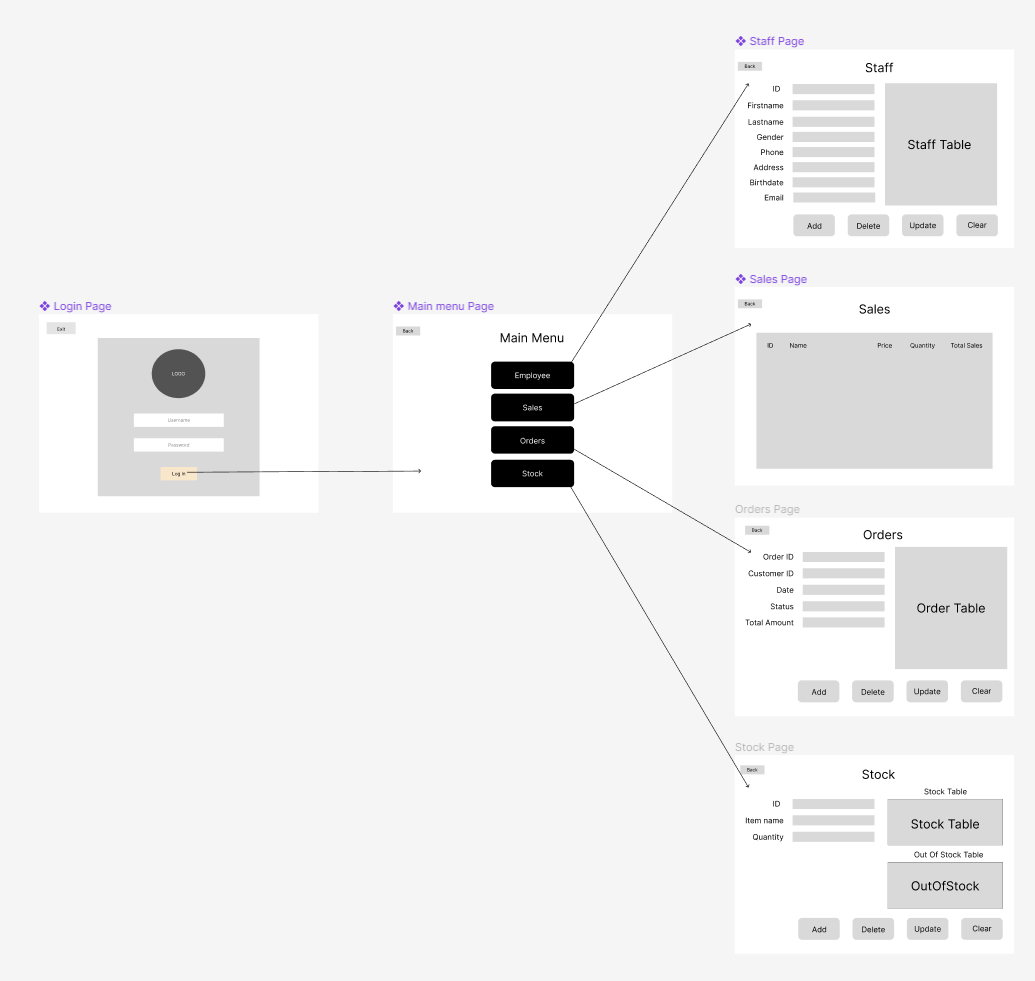


**5.1.5 Orders Page**



**5.1.6 Stock Page**

### **5.2 Mapping Fields**



### 

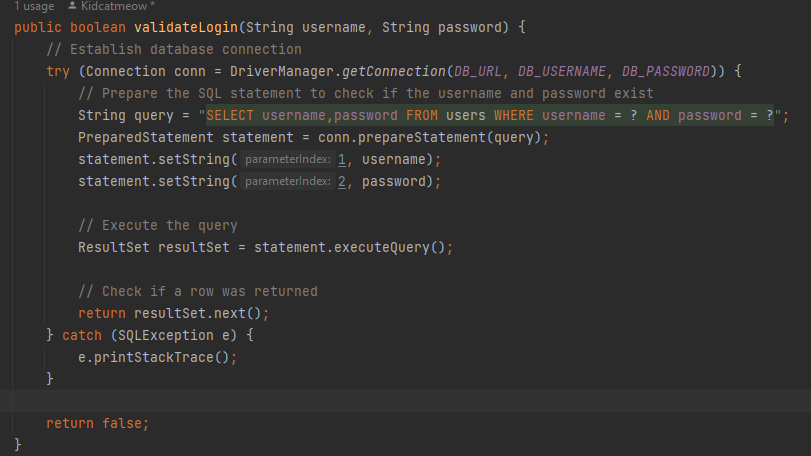
### 

### **5.3 Structured Query Language (SQL)**

### **5.3.1 Data Query Language (DQL)**

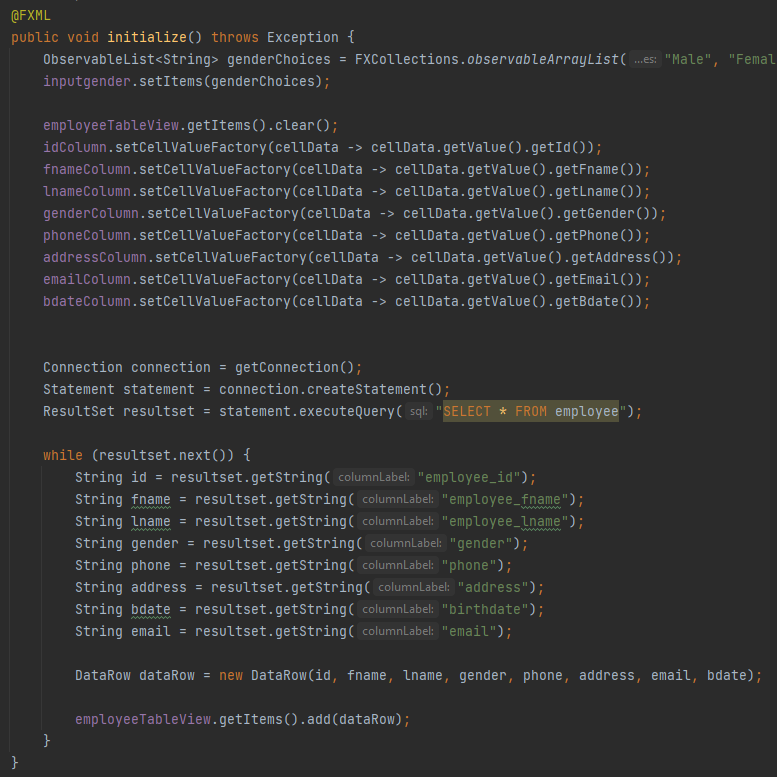
1. Login page

