



Point Of Sale System Project

Preparation:

Laith Ashraf Ali Abu Shanab

9991027568.Ups@htu.edu.jo

Supervised by Eng. Bisharah Estephan

Letter Of Transmittal

Dear IMU,

I'm glad to provide the report on the building's new POS system that is attached. The methods used to create the system are thoroughly examined in this study.

As you can see, the building POS system has increased productivity and streamlined many of our routine tasks. But we've also found a few places where the system may be strengthened, and we've offered suggestions for fixing these problems in the study.

We hope that this research will be beneficial to you as you think about the constructing POS system's future in your company. Do not hesitate to get in touch with me if you have any queries or want to talk more about the report.

Regards,
Laith Abushanab
9 Jan 2023

Table of contents

Subject number	Subject name	Page number
1	Chapter One: Introduction	
1.1	Problem	7
1.2	The goal of the project	7
1.3	The importance of the project	8
1.4	The beneficiaries of the project	8
1.5	Tools and software used	9
2	Chapter Two: System Life Cycle	
2.1	System Analysis Goals	11
2.2	system design	11
2.2.1	System Participants	12
2.2.1.1	The primary webmaster	12
2.2.1.2	The Seller	12
2.2.1.3	The Procurement	12
2.2.1.4	The Accountant	12
2.2.2	Data Analysis	13
2.2.3	Data Flow Diagram	14

2.2.4	Entity relationship diagram	18
2.3	System implementation	20
2.4	Integration & Testing	27
2.5	maintenance	28
3	Chapter 3: Discussion & Conclusion	
3.1	Discussion	29
3.2	Conclusion	29
3.3	Recommendation	30
3.4	References	30
4	Chapter 4: Appendix	

List Of Tables:

Table Number	Description	Page
1	Administrator	19
2	Transactions	19
3	Stocks	19
4	Transaction User	20

List Of Figures:

Figure Number	Description	Page
1	System Lifecycle	10
2	Administrator (DFD)	14
3	Seller (DFD)	15
4	Accountant (DFD)	16
5	Procurement (DFD)	17
6	ER-Diagram	18
7	Logging in to the website	20
8	Home Page	21
9	Selling Dashboard	21
10	Shopping Cart's	22
11	Items Page	22
12	Edit Item	23
13	Create Item	23
14	Transactions Page	24
15	Transaction Edit Page	24
16	Users Page	25

17	Create User Page	25
18	Information User Page	26
19	Edit User Page	26
20	MVC	31
21	method to call the class Route	32
22	method to Set the Value Class Name	32
23	Function to Connection With DB	33
24	Call Controller Function	33
25	Call Model Function	34

GLOSSARY

POS	Point Of Sales System
-----	-----------------------

Equative summary:

In this report, I discussed how I developed a point-of-sale (POS) system that can process transactions, keep track of inventory, and manage goods using PHP and the MVC design. We used PHP, MySQL, and the XAMPP server for the backend while using HTML, CSS, Bootstrap, JavaScript, and jQuery for the front end.

Following the creation of the database structure, which includes tables for users, items, and transactions, an MVC application architecture was established. Then we developed a point-of-sale system that can manage items, maintain inventories, and handle transactions.

And this approach will boost sales effectiveness, boosting earnings and giving the store a significant competitive advantage.

1 Introduction

A Point of Sale (POS) System is necessary for almost all retail-based activities to accurately record customer and product activity.[1] Every business owner wants to know which products sell the most, which customers purchase the most frequently, how much inventory is on hand, which payment method customers prefer (cash, credit, checks, or debit), and how much was really sold on a given day.[2]

1.1 Problem:

Our project is driven by the need to develop a Point of Sale (POS) website that successfully addresses the difficulties HTU has managing the store at King Abdullah Business Park. The POS system will oversee handling customer transactions, monitoring sales and inventory, and giving the administrator access to real-time information about the operation of the store. We want to increase the productivity and efficiency of the store's operations while also enhancing the overall shopping experience for customers by automating these activities. The POS website will also be user-friendly and intuitively built to make it simple for staff members in a range of positions to use. Our overall objective is to create a POS system that would enable HTU to properly manage

1.2 The goal of the project:

Point of sale (POS) refers to the system or location where a transaction takes place, such as a checkout counter at a retail store or a restaurant. The aim of a POS system is to facilitate the sale of goods or services and to provide an efficient and accurate way for businesses to process transactions. A POS system typically includes a computer or terminal, a printer, a payment processing device (such as a cash drawer or card reader), and software that manages the transactions and inventory. The aim of a POS system is to streamline the sales process, reduce errors, and provide valuable data and insights for businesses. POS systems may also offer additional features such as customer relationship management, inventory management, and reporting capabilities.

1.3 The importance of the project:

- ✓ Increased efficiency: A POS system can help a business process transaction more quickly and accurately, reducing the time customers spend waiting in line and increasing
- ✓ Improved customer service: A POS system can provide helpful information to staff, such as customer purchase history and special promotions, which can enhance the customer experience.
- ✓ Enhanced data management: A POS system can collect and store data on sales, customer behavior, and inventory levels, providing valuable insights for businesses to make informed decisions about operations and marketing.

1.4 The beneficiaries of the project:

There are several beneficiaries of a point of sale (POS) system:

- ✓ Businesses: A POS system can improve efficiency, customer service, and data management for businesses, leading to increased profits and success.[3]
- ✓ Customers: A POS system can improve the speed and accuracy of transactions, enhancing the customer experience and reducing the time spent waiting in line. [3]
- ✓ Employees: A POS system can reduce the workload for employees by automating tasks and providing helpful information, such as customer purchase history and special promotions.
- ✓ Investors: A POS system can improve a business's financial performance, which can be attractive to investors.
- ✓ Government: A POS system can help businesses comply with tax laws and regulations, benefiting the government by increasing tax revenues.[3]

1.5 Tools and software used

Technical Feasibility Our system is a web-based application and the main technologies and tools that are associated with this project are:

1- For Front-end we will use:

- ✓ HTML
- ✓ CSS
- ✓ Bootstrap
- ✓ JavaScript and jQuery

2- For Back-end we will use:

- ✓ PHP
- ✓ MySQL
- ✓ Xampp Server

System life cycle

Each system has a life cycle (System Development Life Cycle), through which the system is built and developed based on sequential stages, as shown in Figure (1):

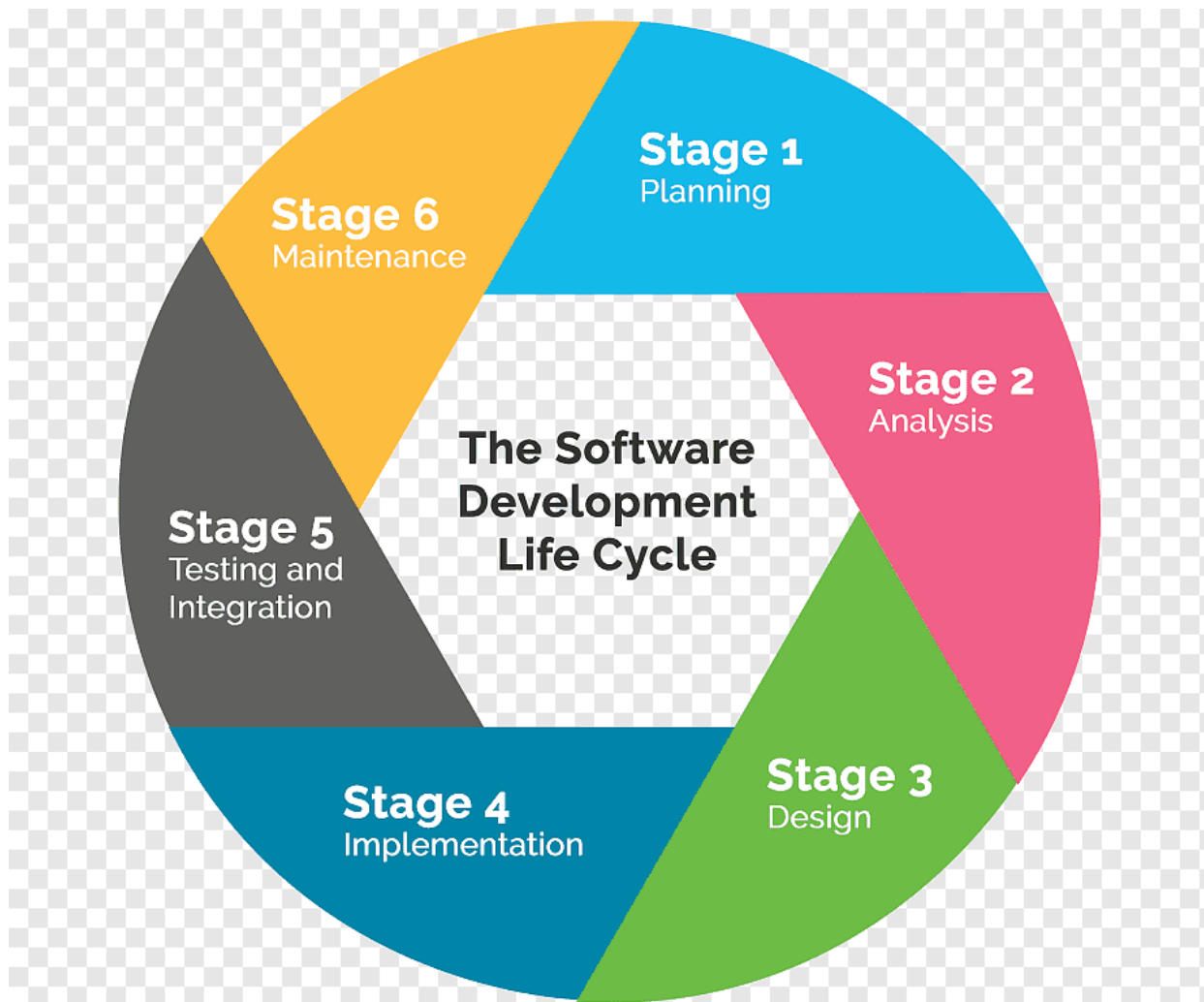


Figure (1): System Lifecycle (SDLC) [4]

