Winter Semester 2022

CSE 250 Database Management System

Project Title: Supermarket Management System

Group Members:

Name	Roll Number	Program
Prasham Mehta	AU2040102	BTech CSE
Jay Shapariya	AU2040208	BTech CSE
Nitya Panchal	AU2040249	BTech CSE

Description Of The Project:

Our project title is **Supermarket Management System.**

System requirement specifications:

1. Front End: Our frontend is a website which is a mixture of **Html** and **PhP**.
2. Back End: We have simply used the MySQL and store database on **phpMyAdmin** tool.
3. To connect the frontend and the backend we use the tool name **XAMPP** server.

So, our first web page is a login page for the websites where the user has to type the username and password for entering the website. After entering in the website we will see the home page of the site which will include options like bill for going to page which deal with bill processes of the organization, stock and financial management which deals with the finance of the organization analytics which deals with the analysis of the organization's past purchase and sales report then comes the option of managing account which provides the user to manage the data of the users using these websites like the employee of the organization then last second options which allows us to update the details of the organization but this has no relevance with dbms. The last option is for logout. There is one more option for uploading the profile of the employee. If we talk about every option, the individual bill page is for billing for the customer, it contains fields to decide the product, quantity, discount if applicable and then prepare the bill. After the bill option all remaining options are the organization people.

ER Diagram:

ER - DIAGRAM

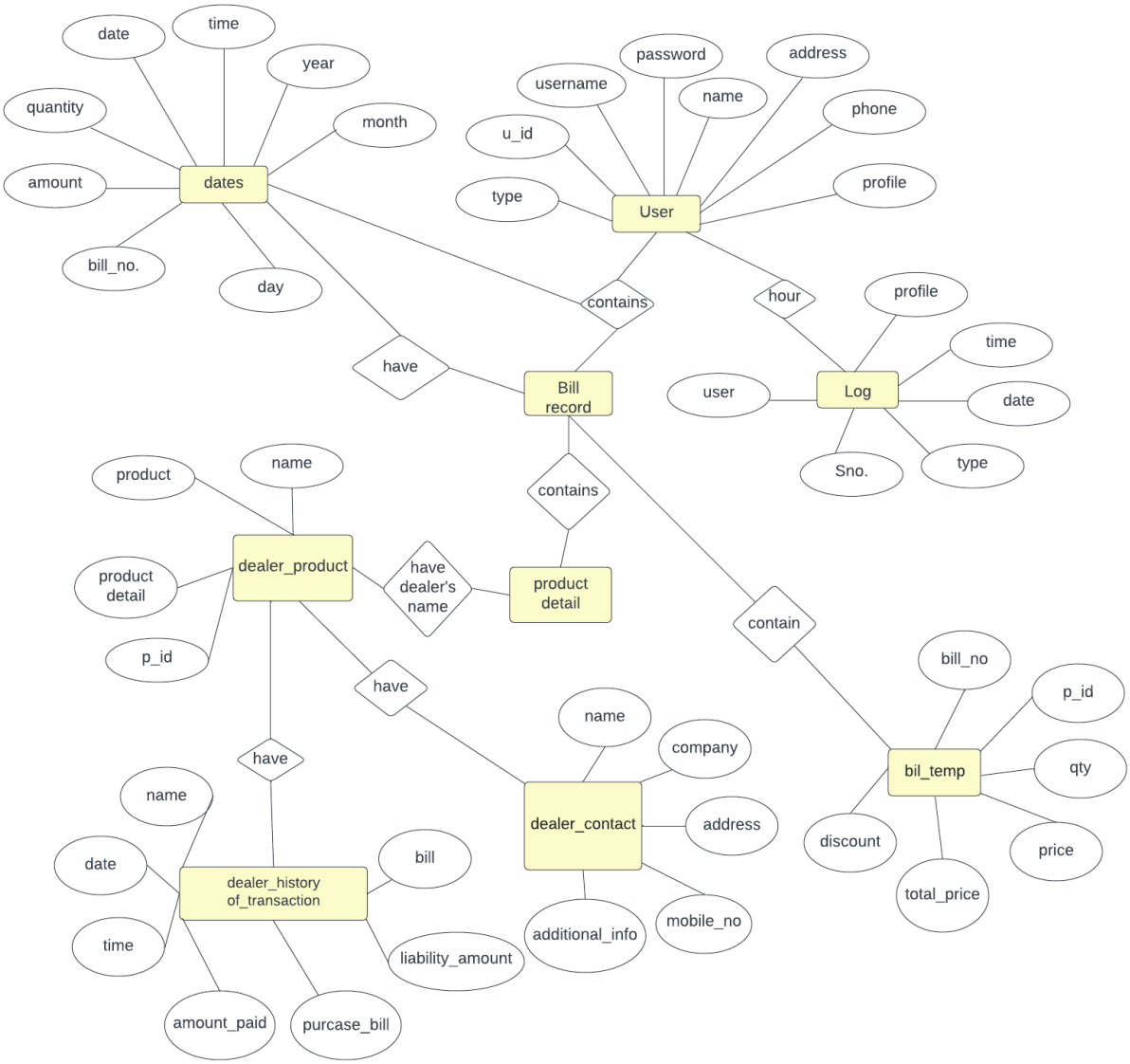
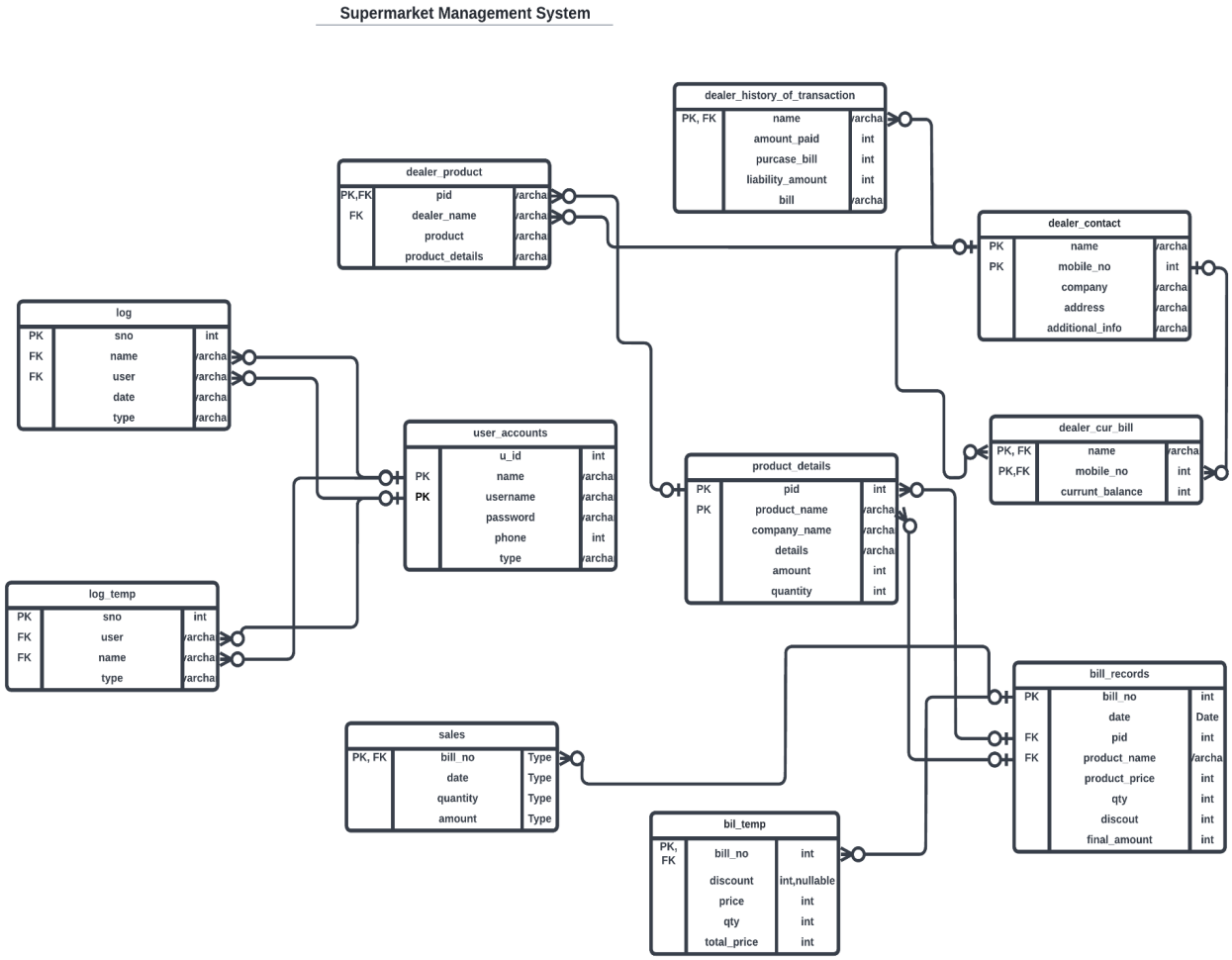