In order to validate the log in, we retrieve the username and password from the users table in the database to check if the input matches with the username/password within the database or not.



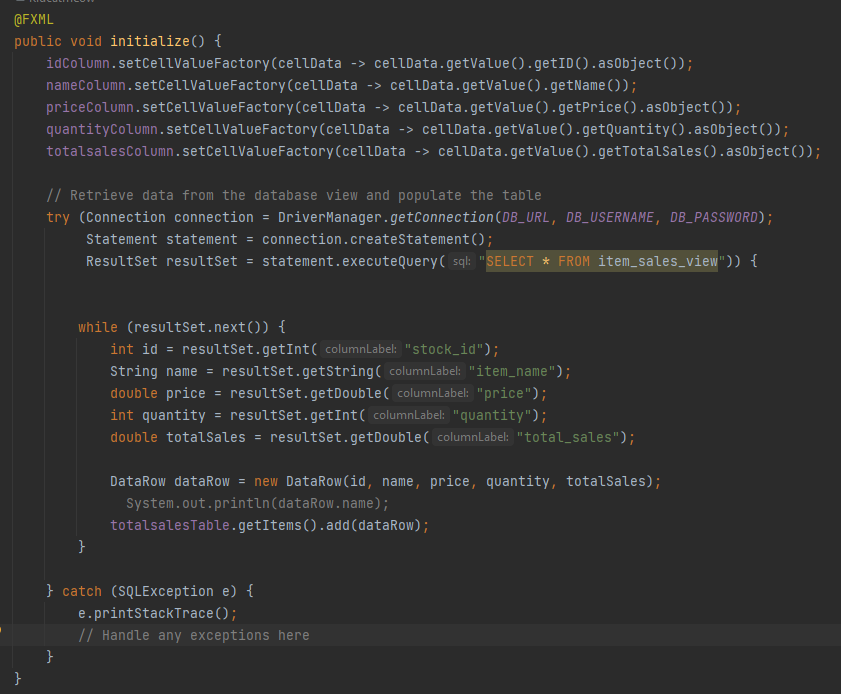
1. Staff Page

The initialize function in the staff page uses a DQL query to retrieve all data columns and rows from the employee table to display on the page.