2.1 System Analysis Goals:

By performing a system analysis, we were able to accomplish several objectives, including:

- ✓ Establish the system requirements.
- ✓ Identify the procedures required to satisfy system requirements.
- ✓ Deciding on the right hardware and software for the system's construction and development.
- ✓ Examining potential solutions, spotting flaws, and ensuring that processes are following the suggested designs to fulfill the necessary standards.
- ✓ He suggested designs that will adhere to the needed criteria.

2.2 system design:

The system analysis stage, one of the most crucial stages of the system life cycle, plays a major role in this step because if the analysis process is done correctly, the design stage will be of a creative and optimal nature. We are looking for optimal solutions to build the application, so it requires creative skills and plans to choose the optimal solution.

The main goal of the design process is to solve problems, that is, to find the best design solutions to build a system with specific and integrated goals. The goal is the basic and first criterion on which the quality of the design reached is evaluated, and thus the design stage begins from the system objectives that were identified in the previous analysis stage.

The user interface and all inputs, outputs, and processes in the system have been determined at this stage, and the most important part at this stage is to determine the structure of the system so that programmers and developers have a clear picture of how to translate the special designs in the system into units and scripts.

2.2.1 System Participants:

The system consists of four main parties:

- 1- Admin
- 2- Seller
- 3- Procurement
- 4- Accountant

2.2.1.1 The primary webmaster:

The system administrator has complete control over all aspects of the system, including:

- 1- Managers can be added, modified, and deleted.
- 2- Adding, changing, and removing items
- 3- Changing and removing sales transactions
- 4- Selling, removing, or altering things

2.2.1.2 The Seller

The Seller has total authority over the Selling dashboard.

- 1- Selling, removing, or altering things

2.2.1.3 The Procurement

The Stocks dashboard is completely under the control of Procurement.

- 1- Adding, changing, and removing items

2.2.1.4 The Accountant

The Transaction Dashboard is completely under the control of Accountant.

- 1- Changing and removing sales transactions

2.2.2 Data Analysis:

Plan the POS system's functionality: Determine the features and capabilities that you want the POS system to have. This could include inventory management, transaction processing, report generation, and other tasks. Create the development environment: Install on your computer a local server, such as Apache or Nginx, as well as a database management system, such as MySQL.

- ❖ Create a database schema: which includes tables for storing information about products, customers, transactions, and any other relevant data.
- ❖ Create the models: The models in an MVC application represent the application's data and business logic. You could create models for products, customers, and transactions for a POS system.
- ❖ Plan the POS system's functionality: Determine the features and capabilities that you want the POS system to have. This could include inventory management, transaction processing, report generation, and other tasks.
- ❖ Create the development environment: Install on your computer a local server, such as Apache or Nginx, as well as a database management system, such as MySQL.

The design phase is an analytical process of project needs and the construction of the needed structure and its pieces, as well as how they interact with one another. This produces a set of files, forms, and diagrams that may be used to implement and program the complete project.

Several instructive graphics were created at this point to aid in the work mechanism.

- 1- ER-Diagram
- 2- Data Flow Diagram – DFD

2.2.3 Data Flow Diagram (DFD):

First: there is the administrator.

The data flow diagram for the system administrator is shown in Figure (2).

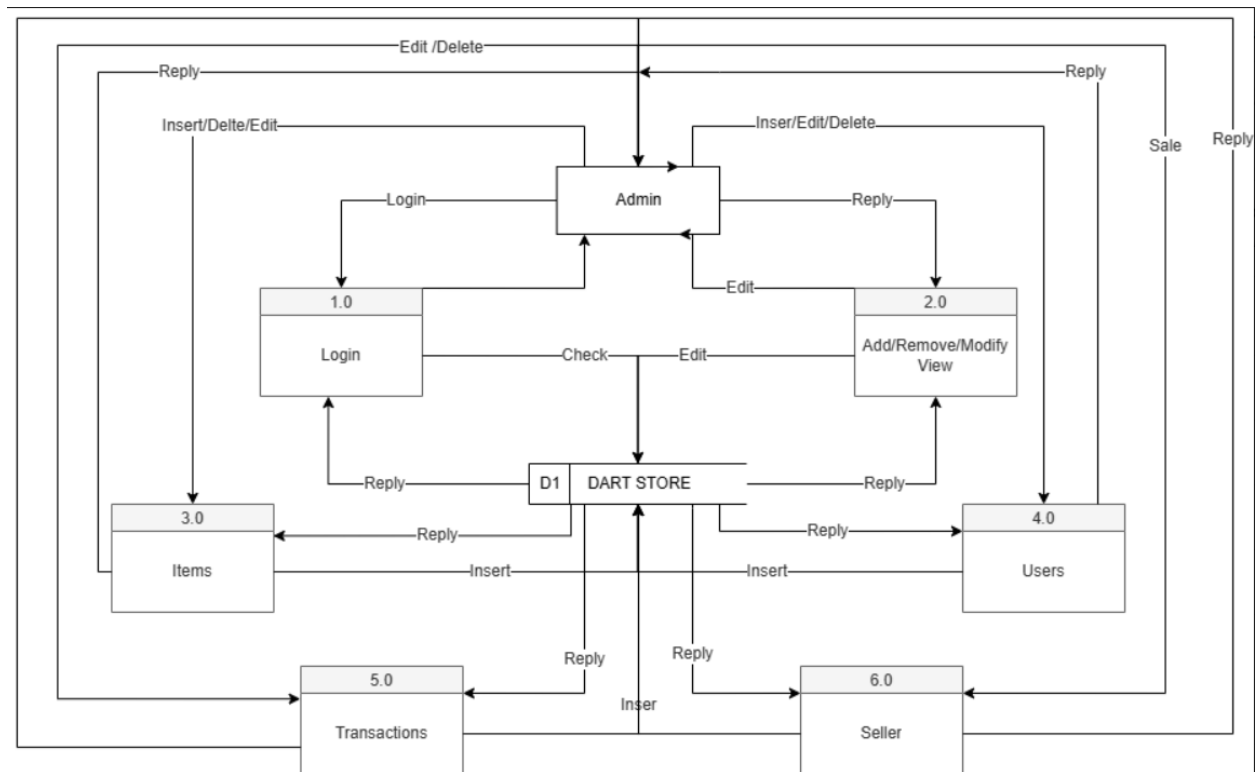


Figure (2): Datagram Data for System Administrator

Secondly: there is the seller.

The data flow diagram for the system seller is shown in Figure (3).