Table Design:



Statements Of CreateTable:

1)

```
DROP TABLE IF EXISTS `bill_records`;
```

```
CREATE TABLE IF NOT EXISTS `bill_records` (
  `bill_no` int NOT NULL AUTO_INCREMENT,
  `Date` date NOT NULL,
  `pid` int NOT NULL,
  `product_name` varchar(3000) CHARACTER SET latin1 COLLATE
latin1_swedish_ci NOT NULL,
  `product_price` int NOT NULL,
  `qty` int NOT NULL,
  `discout` int NOT NULL,
  `final_amount` int NOT NULL,
  PRIMARY KEY (`bill_no`)
) ENGINE=InnoDB AUTO_INCREMENT=66 DEFAULT CHARSET=latin1;
```

2)

```
DROP TABLE IF EXISTS `bil_temp`;
CREATE TABLE IF NOT EXISTS `bil_temp` (
  `bill_no` int NOT NULL,
  `qty` int NOT NULL,
  `price` int NOT NULL,
  `total_price` int NOT NULL,
  `discount` int DEFAULT NULL,
  PRIMARY KEY (`bill_no`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

3)

```
DROP TABLE IF EXISTS `dealer_contact`;
CREATE TABLE IF NOT EXISTS `dealer_contact` (
  `name` varchar(30) CHARACTER SET latin1 COLLATE latin1_swedish_ci
NOT NULL,
  `mobile_no` int NOT NULL,
```

```
  `company` varchar(50) CHARACTER SET latin1 COLLATE
latin1_swedish_ci NOT NULL,
  `address` varchar(75) CHARACTER SET latin1 COLLATE
latin1_swedish_ci NOT NULL,
  `additional_info` varchar(50) NOT NULL,
  PRIMARY KEY (`name`,`mobile_no`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

4)

```
DROP TABLE IF EXISTS `dealer_cur_bal`;
CREATE TABLE IF NOT EXISTS `dealer_cur_bal` (
  `name` varchar(30) NOT NULL,
  `mobile_no` int NOT NULL,
  `current_balance` int NOT NULL,
  PRIMARY KEY (`name`,`mobile_no`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

5)

```
DROP TABLE IF EXISTS `dealer_history_of_transaction`;
CREATE TABLE IF NOT EXISTS `dealer_history_of_transaction` (
  `name` varchar(30) NOT NULL,
  `amount_paid` int NOT NULL,
  `purchase_bill` int NOT NULL,
  `liability_amount` int NOT NULL,
  `bill` varchar(50) NOT NULL,
  PRIMARY KEY (`name`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

6)

```
DROP TABLE IF EXISTS `dealer_product`;
CREATE TABLE IF NOT EXISTS `dealer_product` (
  `pid` int NOT NULL,
  `dealer_n` text NOT NULL,
  `product` varchar(30) NOT NULL,
  `product_details` varchar(50) NOT NULL,
  PRIMARY KEY (`pid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

7)

```
DROP TABLE IF EXISTS `log`;
CREATE TABLE IF NOT EXISTS `log` (
  `sno` int NOT NULL AUTO_INCREMENT,
  `user` varchar(25) NOT NULL,
  `name` varchar(50) CHARACTER SET latin1 COLLATE latin1_swedish_ci
NOT NULL,
  `date` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `type` varchar(25) NOT NULL,
  PRIMARY KEY (`sno`)
) ENGINE=InnoDB AUTO_INCREMENT=49 DEFAULT CHARSET=latin1;
```

8)

```
DROP TABLE IF EXISTS `log_temp`;
CREATE TABLE IF NOT EXISTS `log_temp` (
  `s_no` int NOT NULL,
  `name_t` varchar(30) NOT NULL,
  `user_t` varchar(30) NOT NULL,
  `type_t` varchar(30) NOT NULL,
  PRIMARY KEY (`s_no`)
```

```
) ENGINE=MyISAM DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
```

9)

```
DROP TABLE IF EXISTS `product_details`;
CREATE TABLE IF NOT EXISTS `product_details` (
  `pid` int NOT NULL AUTO_INCREMENT,
  `product_name` varchar(30) CHARACTER SET latin1 COLLATE
latin1_swedish_ci NOT NULL,
  `company_name` varchar(30) CHARACTER SET latin1 COLLATE
latin1_swedish_ci NOT NULL,
  `details` varchar(60) CHARACTER SET latin1 COLLATE
latin1_swedish_ci NOT NULL,
  `amount` int NOT NULL,
  `quantity` int NOT NULL,
  PRIMARY KEY (`pid`)
) ENGINE=InnoDB AUTO_INCREMENT=14 DEFAULT CHARSET=latin1;
```

10)

```
DROP TABLE IF EXISTS `sales`;
CREATE TABLE IF NOT EXISTS `sales` (
  `bill_no` int NOT NULL,
  `date` varchar(30) CHARACTER SET latin1 COLLATE latin1_swedish_ci
NOT NULL,
  `quantity` int NOT NULL,
  `amount` int NOT NULL,
  PRIMARY KEY (`bill_no`),
  KEY `bill_no` (`bill_no`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

11)

```
DROP TABLE IF EXISTS `user_accounts`;
CREATE TABLE IF NOT EXISTS `user_accounts` (
  `uid` int NOT NULL,
  `name` varchar(20) CHARACTER SET latin1 COLLATE latin1_swedish_ci
NOT NULL,
  `username` varchar(30) CHARACTER SET latin1 COLLATE
latin1_swedish_ci NOT NULL,
  `password` varchar(30) CHARACTER SET latin1 COLLATE
latin1_swedish_ci NOT NULL,
  `phone` int NOT NULL,
  `type` varchar(25) NOT NULL,
  PRIMARY KEY (`name`,`username`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Statements Of Procedure:

```
DELIMITER $$
```

-- 1) Procedure to Insert new Product in the List:

```
DROP PROCEDURE IF EXISTS `add_new_product`$$

CREATE DEFINER=`root`@`localhost`

PROCEDURE `add_new_product` (IN `amt` INT, IN `barcode` INT, IN
`dname` VARCHAR(30), IN `detail` VARCHAR(30), IN `pname`
VARCHAR(30), IN `qty` INT, IN `pid` INT)
```

INSERT INTO product_details (amount, barcode, company_name, details, product_name,quantity, pid) VALUES (amt, barcode, dname, detail, pname, qty, pid)$$

-- 2) Procedure to make the Bill:

DROP PROCEDURE IF EXISTS `insert_bill`$$

CREATE DEFINER=`root`@`localhost`

PROCEDURE `insert_bill` (OUT `amt` INT, IN `quantity` INT, IN `pname` VARCHAR(30), IN `dis` INT)

BEGIN

SELECT amount INTO amt FROM product_details WHERE product_name = pname;

INSERT INTO bill_records (discout, final_amount, product_name, product_price, qty) VALUES
(dis,((amt*quantity)-(amt*quantity)*(dis/100)),pname, amt, quantity);

END$$

-- 3) Procedure to view the Login History:

DROP PROCEDURE IF EXISTS `login_history`$$

CREATE DEFINER=`root`@`localhost`

PROCEDURE `login_history` (IN `name_n` VARCHAR(30))

SELECT * FROM log WHERE name = name_n$$

-- 4) Procedure to find which product’s sale is high:

```
DROP PROCEDURE IF EXISTS `max_order`$$

CREATE DEFINER=`root`@`localhost`

PROCEDURE `max_order` ()

SELECT product_name FROM bill_records GROUP BY product_name
ORDER BY COUNT(*) DESC LIMIT 1$$
```

-- 5) Procedure to find which product’s stock is high:

```
DROP PROCEDURE IF EXISTS `max_stock`$$

CREATE DEFINER=`root`@`localhost`

PROCEDURE `max_stock` ()

SELECT * FROM product_details WHERE quantity = ( SELECT
MAX(quantity) FROM product_details )$$
```

-- 6)Procedure to find which product’s sales is low:

```
DROP PROCEDURE IF EXISTS `min_order`$$

CREATE DEFINER=`root`@`localhost`

PROCEDURE `min_order` ()

SELECT product_name FROM bill_records GROUP BY product_name
ORDER BY COUNT(*) LIMIT 1$$
```

-- 7) Procedure to find which product's stocks is low:

```
DROP PROCEDURE IF EXISTS `min_stocks`$$

CREATE DEFINER=`root`@`localhost`

PROCEDURE `min_stocks` ()

SELECT * FROM product_details WHERE quantity = ( SELECT
MIN(quantity) FROM product_details )$$
```

-- 8) Procedure to update the stocks and price of the product available from the firm:

```
DROP PROCEDURE IF EXISTS `update_product_detail`$$

CREATE DEFINER=`root`@`localhost`

PROCEDURE `update_product_detail` (IN `amt` INT, IN `stok` INT, IN `pid`
INT, IN `prod` VARCHAR(30), IN `dname` VARCHAR(30))

UPDATE product_details

SET product_details.amount=amt, product_details.quantity=stok

WHERE product_details.pid=pid AND product_details.product_name=prod
AND product_details.company_name=dname$$
```

-- 9) Procedure to view the bill:

```
DROP PROCEDURE IF EXISTS `veiw_bill`$$

CREATE DEFINER=`root`@`localhost`

PROCEDURE `veiw_bill` (IN `bill_no` INT)
```

```
Select * FROM bill_records

WHERE bill_records.bill_no = bill_no$$
```

-- 10) Procedure to view the products whose stocks is more than given number:

```
DROP PROCEDURE IF EXISTS `view_stock`$$

CREATE DEFINER=`root`@`localhost`

PROCEDURE `view_stock` (IN `stock` INT)

Select * FROM product_details WHERE product_details.quantity >=
stock$$

DELIMITER ;
```

Statements Of Trigger:

-- 1) If something from the bill record is deleted by mistake or intentionally then it will take backup in another table:

```
DROP TRIGGER IF EXISTS `delete_bill`;

DELIMITER $$

CREATE TRIGGER `delete_bill` BEFORE DELETE ON `bill_records` FOR EACH ROW INSERT INTO bil_temp VALUES (OLD.bill_no, OLD.discout,OLD.final_amount, OLD.qty, OLD.product_price) $$

DELIMITER ;
```

-- 2) After every selling of products and making bill it will take sale report in sales table:

```
DROP TRIGGER IF EXISTS `insert_sales`;

DELIMITER $$
```

```
CREATE TRIGGER `insert_sales` AFTER INSERT ON `bill_records` FOR EACH ROW INSERT INTO sales VALUES (new.bill_no, new.Date, new.qty,new.final_amount) $$

DELIMITER ;
```

-- 3) If something is bought then this trigger will subtract the stock bought by the customer from total stock available. In-short this will keep the stock of product uptodate.

```
DROP TRIGGER IF EXISTS `update_product_stock`;

DELIMITER $$

CREATE TRIGGER `update_product_stock` AFTER INSERT ON `bill_records` FOR EACH ROW UPDATE product_details SET quantity = (quantity-NEW.qty) WHERE product_name =NEW.product_name  $$

DELIMITER ;
```

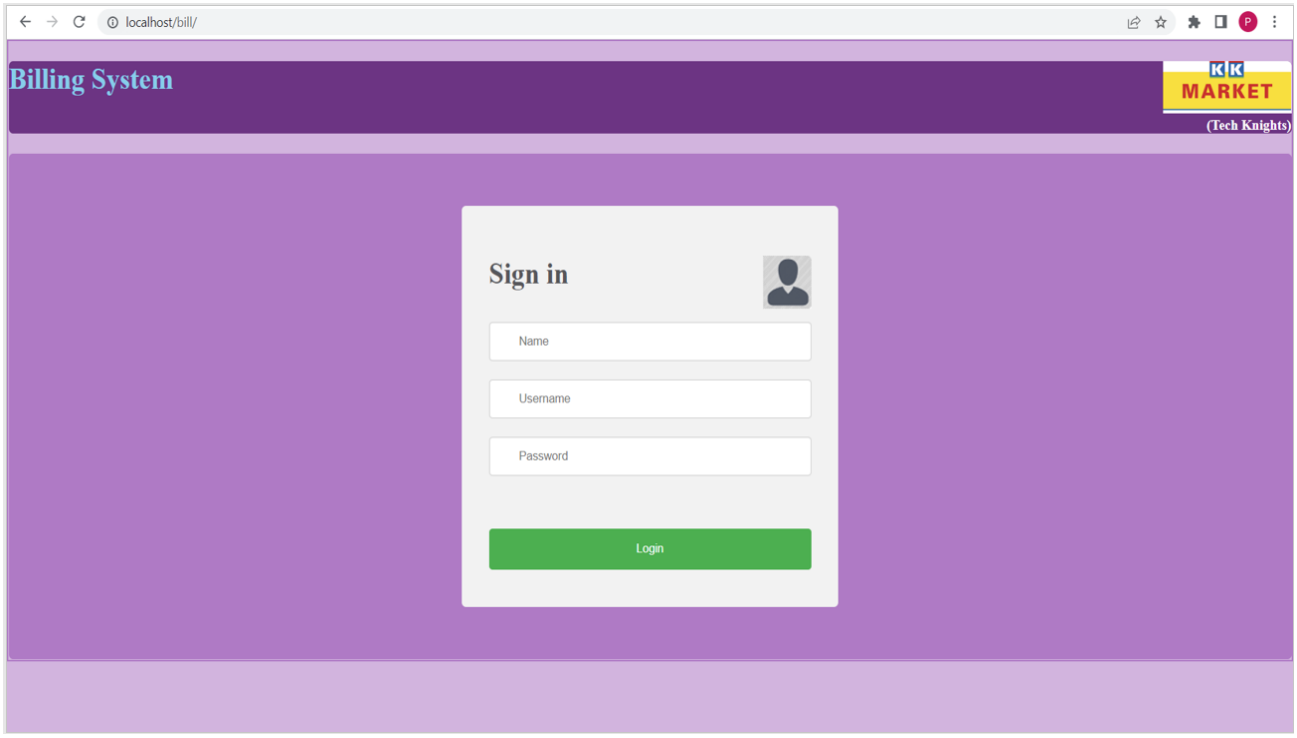
-- 4) If something from the user_log is deleted by mistake or intentionally then it will take backup in another table:

```
DROP TRIGGER IF EXISTS `log_delete`;

DELIMITER $$

CREATE TRIGGER `log_delete` BEFORE DELETE ON `log` FOR EACH ROW INSERT INTO log_temp VALUES (OLD.sno, OLD.name,OLD.user, OLD.type)  $$

DELIMITER ;
```

-- 5) It will check that before someone buys something then the stock of that available with the firm is not less than some value. If it is low then it will show error “Out OF Stock”:

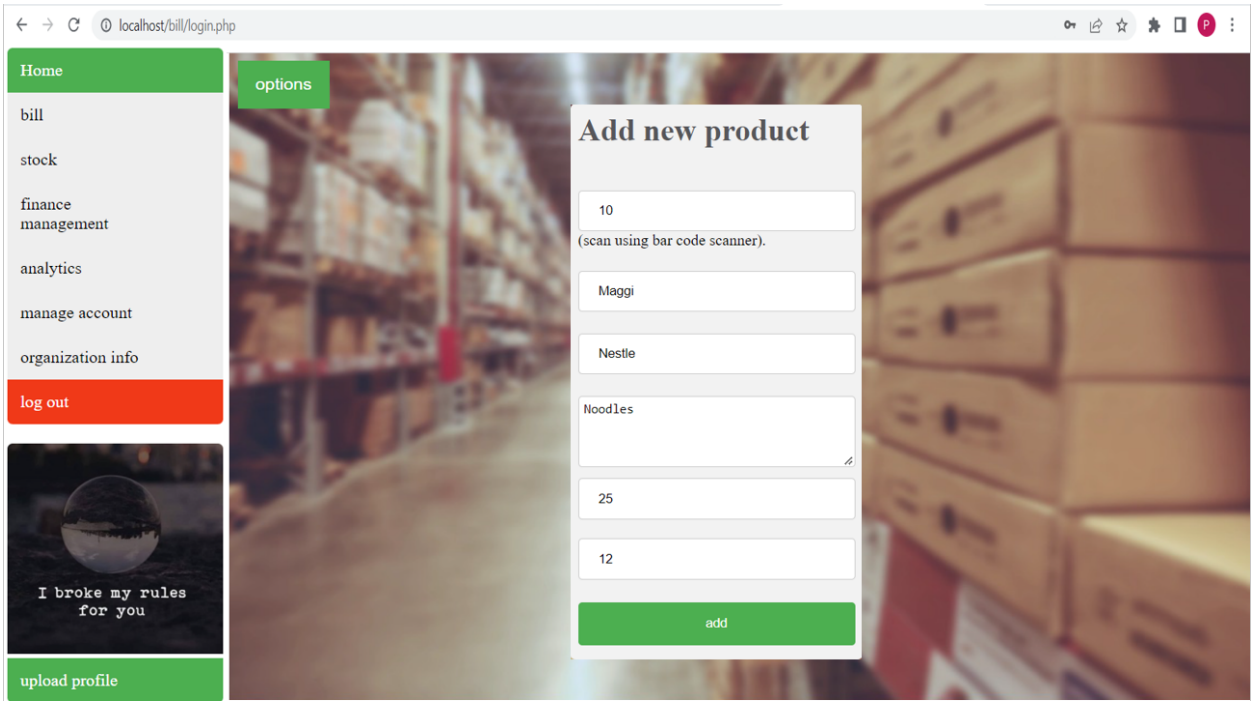
```
DROP TRIGGER IF EXISTS `min_stock`;