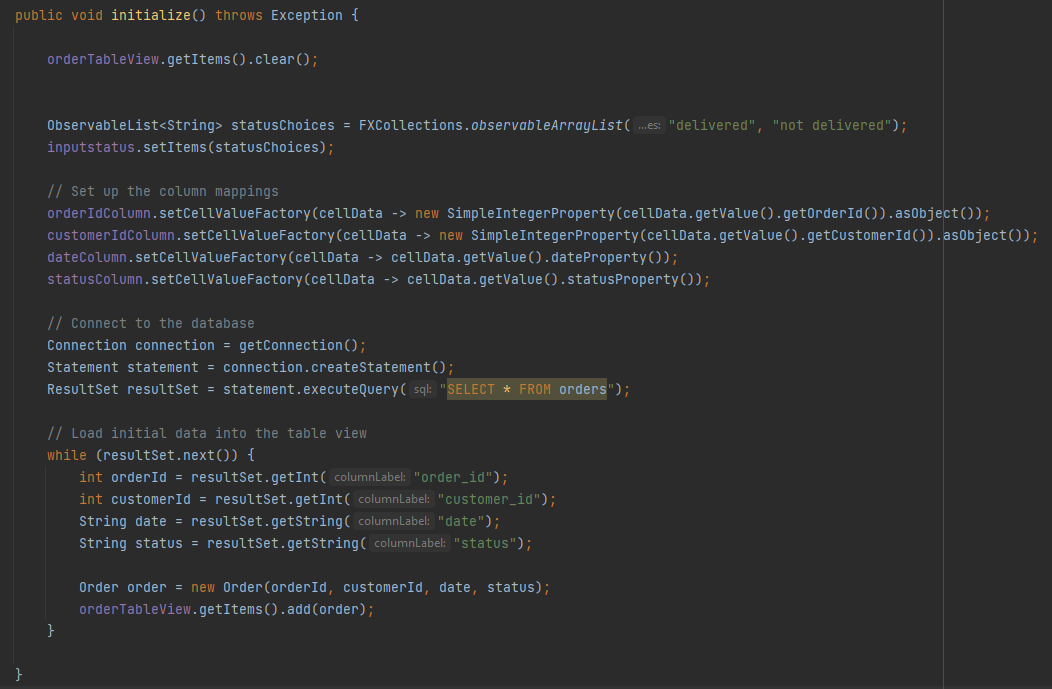
1. Sales Page

The initialize function in the Sales page uses a DQL query to retrieve all data columns and rows from the item\_sales\_view to display on the page.