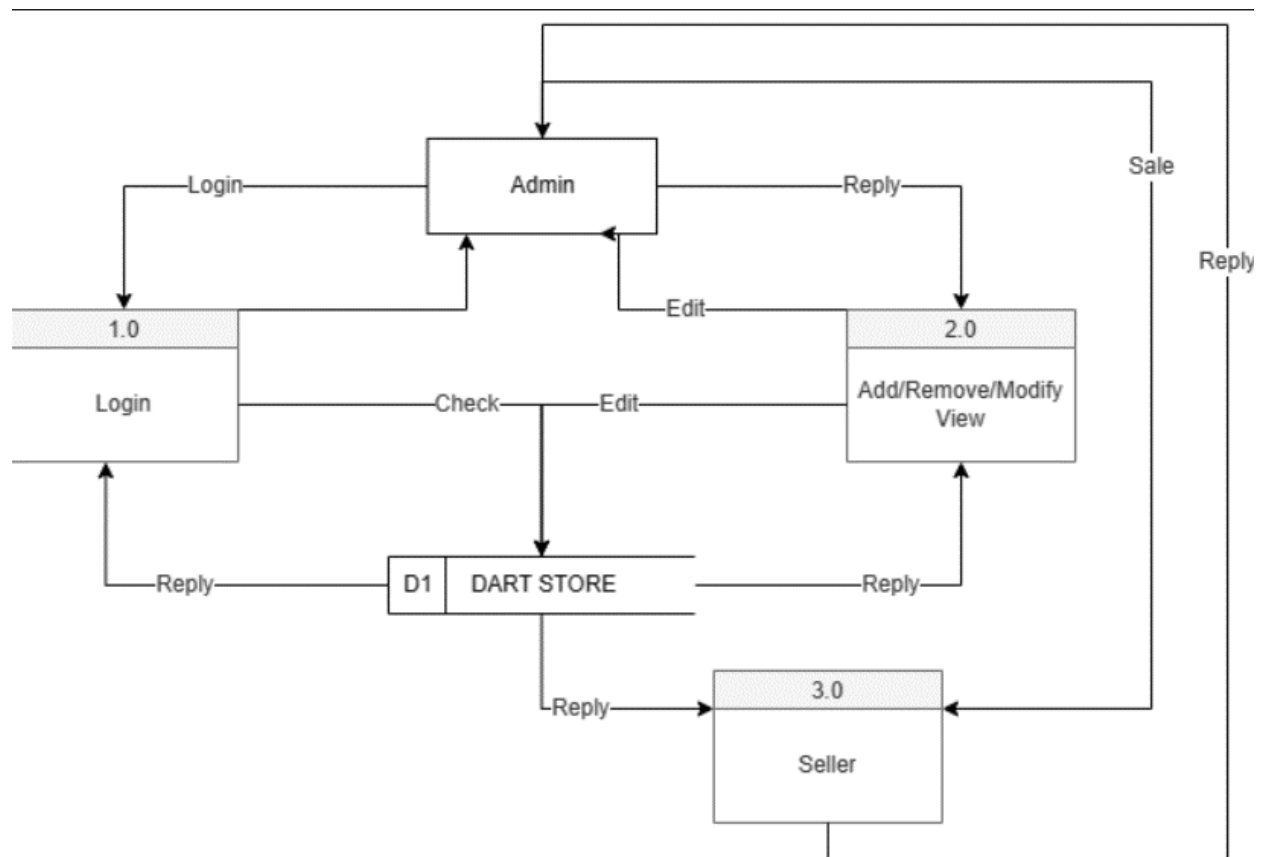


Figure (3): Datagram Data for System Seller

Third: there is the accountant.

The data flow diagram for the system Accountant is shown in Figure (4).

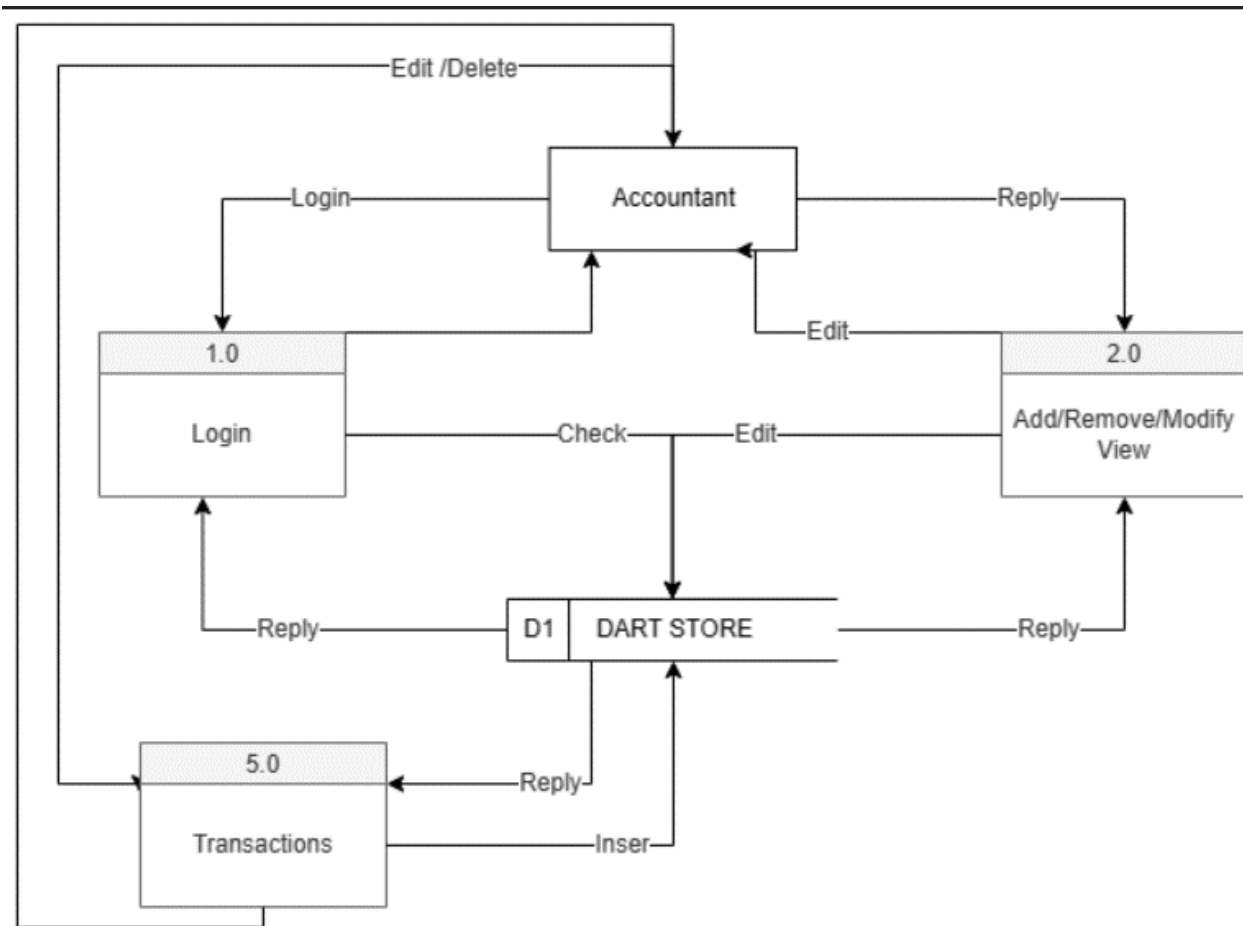


Figure (4): Datagram Data for System Accountant

Fourthly: there is the Procurement.

The data flow diagram for the system Procurement is shown in Figure (5).

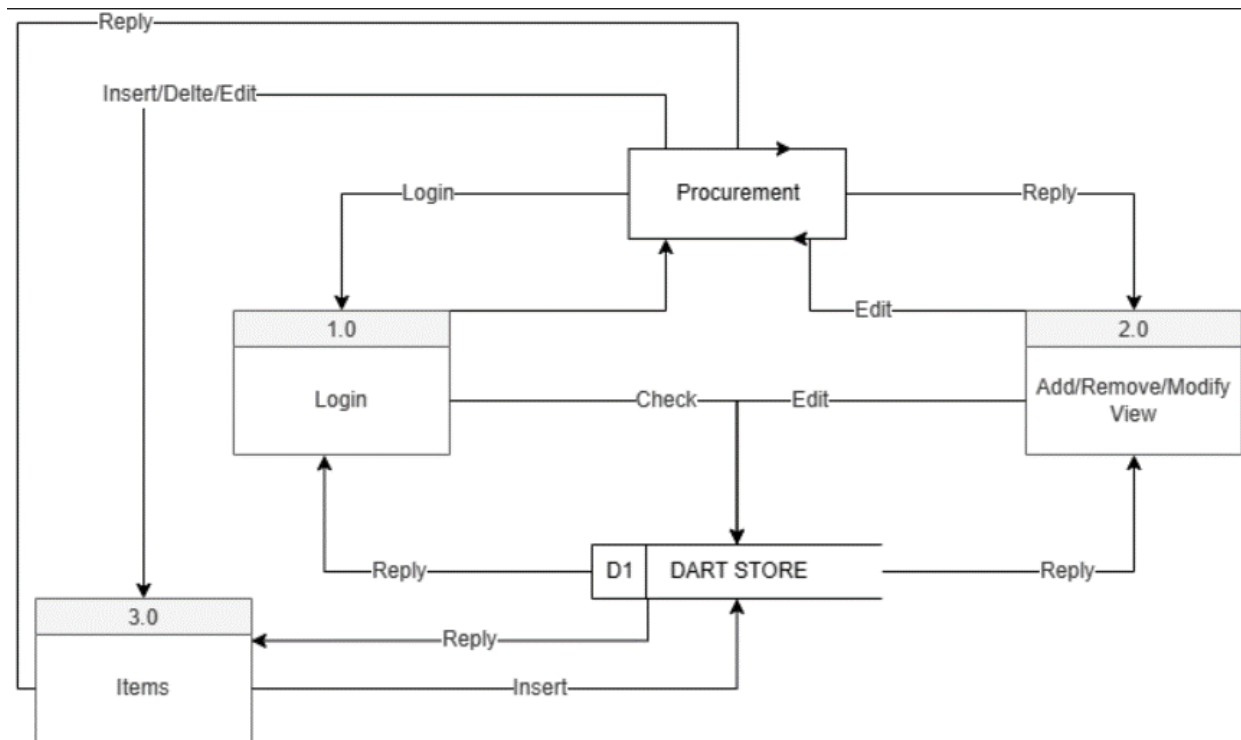


Figure (5): Datagram Data for System Procurement

2.2.4 Entity relationship diagram

The entity relationship diagram is depicted in Figure (6).