DELIMITER $$

CREATE TRIGGER `min_stock` BEFORE UPDATE ON `product_details`
FOR EACH ROW IF (product_details.quantity < '10') THEN

SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'Out Of Stock.';

END IF  $$

DELIMITER ;
```

1) Login Page:

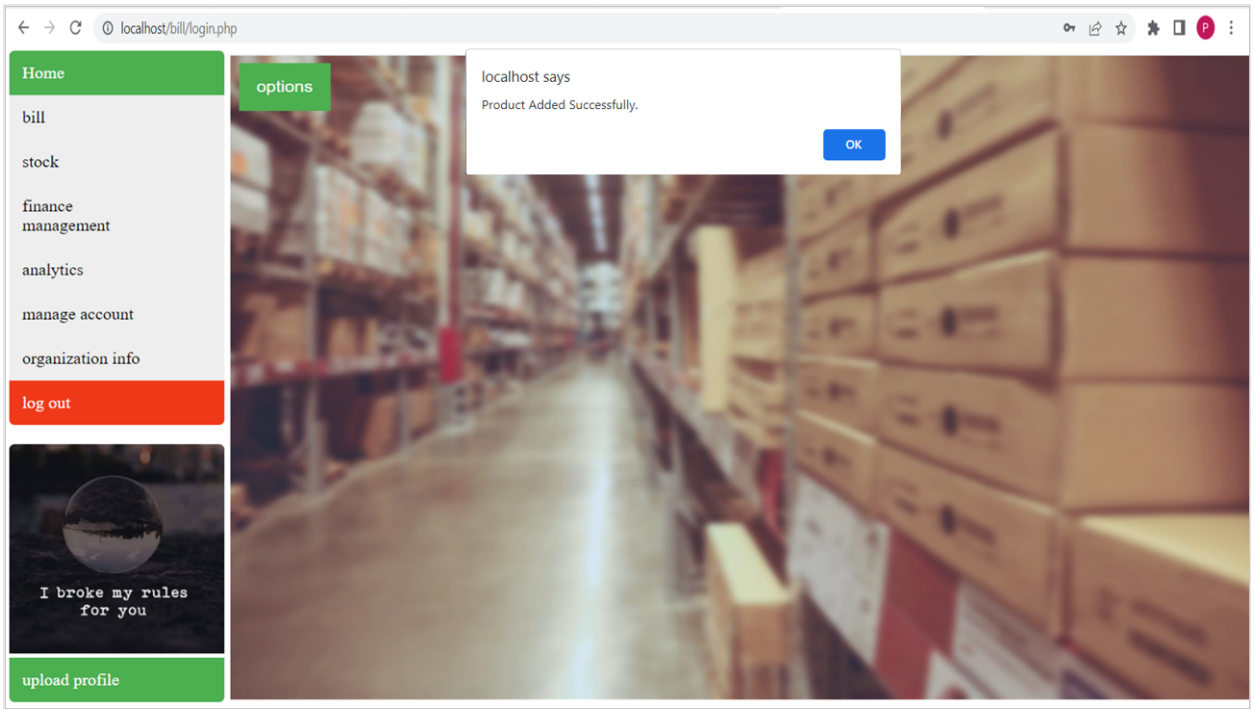


2) Home Page:

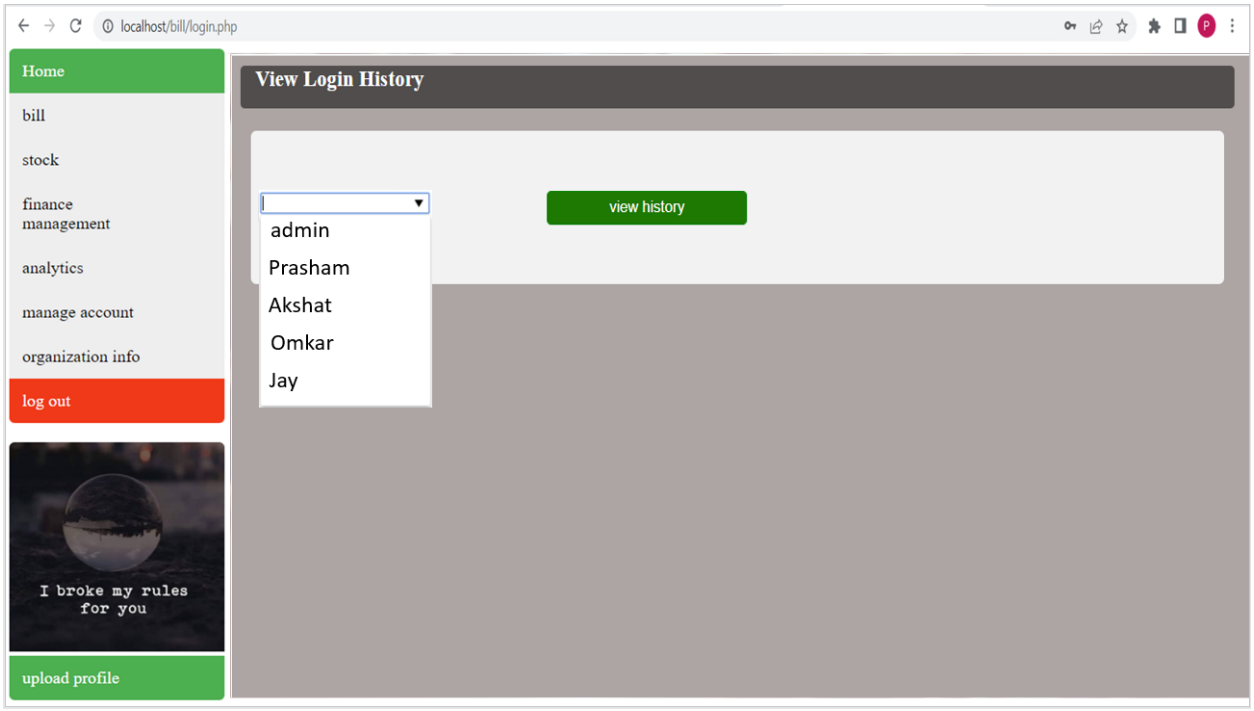