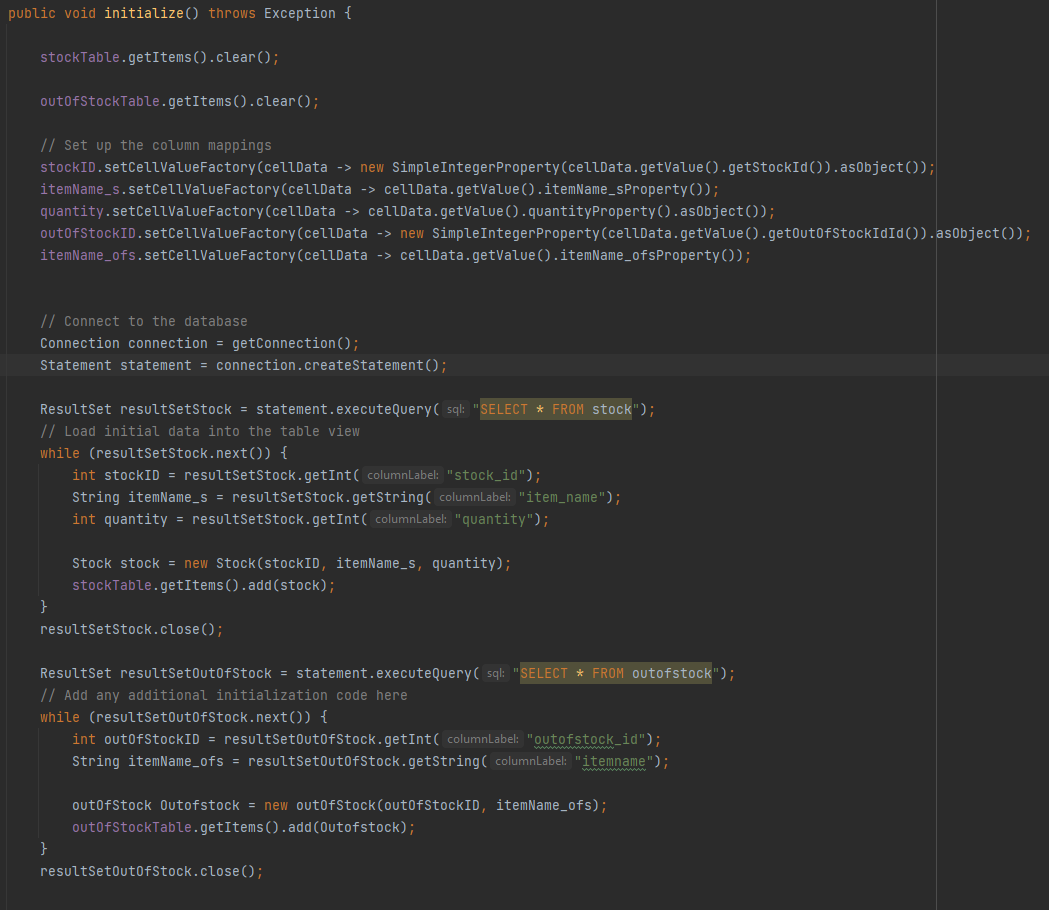
1. Orders Page

The initialize function in Orders page use DQL query to retrieve all data columns and rows from the orders table to display it on the page.



1. Stock Page

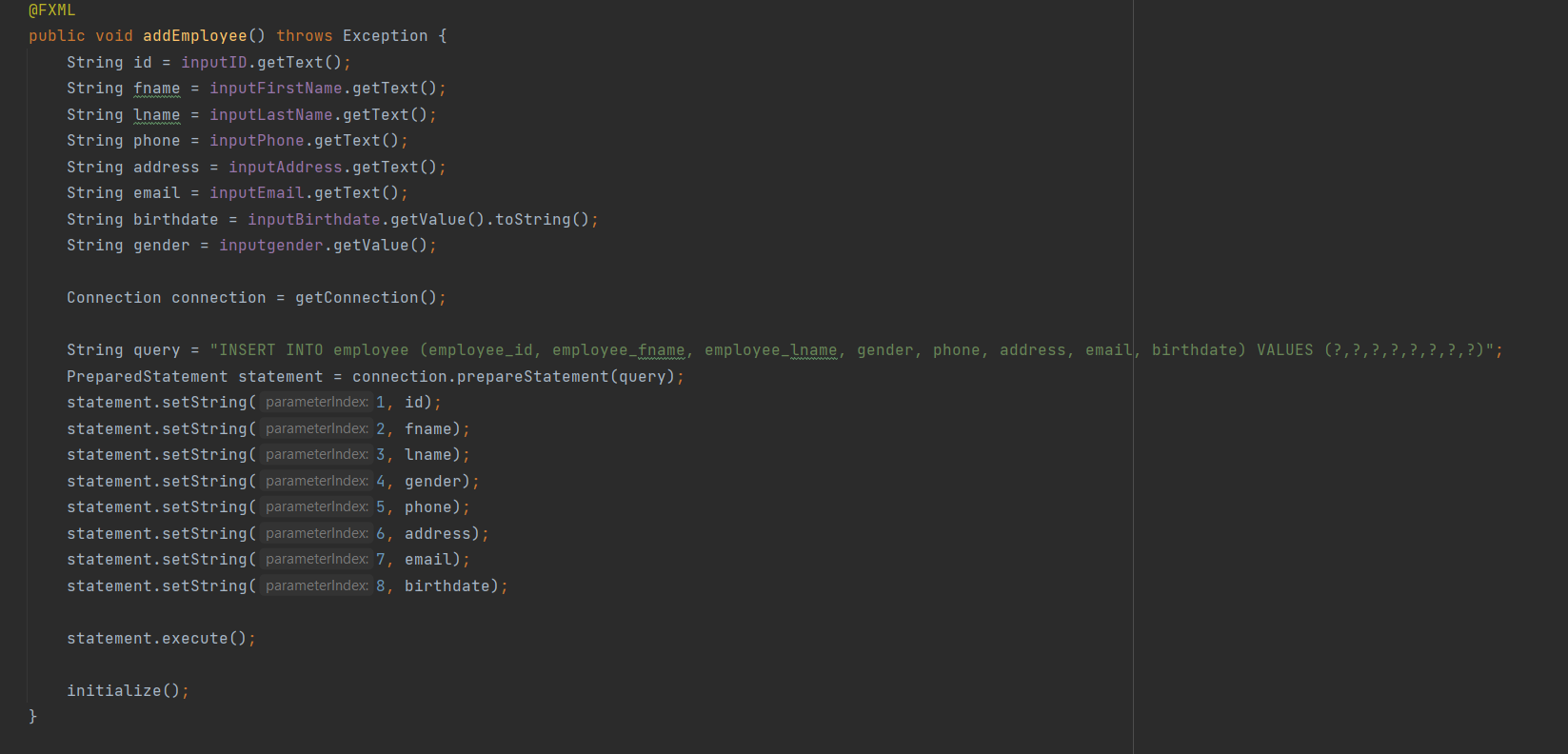
The initialize function in Orders page uses DQL query to retrieve all data columns and rows from the stocks and outofstock table to display it on the page.