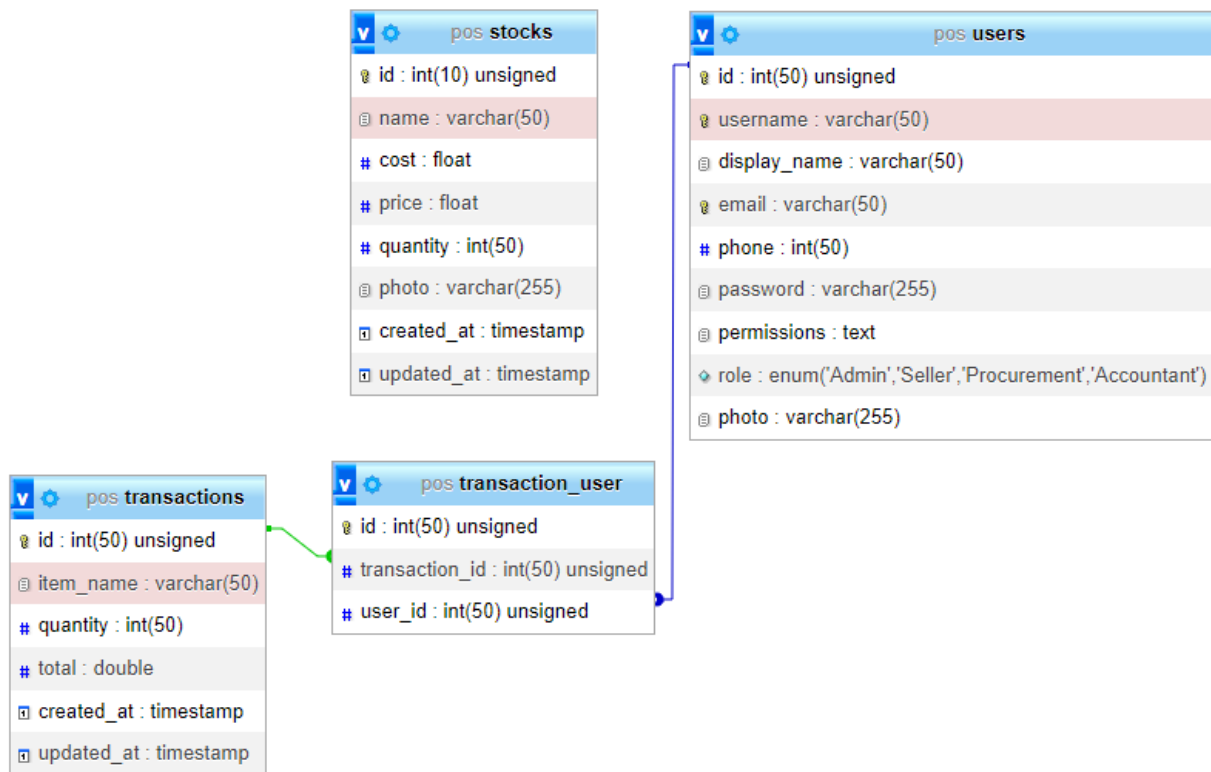


Figure (6): Entity Relationship Diagram

Database and system page design:

The system includes the following primary tables:

Table (1): The site's administrator

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 id	int(50)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/>	2 username	varchar(50)	utf8_general_ci		No	None			Change Drop More
<input type="checkbox"/>	3 display_name	varchar(50)	utf8_general_ci		No	None			Change Drop More
<input type="checkbox"/>	4 email	varchar(50)	utf8_general_ci		No	None			Change Drop More
<input type="checkbox"/>	5 phone	int(50)			Yes	NULL			Change Drop More
<input type="checkbox"/>	6 password	varchar(255)	utf8_general_ci		No	None			Change Drop More
<input type="checkbox"/>	7 permissions	text	utf8_general_ci		Yes	NULL			Change Drop More
<input type="checkbox"/>	8 role	enum('Admin', 'Seller', 'Procurement', 'Accountant', ...)	utf8_general_ci		Yes	NULL			Change Drop More
<input type="checkbox"/>	9 photo	varchar(255)	utf8_general_ci		Yes	NULL			Change Drop More













Table (2): The Table Transactions

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 id	int(50)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/>	2 item_name	varchar(50)	utf8_general_ci		No	None			Change Drop More
<input type="checkbox"/>	3 quantity	int(50)			No	None			Change Drop More
<input type="checkbox"/>	4 total	double			No	None			Change Drop More
<input type="checkbox"/>	5 created_at	timestamp			Yes	current_timestamp()			Change Drop More
<input type="checkbox"/>	6 updated_at	timestamp			No	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()	Change Drop More

Table (3): The Table Stocks

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 id	int(10)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/>	2 name	varchar(50)	utf8_general_ci		No	None			Change Drop More
<input type="checkbox"/>	3 cost	float			No	None			Change Drop More
<input type="checkbox"/>	4 price	float			No	None			Change Drop More
<input type="checkbox"/>	5 quantity	int(50)			No	None			Change Drop More
<input type="checkbox"/>	6 photo	varchar(255)	utf8_general_ci		Yes	NULL			Change Drop More
<input type="checkbox"/>	7 created_at	timestamp			No	current_timestamp()			Change Drop More
<input type="checkbox"/>	8 updated_at	timestamp			No	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()	Change Drop More

Table (4): Relationship table between the transaction and the user

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 id 	int(50)		UNSIGNED	No	None		AUTO_INCREMENT	 Change  Drop  More
<input type="checkbox"/>	2 transaction_id 	int(50)		UNSIGNED	No	None			 Change  Drop  More
<input type="checkbox"/>	3 user_id 	int(50)		UNSIGNED	No	None			 Change  Drop  More

2.3 System implementation

After depending on the system's analysis and design stages, the implementation stage builds the whole system's components, including pages and lists, so that all of the system's pages are presented and the function of each page is explained separately. The website is meant to be easy and adaptable in order to suit the demands of all users. It was developed in English, and if the site's concept is successful, the Arabic language will be incorporated in the future.

Login page for the website:

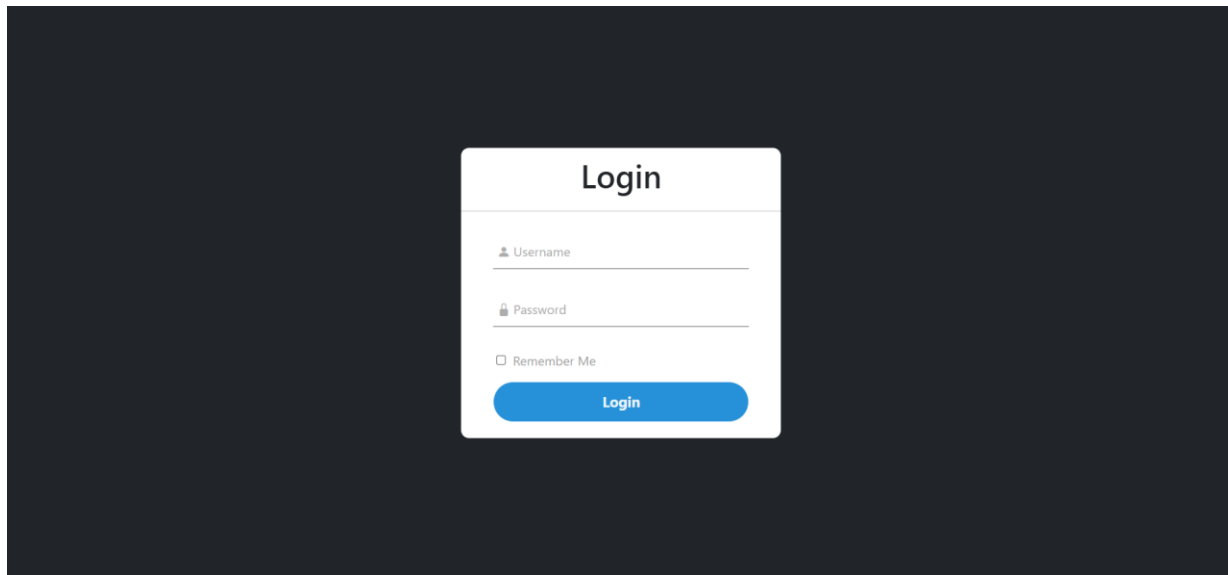


Figure (7): Logging in to the website

This screen displays a form with two components (Email and Password), a login button, and a verification button (REMEMBER ME). Its purpose is to save the (Username and Password) when you return to the site. It is the initial page the user encounters. It was developed in a basic and adaptable manner, with no complication, and from here the user may log in to the site.