3) For adding new product details. Here “add_new_product” is use :



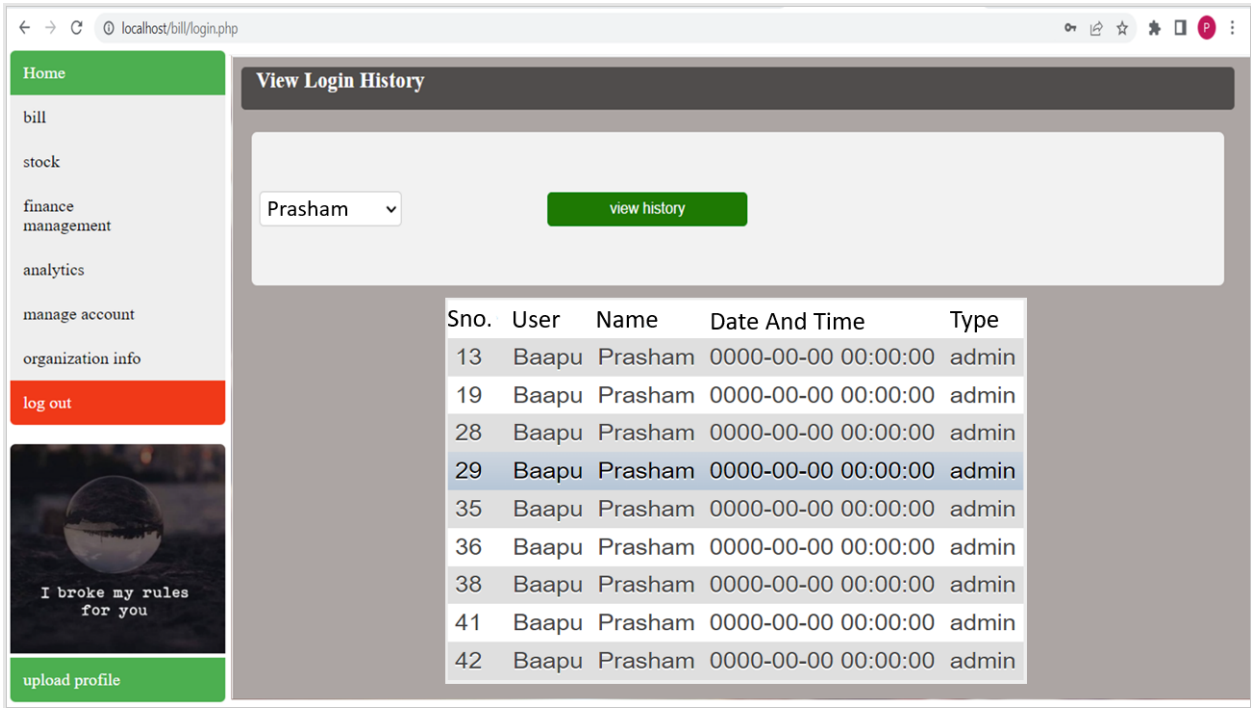
4) Result of “add_new_product” procedure :



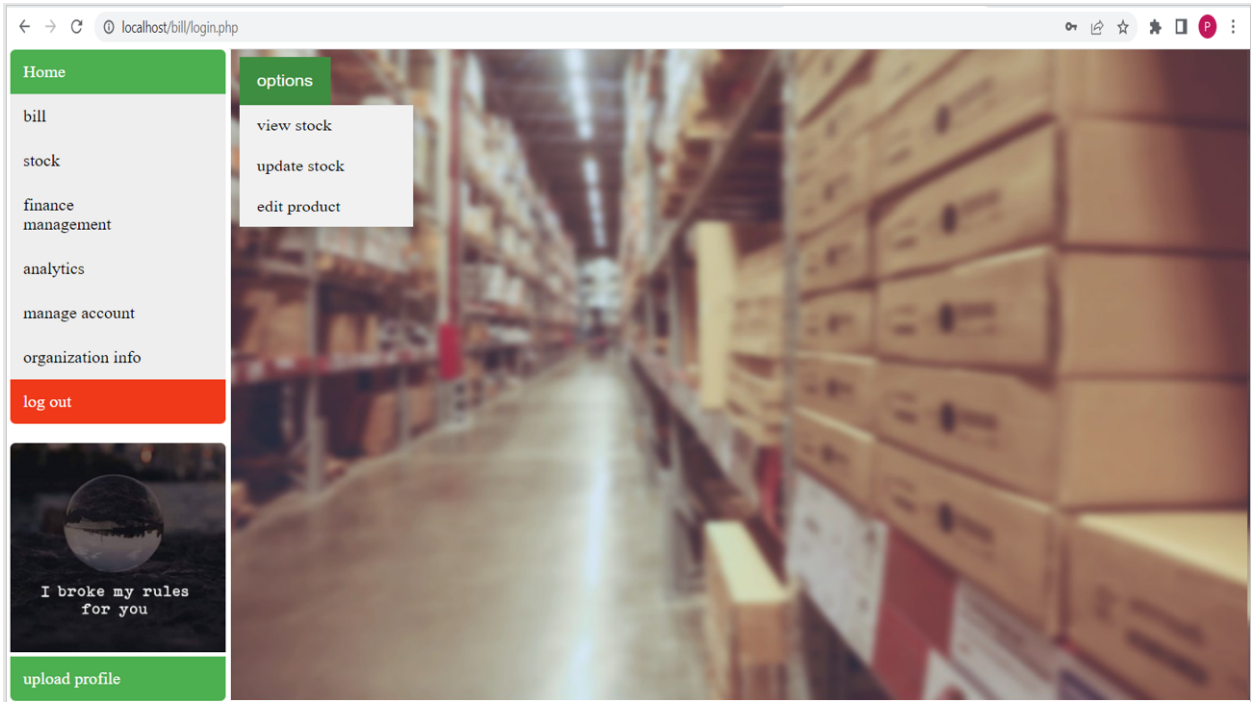
5) Login History viewing page:



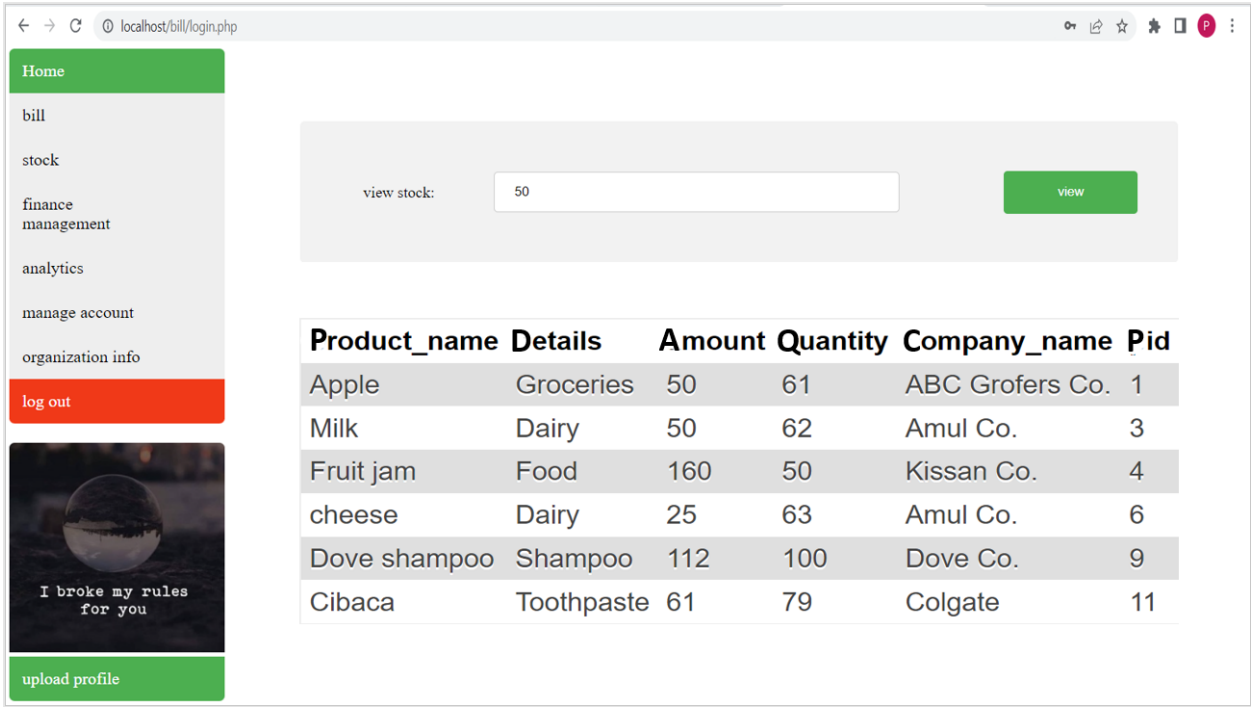
6) Result Of Request:



7) Operations related to stocks:

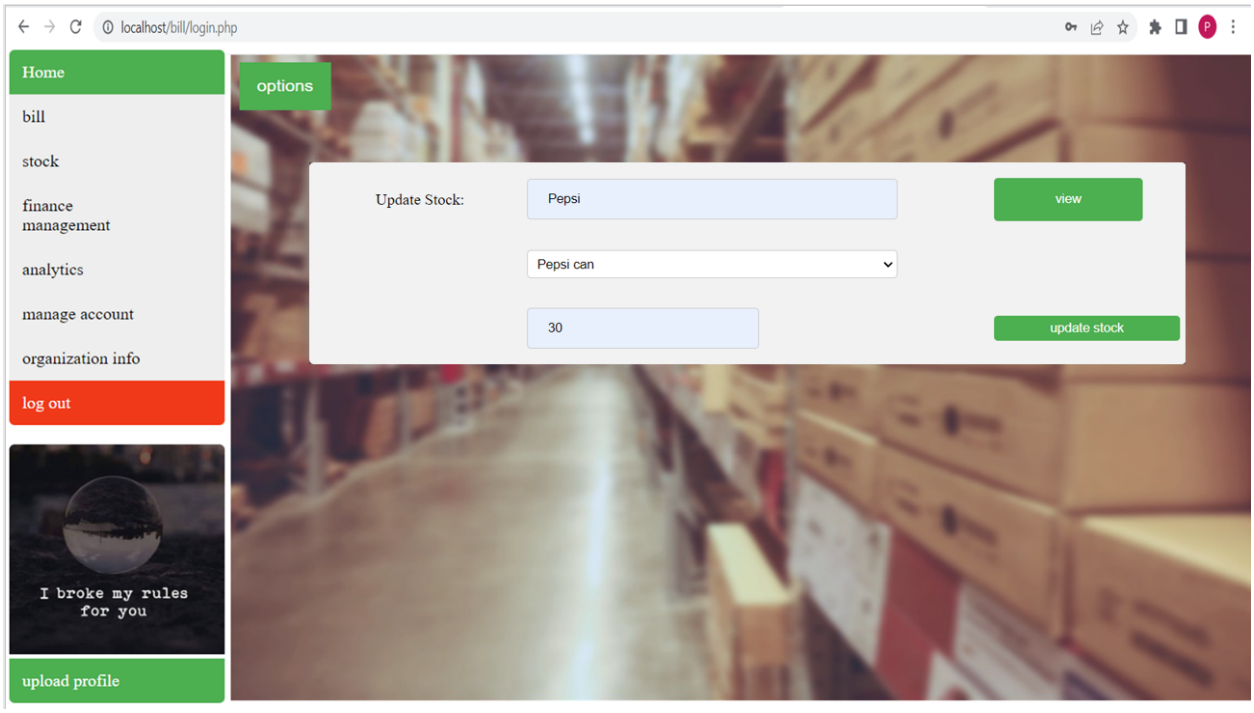