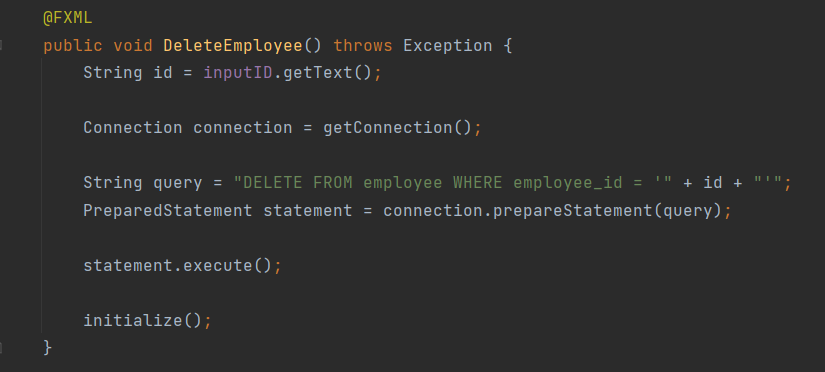
### **5.3.2 Data Manipulation Language (DML)**

1. Staff page (Add,Delete,update)

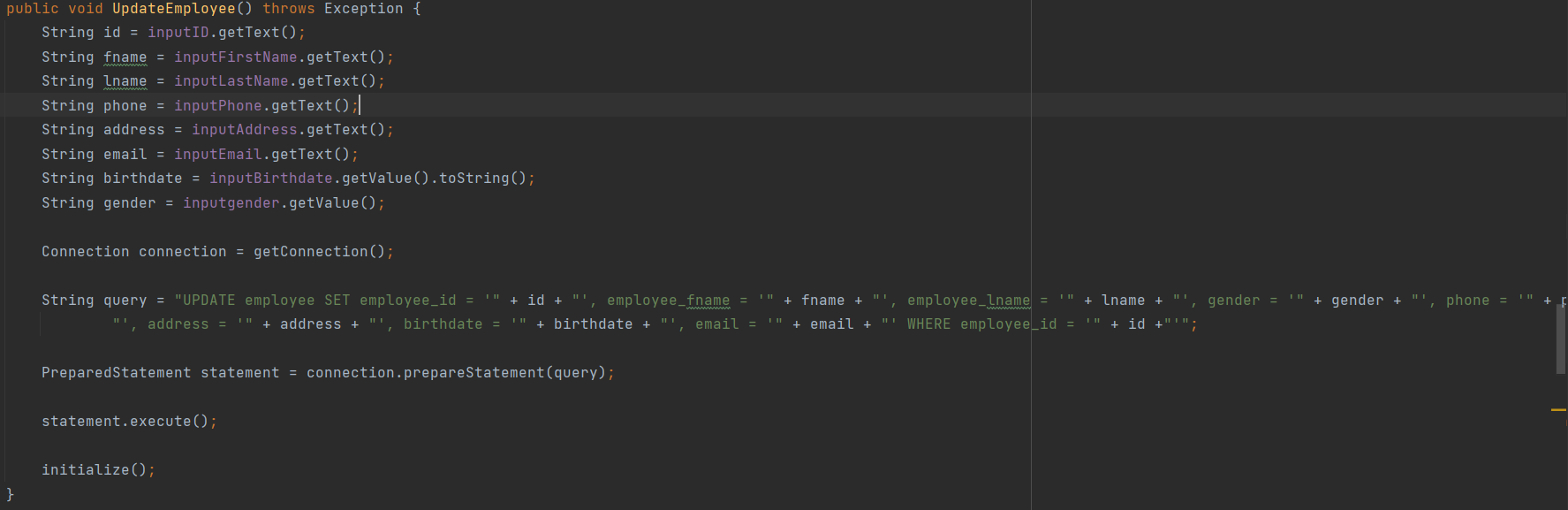
1.1 Add button



1.2 Delete Button



1.3 Update Button

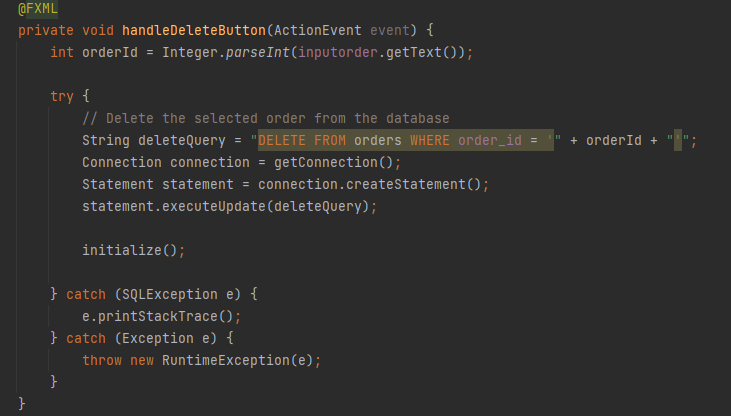


1. Orders page (Add,Delete,Update)

2.1 Add button



2.2 Delete Button



2.3 Update Button

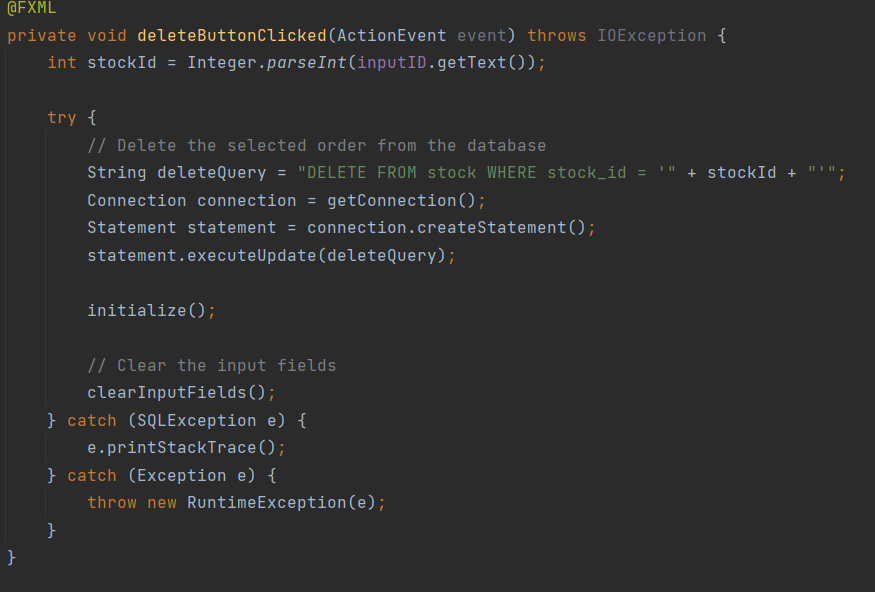


1. Stock Page (Add, Delete, Update)

3.1 Add



3.2 Delete



3.3 update



### **5.3.3 Front-end programming**

The code of the application can be viewed here : <https://github.com/Kidcatmeow/store_management>

## **6. Roles**

## Watsamon Phongwanit: Figma,Docker(Database schema,triggers,view),Log-in page, Main menu page (fxml , Java) ,Sales page (fxml , java) , Orders and Staff page (java) , report , powerpoint.

## Karnsiree Karnka: Figma, Stock and out of stock management page (Java), Scene Builder → Stock and out of stock management page (fxml file), ER diagram, Relational schema, data dictionary, PowerPoint, Report.

## Krittawit Suriyawong: Scene Builder → Orders page (fxml file) , Report , Powerpoint , Data Dictionary.

## Phatchathakorn: Scene Builder → Staff page (fxml file) , Powerpoint , Database insert queries , Data Dictionary.