Home Page (Dashboard):

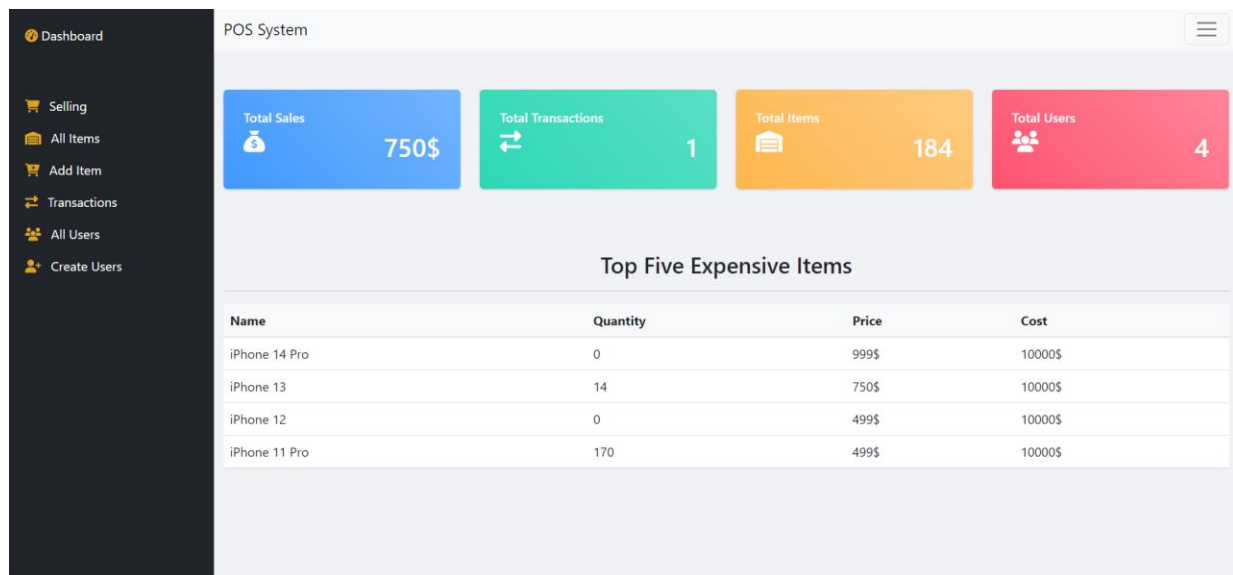


Figure (8): Home Page

The system's core categories and certain information are presented on the system's main page, such as: Total Sales, Total Transactions, Total Items, Total Users, Top Five Expensive Items. It also has the pages that the manager works with on the left side.

Selling Dashboard:

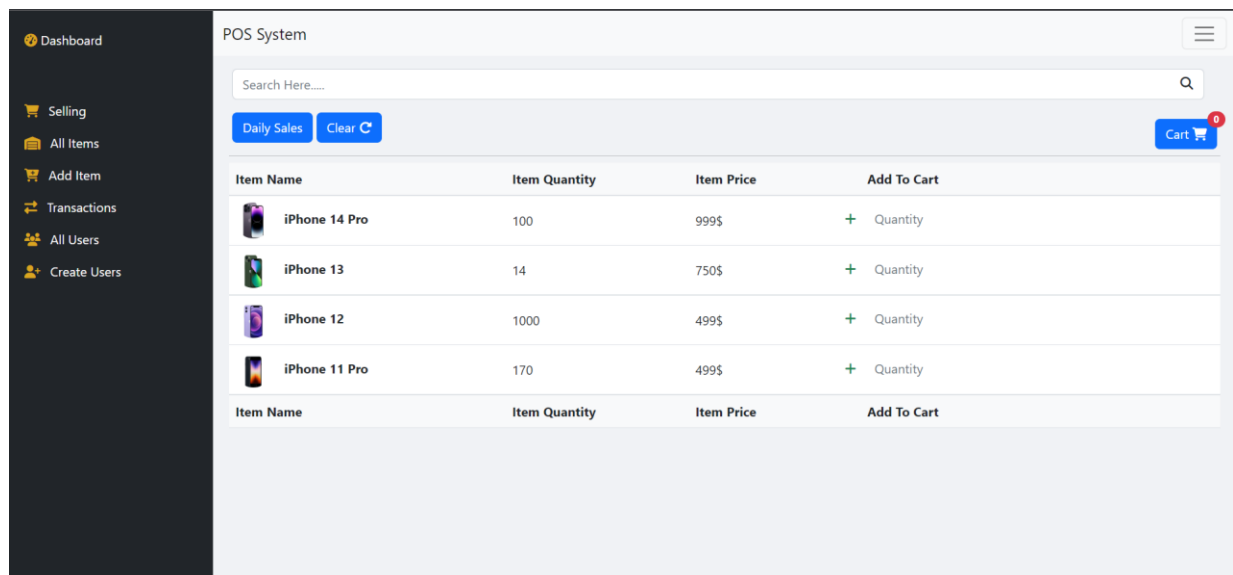


Figure (9): Selling Dashboard

This page lists all of the things in the store that are currently for sale. To add the product to the shopping cart, the seller can pick the quantity desired and hit the add button. The top of the website also has a search box, which allows the seller to readily find the goods. The page also includes the daily sales button, which displays all of the sales made by the user today, and the reload button, which allows the user to empty the shopping cart in preparation for a new purchase.





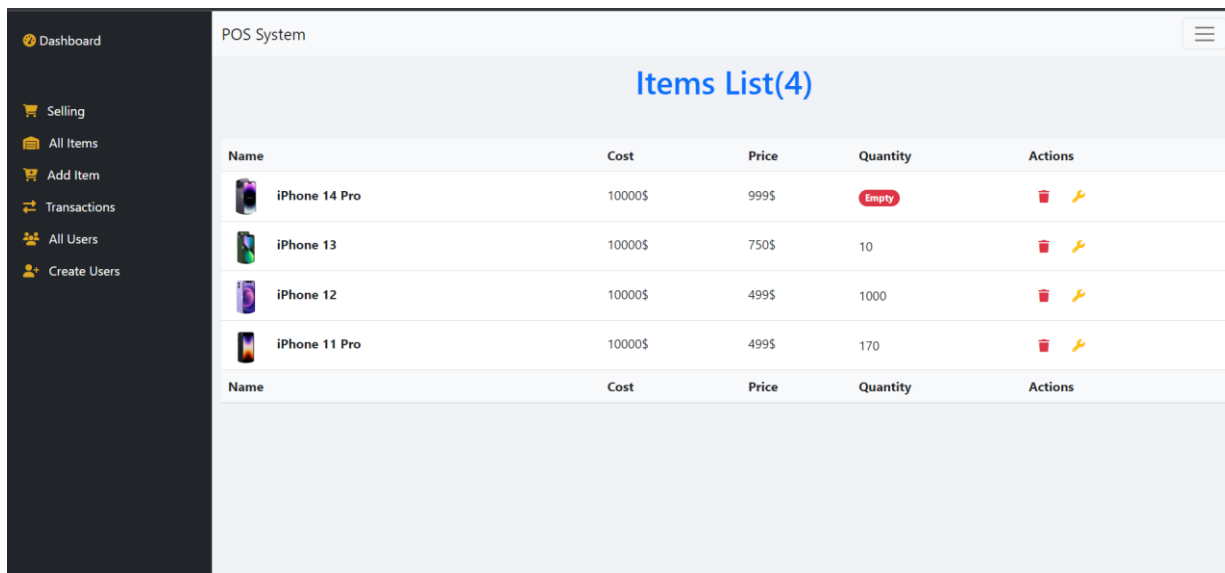
Item Name	Quantity	Total	Delete
iPhone 13	4	3000\$	 

Figure (10): Shopping Cart's

This is the shopping cart's contents, from which the seller may either adjust the quantity or remove the order entirely.

Items page:















Name	Cost	Price	Quantity	Actions
 iPhone 14 Pro	10000\$	999\$	Empty	 
 iPhone 13	10000\$	750\$	10	 
 iPhone 12	10000\$	499\$	1000	 
 iPhone 11 Pro	10000\$	499\$	170	 

Figure (11): Items Page

This website functions as a store, allowing you to see how many things we have in stock as well as information about each product, such as cost, pricing, and quantity. There are also two buttons

for each item, the first of which allows you to alter the product and the second of which allows you to delete the product.

The screenshot shows a web application interface for a POS System. On the left is a dark sidebar with navigation links: Dashboard, Selling, All Items, Add Item, Transactions, All Users, and Create Users. The main content area has a header 'POS System' and a sub-header 'Updating Items in the store' with a red 'cancel' button. A central form is titled 'Updating Item' and contains the following fields: 'Item name' (text input with 'iPhone 14 Pro'), 'Item cost' (text input with '10000'), 'Item price' (currency input with '\$ 999' and '.00'), 'Item quantity' (text input with '0'), and 'Item photo' (file upload area with 'Choose File' and 'No file chosen' buttons). A yellow 'Update' button is at the bottom right of the form.