8) Result of finding products which have minimum stock of which is asked in request:

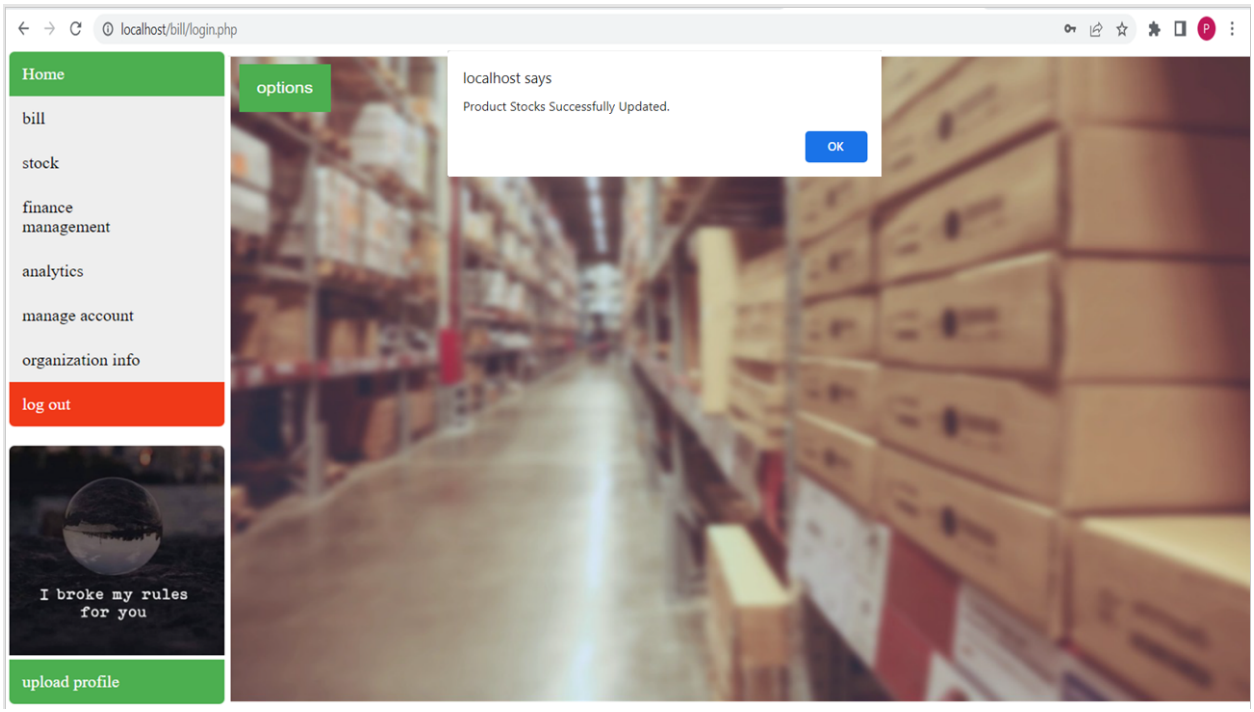


Product_name	Details	Amount	Quantity	Company_name	Pid
Apple	Groceries	50	61	ABC Grofers Co.	1
Milk	Dairy	50	62	Amul Co.	3
Fruit jam	Food	160	50	Kissan Co.	4
cheese	Dairy	25	63	Amul Co.	6
Dove shampoo	Shampoo	112	100	Dove Co.	9