Figure (12): Edit Item

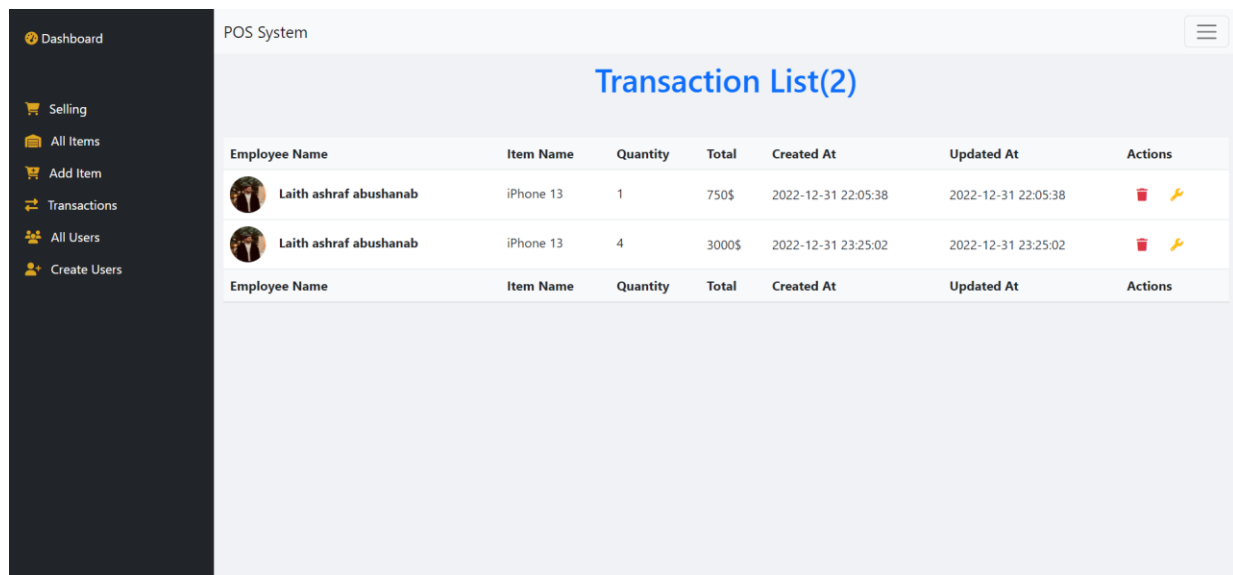
This page is about modifying the product, through which we can amend the product name, price, cost, or quantity, and modify the product image

The screenshot shows the same web application interface as Figure 12, but for adding a new item. The sub-header is 'Adding Items to the store' with a red 'cancel' button. The central form is titled 'Adding Item' and contains the following fields: 'Item name' (empty text input), 'Item cost' (empty text input), 'Item price' (currency input with '\$' and '.00'), 'Item quantity' (empty text input), and 'Item photo' (file upload area with 'Choose File' and 'No file chosen' buttons). A green 'Create' button is at the bottom right of the form.

Figure (13): Create Item

We may add a new product to the store using this page and give its name, cost, price, and picture.

Transactions Page:









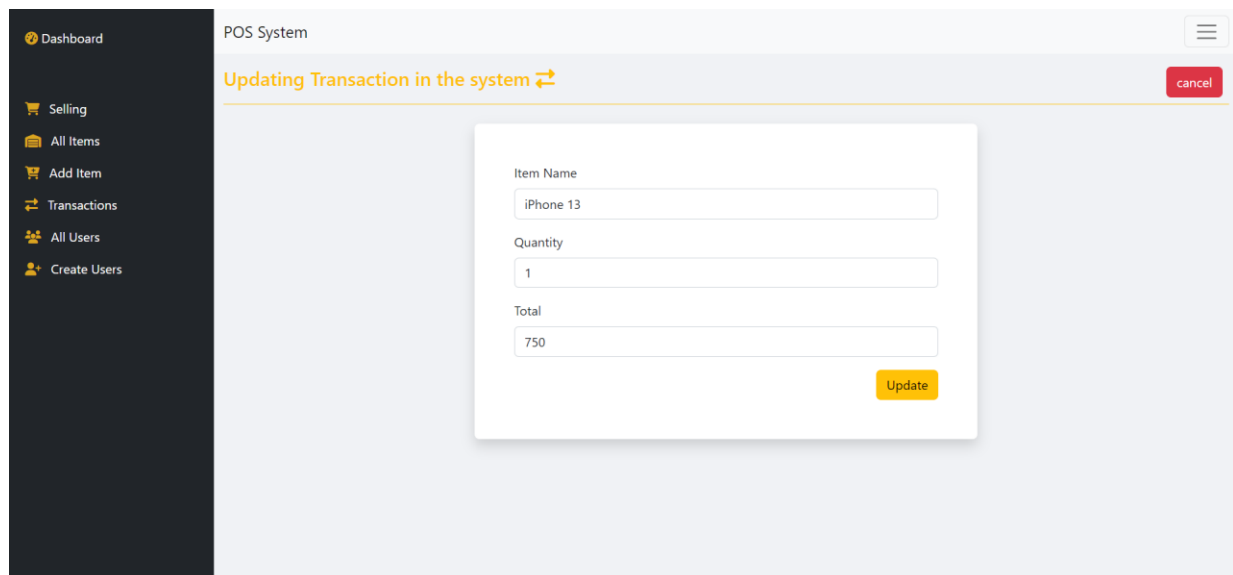

Employee Name	Item Name	Quantity	Total	Created At	Updated At	Actions
 Laith ashraf abushanab	iPhone 13	1	750\$	2022-12-31 22:05:38	2022-12-31 22:05:38	 
 Laith ashraf abushanab	iPhone 13	4	3000\$	2022-12-31 23:25:02	2022-12-31 23:25:02	 
Employee Name	Item Name	Quantity	Total	Created At	Updated At	Actions

Figure (14): Transactions Page

This page is for the accountant to see all of the sales transactions that have occurred, as well as the name of the seller, the name of the product, and the amount sold by the seller. Each process also has two buttons, the first of which allows you to modify the sale process and the second of which allows you to remove the process.



Updating Transaction in the system 

cancel

Item Name
iPhone 13

Quantity
1

Total
750

Update

Figure (15): Transaction Edit Page

Here, the accountant may adjust the quantity of a certain sale without changing the name or the quantity, and as soon as the amount changes, the total changes automatically.

Users Page:

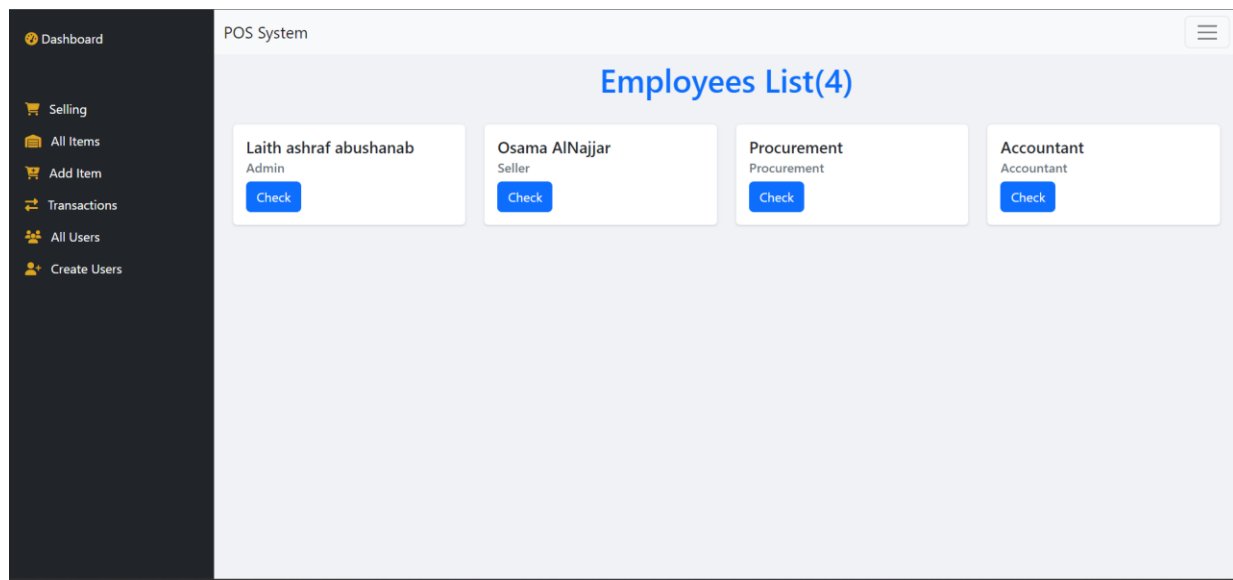


Figure (16): Users Page

This page is for the site management, and it allows them to see the number of users on the site as well as the powers of each user, as well as the check me button for each user, which provides particular information for each user.

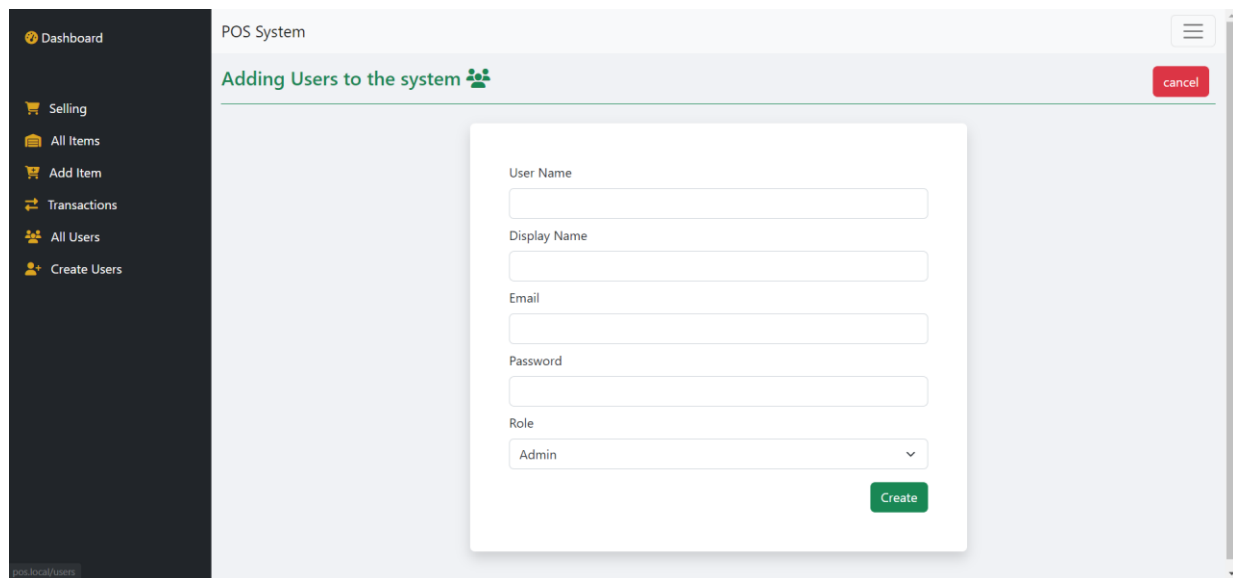


Figure (17): Create User Page

This page allows the site administrator to add users and delegate authority to them, such as an accountant, salesperson, or shop manager.

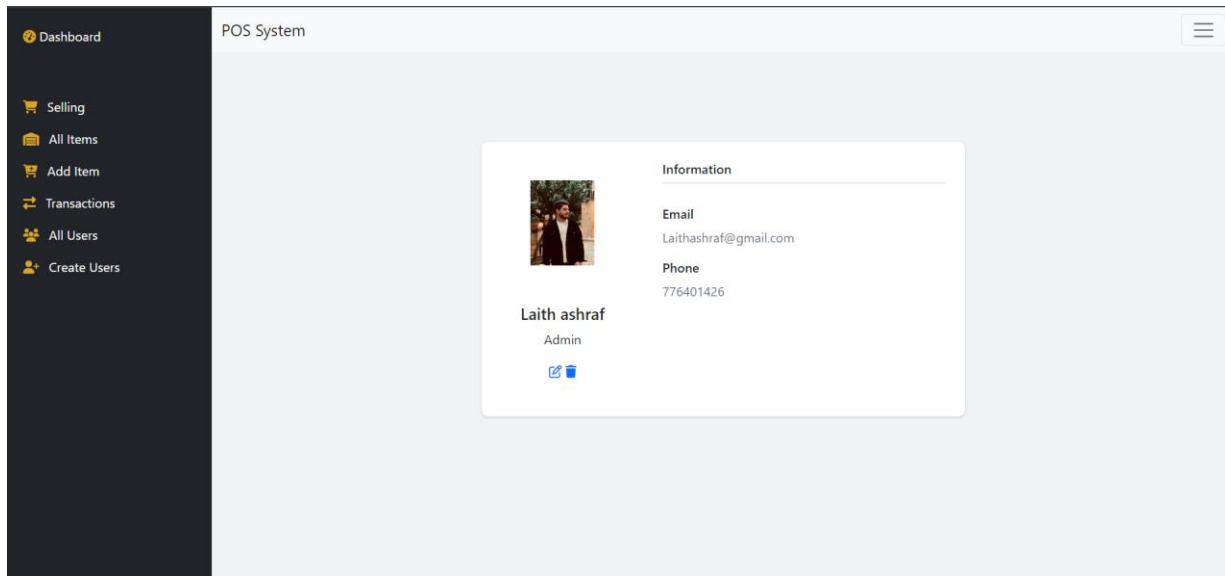


Figure (18): Information User Page

Through this page, the system administrator may view the information for each user, and there are two buttons, the first to adjust the user's privileges, and the other to limit the user.

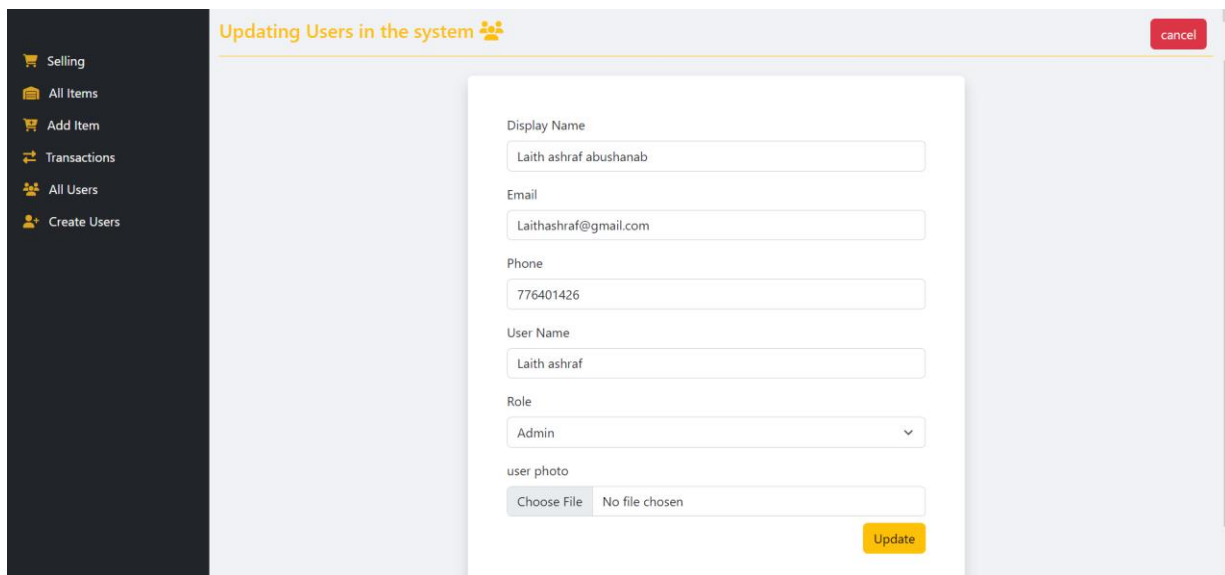


Figure (19): Edit User Page

This page is intended for the system administrator and the user but the user lacks the permission function, which allows the system administrator to update the user's information, his role in the system and even the user's photo. The user can only modify their personal information, such as their photo, name, and phone number.

2.4 Integration & Testing

The first stage of system testing involves evaluating each component individually and isolating it from the rest of the system components to ensure that it functions as intended by checking the information gained from it (Output) after giving it with the relevant data (Input).

First, there is the integration test:

After testing all the system's components and establishing their design integrity, we must ensure that they will perform correctly together and that there will be no conflict between them so that the information communicated between these components arrives in the intended form. This is what integration testing is all about.

Second, there is employment testing:

It entails testing the system after assembling all of its components to confirm that it performs the function specified in the system requirements specifications. When the system passes this test, we may consider it to be operational.

Third, there is performance testing.

The performance of the program is tested in the customer's work environment in this stage to confirm that the system is compliant with the remaining criteria. When this test is passed, the system is certified. As a result, we believe the system is complete based on our understanding of what the customer desired.

2.5 maintenance

Software maintenance is defined as the process by which programs are updated, new tasks are added, software errors are corrected, and programming problems are resolved through maintenance contracts signed by programming companies with their customers and calculated as annual fees based on a percentage of the program's total cost.

Types of software maintenance:

- 1- Adaptive maintenance results from internal changes to an organization's software systems, such as transferring programs to new devices, or to compilers and other operating systems, in order to adapt to external requirements and keep up with modernity in meeting the needs of the user and business sectors.
- 2- Corrective maintenance: The essence of corrective maintenance work is the process of modifying and improving defective problems in systems and programs, so that the code, program structures, and program alerts are modified, and either the need for them comes from the user or from error reports that appear in the programs, so the repair is either for emergency failures, or a scheduled operation for adjustment and correction.
- 3- Preventive maintenance: programs are reorganized, also known as software re-engineering, with the goal of preventing future software issues, so that systems become more intelligible, their benefits improve, and therefore maintenance becomes easier.
- 4- To preserve the ideal: They are extra improvements to the programs to keep them functional for as long as feasible, lowering the expenses of using and maintaining them, increasing their speed and reliability, and adding new features.

3.1 Discussion:

In this report, I used the MVC pattern and PHP to create a POS (Point of Sale) system that can process transactions and track inventory, and I followed the following steps:

I created the database schema to store data for the POS system. This includes product, user, and transaction tables.

The models for the MVC application were then created. The models oversee interacting with the database to retrieve and store data.

Following that, I developed the views for the MVC application. The views oversee displaying the user interface and allowing the user to interact with the system.

The controllers for your MVC application were then created by me. Controllers will be in charge of managing communication between models and views, as well as handling user input and directing it to the appropriate model or view.

Finally, I tested and debugged the MVC application to ensure that it was working properly.