Cibaca	Toothpaste	61	79	Colgate	11

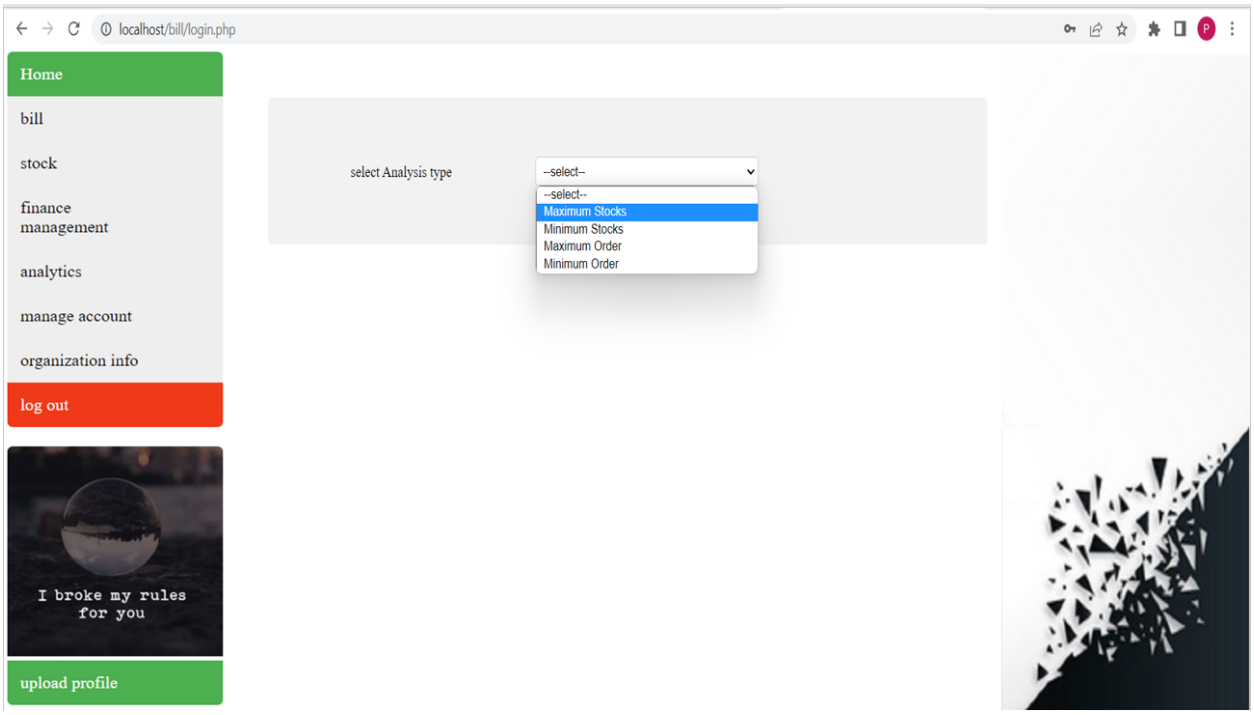
9) This is for updating the stock:



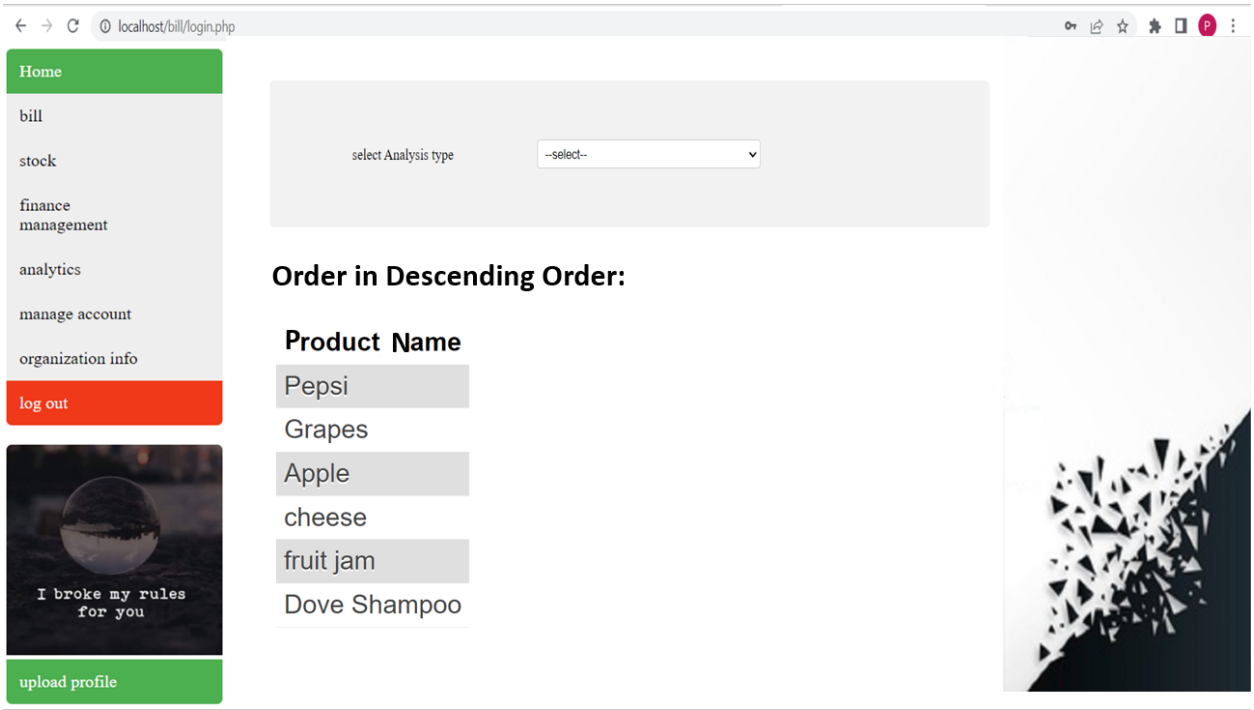
10) Result of request:



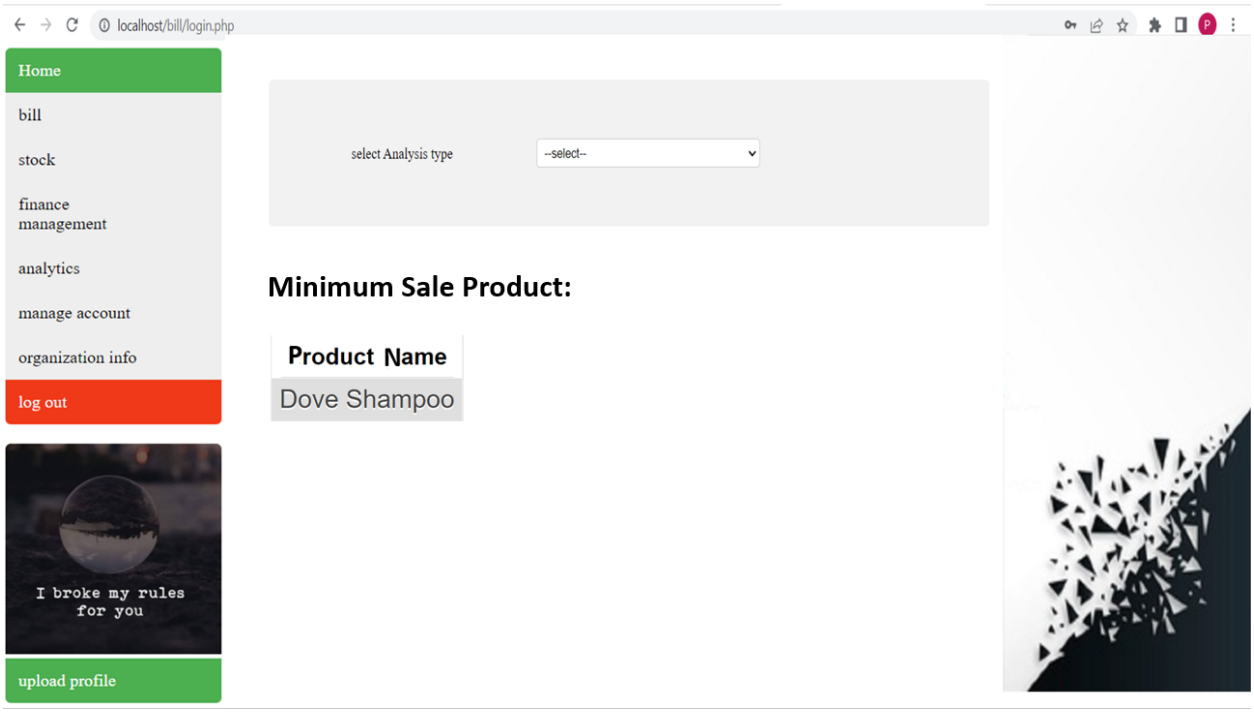
11) This is for analyzing the sales and the stocks:



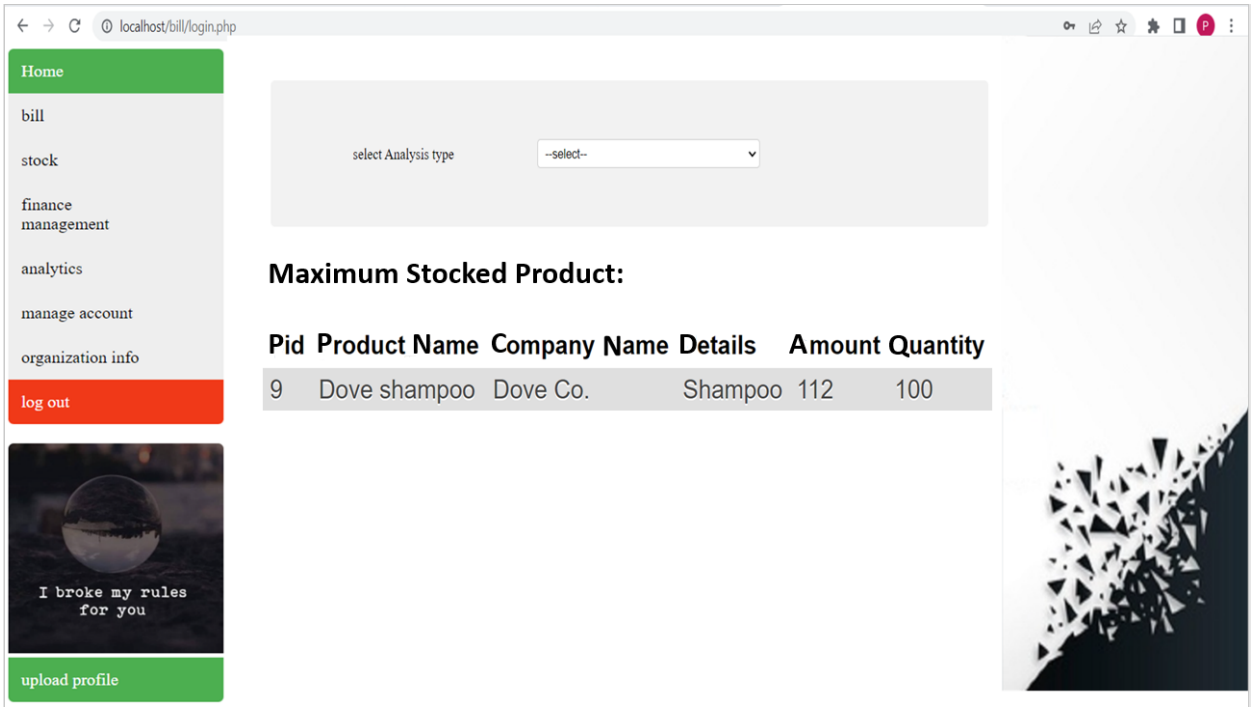
12) Result of maximum sales :