3.2 Conclusion

Finally, in order to effectively handle customer transactions, track sales and inventory, and provide real-time data to the administrator, HTU's store at King Abdullah Business Park requires the development of a Point of Sale (POS) system. The POS system will increase the store's efficiency, accuracy, and overall customer experience, resulting in increased profits and a stronger competitive advantage. The project's scope includes the creation of a web application that meets these requirements but excludes physical hardware or integration with external systems. The POS system will enable HTU to effectively manage its store and drive business success if the project objectives are met.

3.3 Recommendation

The following recommendations can be made based on the project's objectives:

- ❖ To provide customers with a convenient checkout experience, we can integrate various payment methods into the system, such as cash, credit and debit cards, and mobile payments.
- ❖ Put in place strong security measures to safeguard sensitive store data and prevent fraud and security breaches.
- ❖ Maintain and update the POS system on a regular basis to ensure it remains reliable and effective in meeting the needs of the store.
- ❖ Conduct user testing and collect feedback from employees and customers to continuously improve the POS system's functionality and usability.

3.4 References

- [1] M. Syaifudin, F. Fauziah, and B. Rahman, "Point of Sale Framework-Based Code Igniter and Model View Controller Using Lighthouse Testing," *J. Comput. Netw. Archit. High Perform. Comput.*, vol. 3, no. 2, pp. 202–212, 2021.
- [2] H. Hendriyanto and P. A. Cakranegara, "Web-Based Online Sales Information System Using PHP and MYSQL Database in Nara Collection," *JMKSP J. Manaj. Kepemimp. Dan Supervisi Pendidik.*, vol. 7, no. 1, pp. 35–52, 2022.
- [3] M. P. TIKAPICHART and M. C. Smith, "Factors influencing Thai SMEs to adopt point-of-sales system (POS)," 2018.
- [4] S. Bergmann, *Integrating PHP Projects with Jenkins: Continuous Integration for Robust Building and Testing*. O'Reilly Media, Inc., 2011.
- [5] P. Simajuntak, "Analisis model view controller (mvc) pada bahasa php," *J. Inf. Syst. Dev. ISD*, vol. 1, no. 2, 2016.

4 Appendix

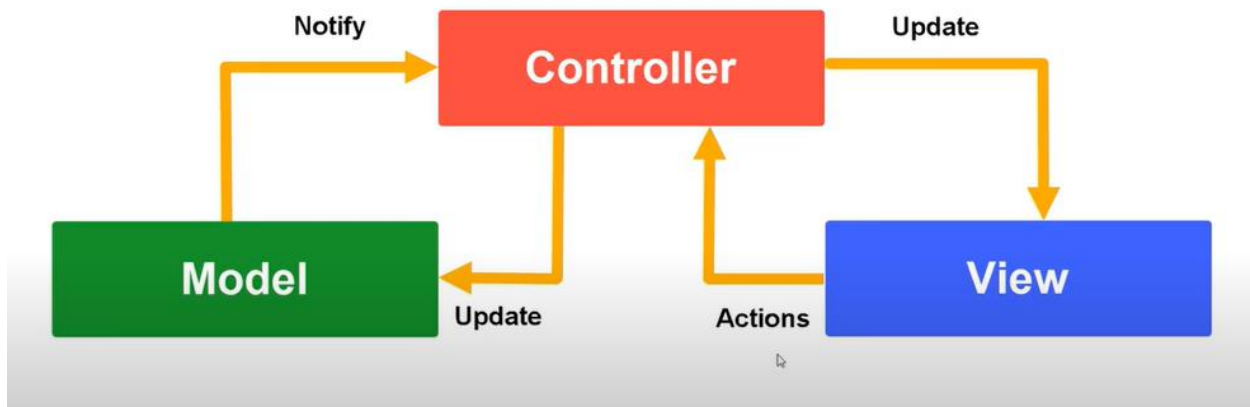


Figure (20): MVC [5]

MVC stands for Model-View-Controller, and it is a software architectural pattern that is used to develop software applications. It is designed to separate the application's data model, user interface, and control logic.

- ❖ The Model represents the data of the application and the business logic that manipulates that data. It is responsible for storing, retrieving, and manipulating the data in the application.
- ❖ The View is responsible for presenting the data to the user in a visual format, such as a web page or a graphical user interface (GUI). It is responsible for rendering the data to the user, but it does not handle any of the application's business logic or data manipulation.
- ❖ The Controller is the bridge between the Model and the View. It receives user input and requests, and then interacts with the Model to retrieve the necessary data. It then passes that data to the View to be displayed to the user.

In an MVC application, the user interacts with the View, which sends requests to the Controller. The Controller then interacts with the Model to retrieve the necessary data, and then sends that data back to the View to be presented to the user. This separation of concerns allows for easier maintenance and development of the application, as the different components can be worked on independently.

- An example of MVC tracing:

```
//=====User=====
Router::get('/users', "users.index");
Router::get('/user', "users.single");
Router::get('/user/profile', "users.profile");
Router::get('/users/create', "users.create");
Router::post('/users/store', 'users.store');
Router::get('/users/edit', 'users.edit');
Router::post('/users/update', 'users.update');
Router::get('/users/delete', 'users.delete');
//=====User=====
```

Figure (21): method to call the class Route

```
public static function get($route, $controller): void
{
    self::$get_routes[$route] = $controller;
}

public static function post($route, $controller): void
{
    self::$post_routes[$route] = $controller;
}

public static function put($route, $controller): void
{
    self::$put_routes[$route] = $controller;
}

public static function delete($route, $controller): void
{
    self::$delete_routes[$route] = $controller;
}
```

Figure (22): method to Set the Value Class Name


```

/**
 * CHECK CONNECTION DATABASE
 *
 * @return void
 */
protected function connection()
{
    $servername = "localhost";
    $username = "root";
    $password = "";
    $database = "POS";

    $this->connection = new \mysqli($servername, $username, $password, $database);

    if ($this->connection->connect_error) {
        die("Connection failed: " . $this->connection->connect_error);
    }
}

```

Figure (23): Function to Connection With DB

```

/**
 * GET ALL USERS FROM DATABASE
 *
 * @return void
 */
public function index()
{
    $this->permissions(['user:read']);
    $this->view = 'users.index';
    $user = new User; // new model users
    $this->data['users'] = $user->get_all();
    $this->data['user_count'] = count($user->get_all());
}

/**
 * GET THE USER INFORMATION AND DISPLAY IN HTML PROFILE
 *
 * @return void
 */
public function profile()
{
    $this->view = 'users.profile';
    $user = new User;
    $this->data['user'] = $user->get_by_id($_GET['id']);
}

```

Figure (24): Call Controller Function

```
/**
 * GET ALL DATA FROM DATABASE
 *
 * @return ARRAY
 */
public function get_all()
{
    $data = array();
    $stmt = $this->connection->prepare("SELECT * FROM $this->table");
    $stmt->execute();
    $result = $stmt->get_result();
    $stmt->close();
    if ($result->num_rows > 0) {
        while ($row = $result->fetch_object()) {
            $data[] = $row;
        }
    }
    return $data;
}
```

Figure (25): Call Model Function

```

/**
 * INCLUDED THE PHP HTML TEMPLATE
 *
 * @param sting $view
 * @param array $data
 * @return void
 */
class View
{
    public function __construct(string $view, array $data = array())
    {
        $view = str_replace('.', '/', $view);
        $data = (object) $data;

        $header = 'header';
        $footer = 'footer';

        if (isset($_SESSION['user']['is_admin_view'])) {
            if ($_SESSION['user']['is_admin_view']) {
                $header = 'header-admin';
                $footer = 'footer-admin';
            }
        }

        include \dirname(__DIR__, 2) . "/resources/views/partials/$header.php";
        include_once \dirname(__DIR__, 2) . "/resources/views/$view.php";
        include \dirname(__DIR__, 2) . "/resources/views/partials/$footer.php";
    }
}

```

Figure (25): Call View Function

```

index.php x
resources > views > users > index.php > div.container.my-2
1 <h1 class="text-center my-2 text-primary">Employees List(<?= $data->user_count ?>)</h1>
2
3
4 <div class="container my-2">
5     <div class="row">
6         <?php foreach ($data->users as $user) : ?>
7             <div class="col-md-6 col-lg-4 col-xl-3 mt-4 mb-2">
8                 <div class="card">
9                     <div class="card-body">
10                        <div class="d-flex">
11                            
12                            <h5 class="card-title mt-2"> &nbsp;<?= $user->display_name ?></h5>
13                        </div>
14                        <?php if ($user->active) : ?>
15                            <p class="text-success mb-1">Online <i class="fa-solid fa-power-off"></i></p>
16                        <?php else : ?>
17                            <p class="text-danger mb-1">Offline <i class="fa-solid fa-power-off"></i></p>
18                        <?php endif ?>
19                        <h6 class="card-subtitle mb-2 text-muted"><?= $user->role ?></h6>
20                        <a href="/user/profile?id=<?= $user->id ?>" class="card-link btn btn-primary">Check</a>
21                    </div>
22                </div>
23            <?php endforeach ?>
24        </div>
25    </div>

```