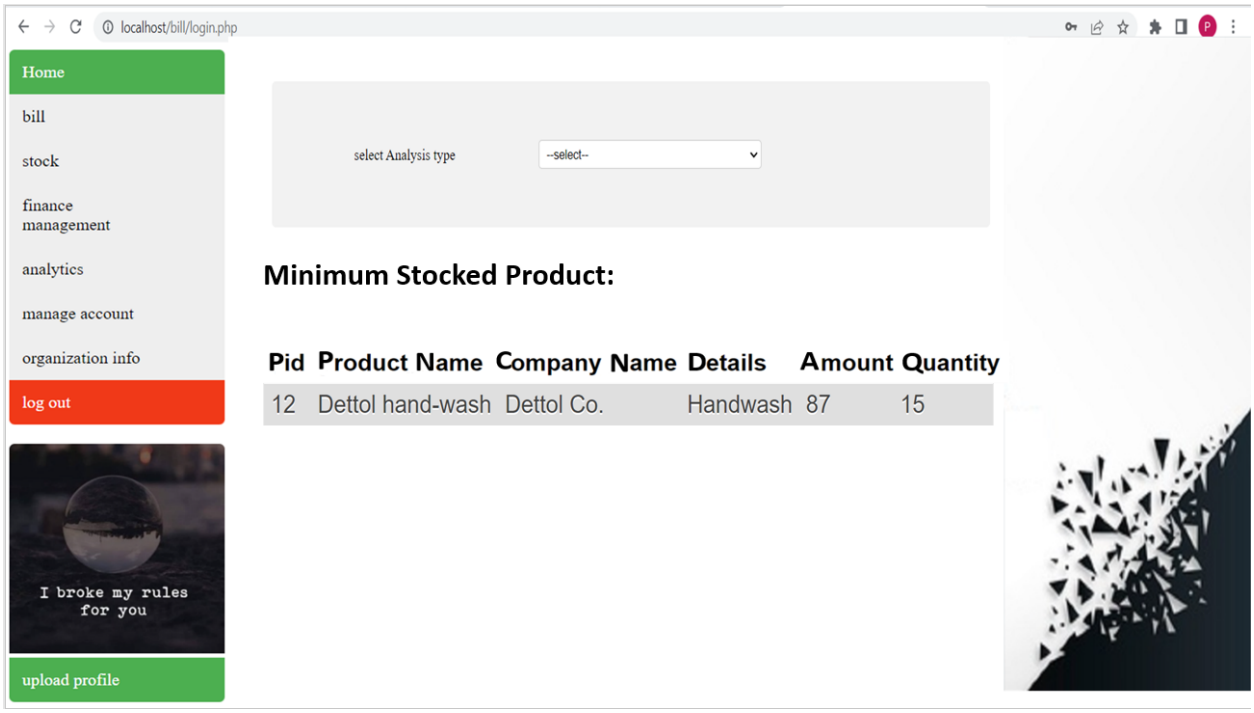
13) Result of minimum sales:



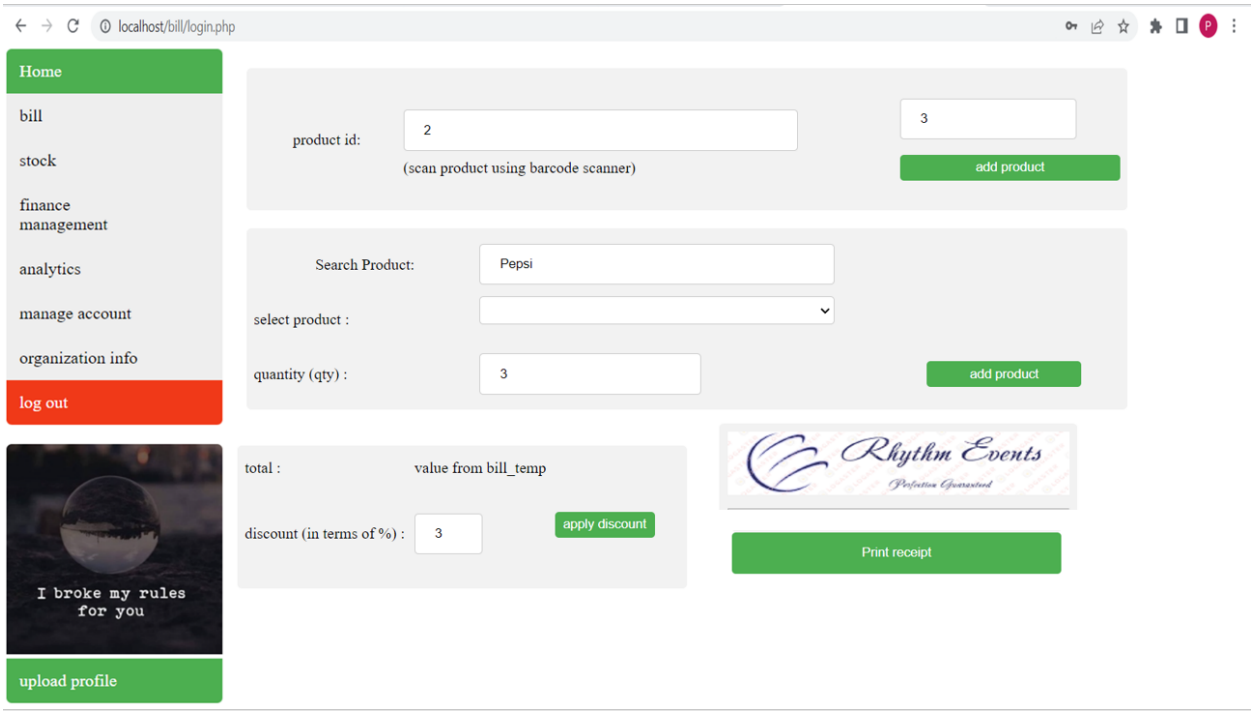
14) Result for maximum stock of which product:



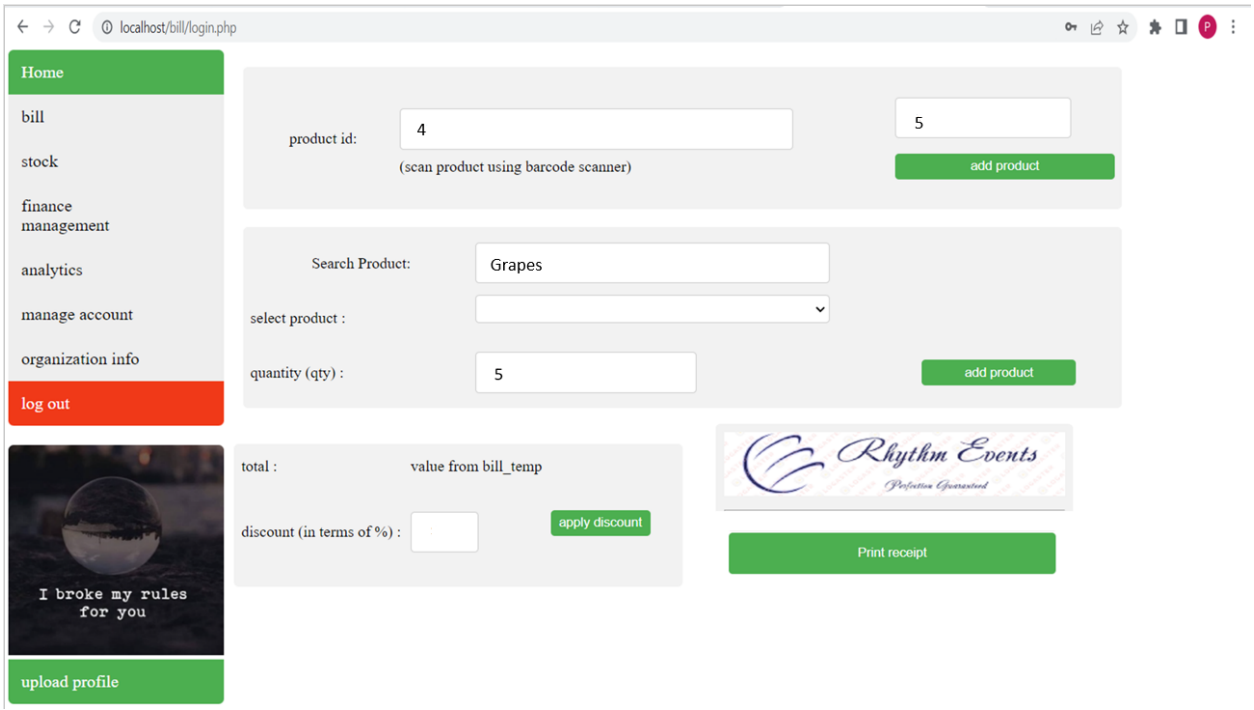
15) Result for minimum stock of which product:



16) Buying of the product:



17) Trying to activate the trigger by requesting the favorable request:



18) Result Of Trigger: