Even Semester (2019)



**BINUS UNIVERSITY**



**BINUS INTERNATIONAL**



**Assignment Cover Letter**

**(Group Work)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | |  |  |  |  |
| **Student Information**: | **Surname** | | |  |  | **Given Names**  **David**  **Livander**  **Muhammad**  **Zefanya** | **Student ID Number**  **2201797304**  **2201796384**  **2201797052**  **2201796870** |
| 1.  2.  3.  4. | **Amadeo**  **Surya**  **Erizky**  **Gedalya B. L. T** | |
|  |  |  |  |  |  |  |  |
| **Course Code** | **: ISYS6169** |  |  |  |  | **Course Name** | **: Database Systems** |
| **Class** | **: L3AC** |  |  |  |  | **Name of Lecturer(s)** | : **Nunung Nurul Qomariyah** |
|  |  |  |  |  |  |  |  |
| **Major** | **: CS** |  |  |  |  |  |  |
| **Title of Assignment**  (if any) | : **Zalgo Interpreter** |  |  |  |  |  |  |
| **Type of Assignment**    **Submission Pattern** | **: Final Project** | |  |  |  |  |  |
| **Due Date** | **: 03 – 12 - 2019** | |  |  |  | **Submission Date** | **: 03 – 12 - 2019** |

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer’s instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

# Plagiarism/Cheating

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

# Declaration of Originality

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student:

**II. Table of Contents**

1. **Cover**
2. **Table of content**
3. **Team roles**
4. **Problem description**
5. **Database design**
6. **Sample queries**
7. **User interface**
8. **Database Security**

**III. Team Roles**

**David Amadeo 2201797304** – UI/UX, database implementation and project documentation  
**Livander Surya 2201796384** – Processing and database creation, manipulation and implementation  
**Muhammad Erizky 2201797052** – UI/UX and database implementation  
**Zefanya Geraldya B.L.T 2201796870** – Processing and database creation, manipulation and implementation

**IV. Problem Description**

Lokalisasi Bali – founded by four med-school students from Udayana University in 2017, a coffee shop based in Bali that also advocates towards spreading the importance and hobby of coffee making and tasting. Lokalisasi Bali also ventures in making podcasts about their experience and knowledge in med school in [spotify](https://open.spotify.com/show/7cRhkgkWHRUn8xUMqTNDpV?si=z4tzq5GvTtSKnL96LI-D9w). Lokalisasi Bali, which you can also check out in [@lokalisasibali](https://www.instagram.com/lokalisasibali/), is based in Udayana University – open from 9 to 17 every Monday to Friday.

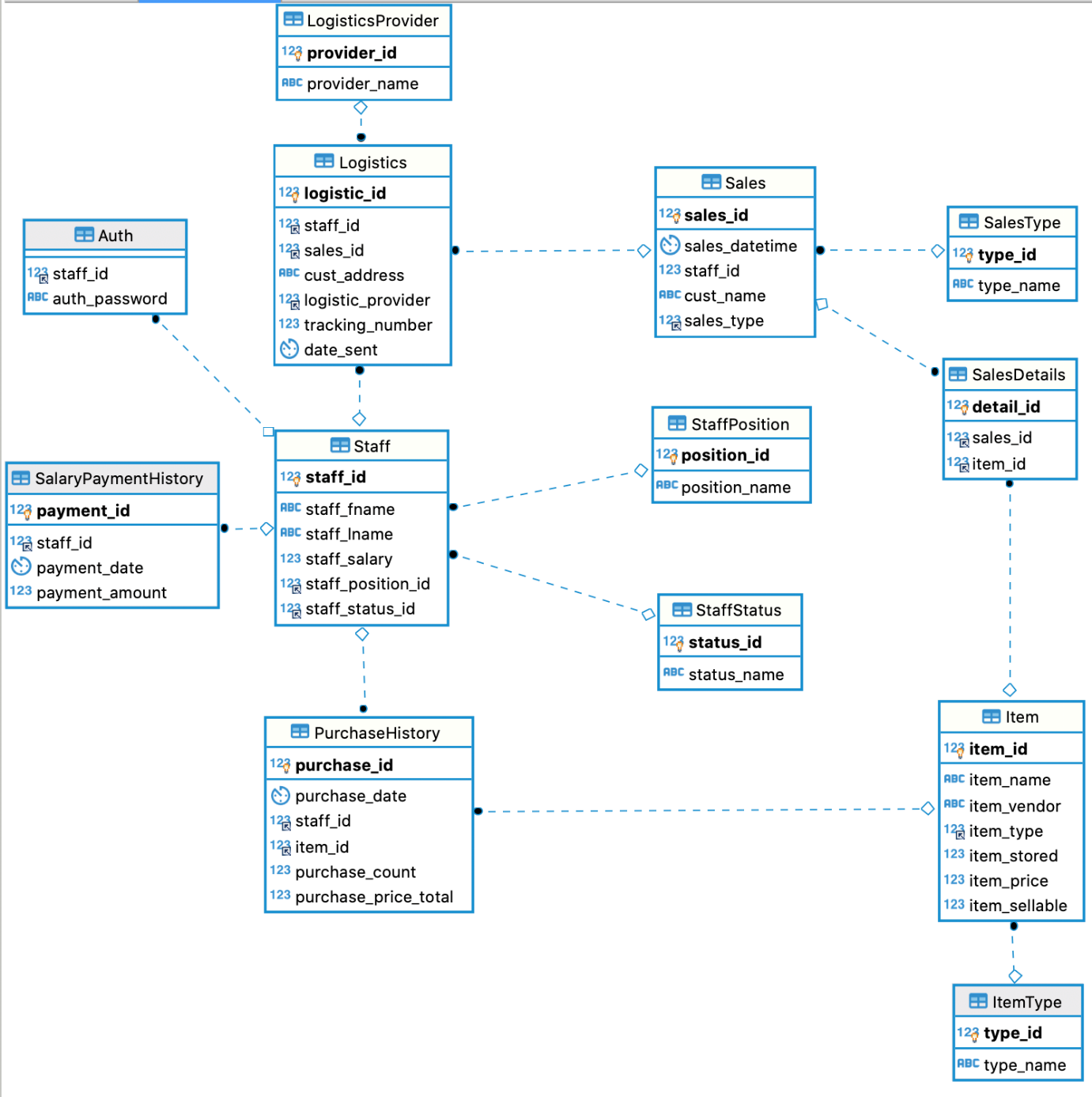
The problem at hand is that the owners of Lokalisasi, still uses pen and paper to track down all their sales, finance and logistics. All these processes that undergoes changes everyday are still being run manually. Moreover, they do not have a database system that enables all the owners to view the information that they need at the same time. Using paper and various applications such as Microsoft excel to keep track of their work is also a major problem since it is more efficient to have one application or website that they can use globally to keep track of everything they need.

The time spent in these tiresome administrative works could be used to further improve their business and their reach. We believe that with a system like ours, they could use the indeed to improve the company as whole.

So, for our final project we would like to make a database system where the admin is able to keep track on sales, finance and logistics.

The target user will primarily be for the admin. This system will make it easier for the admin to keep track of everything from sales, finance to logistics. This will minimize the need for manual administrative work as this process will not need pen and paper. This will lessen the possibility of the data lost and instead everything will be up in the database, ready to be accessed by the authorized personnel anywhere at any time.

**V. Database Design**

****

Normalization is already visualized in the ER diagram above.

Relations:

“CREATE TABLE `Auth` (

`staff\_id` int(11) unsigned NOT NULL,

`auth\_password` varchar(255) NOT NULL,

KEY `fk\_auth\_staff` (`staff\_id`),

CONSTRAINT `fk\_auth\_staff` FOREIGN KEY (`staff\_id`) REFERENCES `Staff` (`staff\_id`) ON DELETE CASCADE ON UPDATE CASCADE

)”

“CREATE TABLE `Logistics` (

`logistic\_id` int(11) unsigned NOT NULL AUTO\_INCREMENT,

`staff\_id` int(11) unsigned DEFAULT NULL,

`sales\_id` int(11) unsigned NOT NULL,

`cust\_address` varchar(255) NOT NULL,

`logistic\_provider` int(11) unsigned NOT NULL,

`tracking\_number` int(11) unsigned NOT NULL,

`date\_sent` datetime NOT NULL,

PRIMARY KEY (`logistic\_id`),

KEY `fk\_logistics\_staff` (`staff\_id`),

KEY `fk\_logistics\_sales` (`sales\_id`),

KEY `fk\_logistics\_logisticsprovider` (`logistic\_provider`),

CONSTRAINT `fk\_logistics\_logisticsprovider` FOREIGN KEY (`logistic\_provider`) REFERENCES `LogisticsProvider` (`provider\_id`) ON UPDATE CASCADE,

CONSTRAINT `fk\_logistics\_sales` FOREIGN KEY (`sales\_id`) REFERENCES `Sales` (`sales\_id`) ON UPDATE CASCADE,

CONSTRAINT `fk\_logistics\_staff` FOREIGN KEY (`staff\_id`) REFERENCES `Staff` (`staff\_id`) ON DELETE SET NULL ON UPDATE CASCADE

)”

“CREATE TABLE `ItemType` (

`type\_id` int(11) unsigned NOT NULL AUTO\_INCREMENT,

`type\_name` varchar(100) NOT NULL,

PRIMARY KEY (`type\_id`)

)”

“CREATE TABLE `Logistics` (

`logistic\_id` int(11) unsigned NOT NULL AUTO\_INCREMENT,

`staff\_id` int(11) unsigned DEFAULT NULL,

`sales\_id` int(11) unsigned NOT NULL,

`cust\_address` varchar(255) NOT NULL,

`logistic\_provider` int(11) unsigned NOT NULL,

`tracking\_number` int(11) unsigned NOT NULL,

`date\_sent` datetime NOT NULL,

PRIMARY KEY (`logistic\_id`),

KEY `fk\_logistics\_staff` (`staff\_id`),

KEY `fk\_logistics\_sales` (`sales\_id`),

KEY `fk\_logistics\_logisticsprovider` (`logistic\_provider`),

CONSTRAINT `fk\_logistics\_logisticsprovider` FOREIGN KEY (`logistic\_provider`) REFERENCES `LogisticsProvider` (`provider\_id`) ON UPDATE CASCADE,

CONSTRAINT `fk\_logistics\_sales` FOREIGN KEY (`sales\_id`) REFERENCES `Sales` (`sales\_id`) ON UPDATE CASCADE,

CONSTRAINT `fk\_logistics\_staff` FOREIGN KEY (`staff\_id`) REFERENCES `Staff` (`staff\_id`) ON DELETE SET NULL ON UPDATE CASCADE

)”

“CREATE TABLE `LogisticsProvider` (

`provider\_id` int(11) unsigned NOT NULL AUTO\_INCREMENT,

`provider\_name` varchar(100) NOT NULL,

PRIMARY KEY (`provider\_id`)

)”

“CREATE TABLE `PurchaseHistory` (

`purchase\_id` int(11) unsigned NOT NULL AUTO\_INCREMENT,

`purchase\_date` datetime NOT NULL,

`staff\_id` int(11) unsigned DEFAULT NULL,

`item\_id` int(11) unsigned DEFAULT NULL,

`purchase\_count` int(11) unsigned NOT NULL,

`purchase\_price\_total` int(11) unsigned NOT NULL,

PRIMARY KEY (`purchase\_id`),

KEY `fk\_purchasehistory\_staff` (`staff\_id`),

KEY `fk\_purchasehistory\_item` (`item\_id`),

CONSTRAINT `fk\_purchasehistory\_item` FOREIGN KEY (`item\_id`) REFERENCES `Item` (`item\_id`) ON DELETE SET NULL ON UPDATE CASCADE,

CONSTRAINT `fk\_purchasehistory\_staff` FOREIGN KEY (`staff\_id`) REFERENCES `Staff` (`staff\_id`) ON DELETE SET NULL ON UPDATE CASCADE

)”

“CREATE TABLE `SalaryPaymentHistory` (

`payment\_id` int(11) unsigned NOT NULL AUTO\_INCREMENT,

`staff\_id` int(11) unsigned DEFAULT NULL,

`payment\_date` datetime NOT NULL,

`payment\_amount` int(11) unsigned NOT NULL,

PRIMARY KEY (`payment\_id`),

KEY `fk\_salarypaymenthistory` (`staff\_id`),

CONSTRAINT `fk\_salarypaymenthistory` FOREIGN KEY (`staff\_id`) REFERENCES `Staff` (`staff\_id`) ON DELETE SET NULL ON UPDATE CASCADE

)”

“Sales | CREATE TABLE `Sales` (

`sales\_id` int(11) unsigned NOT NULL AUTO\_INCREMENT,

`sales\_datetime` datetime NOT NULL,

`staff\_id` int(11) unsigned DEFAULT NULL,

`cust\_name` varchar(100) NOT NULL,

`sales\_type` int(11) unsigned NOT NULL,

PRIMARY KEY (`sales\_id`),

KEY `fk\_sales\_type` (`sales\_type`),

CONSTRAINT `fk\_sales\_type` FOREIGN KEY (`sales\_type`) REFERENCES `SalesType` (`type\_id`) ON UPDATE CASCADE

)”

“CREATE TABLE `SalesDetails` (

`detail\_id` int(11) unsigned NOT NULL AUTO\_INCREMENT,

`sales\_id` int(11) unsigned NOT NULL,

`item\_id` int(11) unsigned DEFAULT NULL,

PRIMARY KEY (`detail\_id`),

KEY `fk\_salesdetails\_sales` (`sales\_id`),

KEY `fk\_salesdetails\_item` (`item\_id`),

CONSTRAINT `fk\_salesdetails\_item` FOREIGN KEY (`item\_id`) REFERENCES `Item` (`item\_id`) ON DELETE SET NULL ON UPDATE CASCADE,

CONSTRAINT `fk\_salesdetails\_sales` FOREIGN KEY (`sales\_id`) REFERENCES `Sales` (`sales\_id`)

)”

“CREATE TABLE `Staff` (

`staff\_id` int(11) unsigned NOT NULL AUTO\_INCREMENT,

`staff\_fname` varchar(100) NOT NULL,

`staff\_lname` varchar(100) NOT NULL,

`staff\_salary` int(11) NOT NULL,

`staff\_position\_id` int(11) unsigned NOT NULL,

`staff\_status\_id` int(11) unsigned NOT NULL,

PRIMARY KEY (`staff\_id`),

KEY `fk\_staff\_staffposition` (`staff\_position\_id`),

KEY `fk\_staff\_status` (`staff\_status\_id`),

CONSTRAINT `fk\_staff\_staffposition` FOREIGN KEY (`staff\_position\_id`) REFERENCES `StaffPosition` (`position\_id`) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT `fk\_staff\_status` FOREIGN KEY (`staff\_status\_id`) REFERENCES `StaffStatus` (`status\_id`) ON UPDATE CASCADE

)”

“CREATE TABLE `StaffPosition` (

`position\_id` int(11) unsigned NOT NULL AUTO\_INCREMENT,

`position\_name` varchar(100) NOT NULL,

PRIMARY KEY (`position\_id`)

)”

“CREATE TABLE `StaffStatus` (

`status\_id` int(11) unsigned NOT NULL AUTO\_INCREMENT,

`status\_name` varchar(100) NOT NULL,

PRIMARY KEY (`status\_id`)

)”

**VI. Sample Queries**

To get data for the staff table, we use this query:

“*SELECT s.staff\_id, s.staff\_fname, s.staff\_lname, s.staff\_salary, p.position\_name, t.status\_name FROM Staff s LEFT JOIN StaffPosition p ON s.staff\_position\_id = p.position\_id LEFT JOIN StaffStatus t ON s.staff\_status\_id = t.status\_id;*”

To get data for the inventory table, we use this query:

“*SELECT item\_id, item\_name, item\_vendor, t.type\_name, item\_stored, item\_price, item\_sellable FROM Item LEFT JOIN ItemType t ON item\_type = t.type\_id*”

To get data for the purchase history table, we use this query:

“*SELECT p.purchase\_id, p.purchase\_date, p.staff\_id, s.staff\_fname, s.staff\_lname, i.item\_name, t.type\_name, p.purchase\_count, p.purchase\_price\_total FROM PurchaseHistory p LEFT JOIN Staff s ON p.staff\_id = s.staff\_id LEFT JOIN Item i on p.item\_id = i.item\_id LEFT JOIN ItemType t ON i.item\_type = t.type\_id;*”

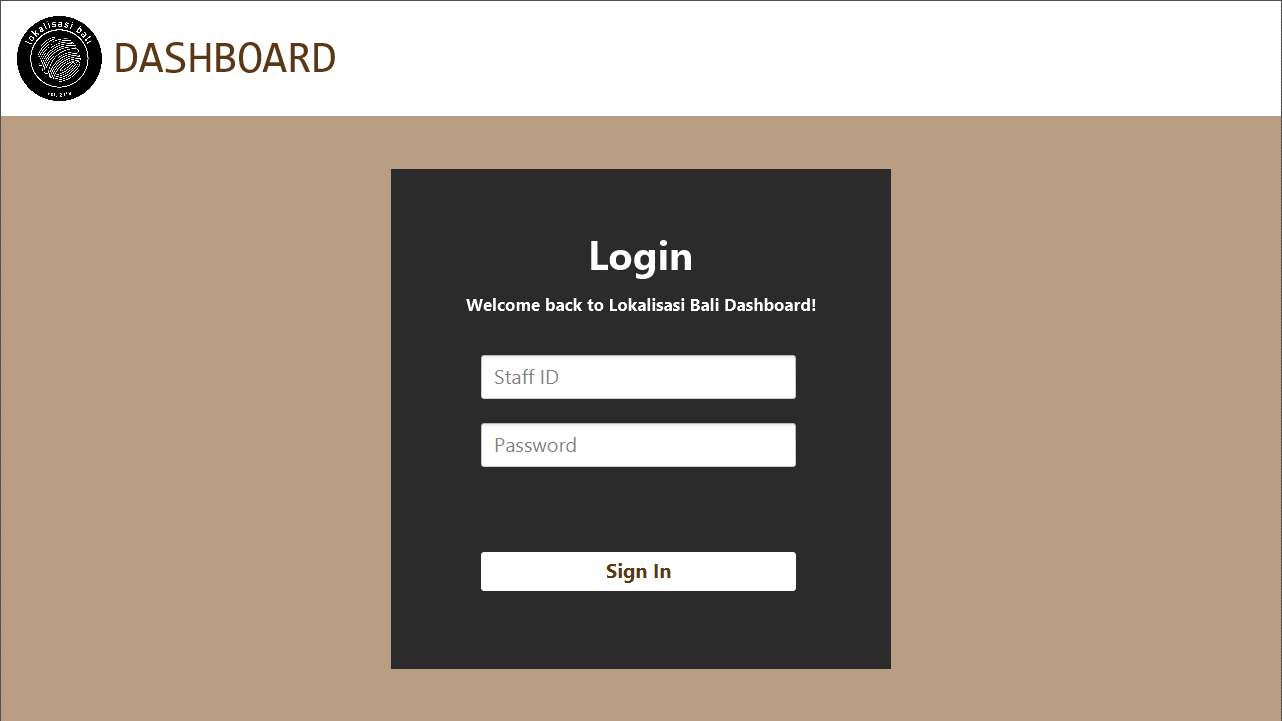
To get data for the sales table, we use this query:

“*SELECT s.sales\_id, s.sales\_datetime, s.staff\_id, s.cust\_name, i.staff\_fname, i.staff\_lname, t.type\_name FROM Sales s LEFT JOIN Staff i ON s.staff\_id = i.staff\_id LEFT JOIN SalesType t ON s.sales\_type = t.type\_id;*”

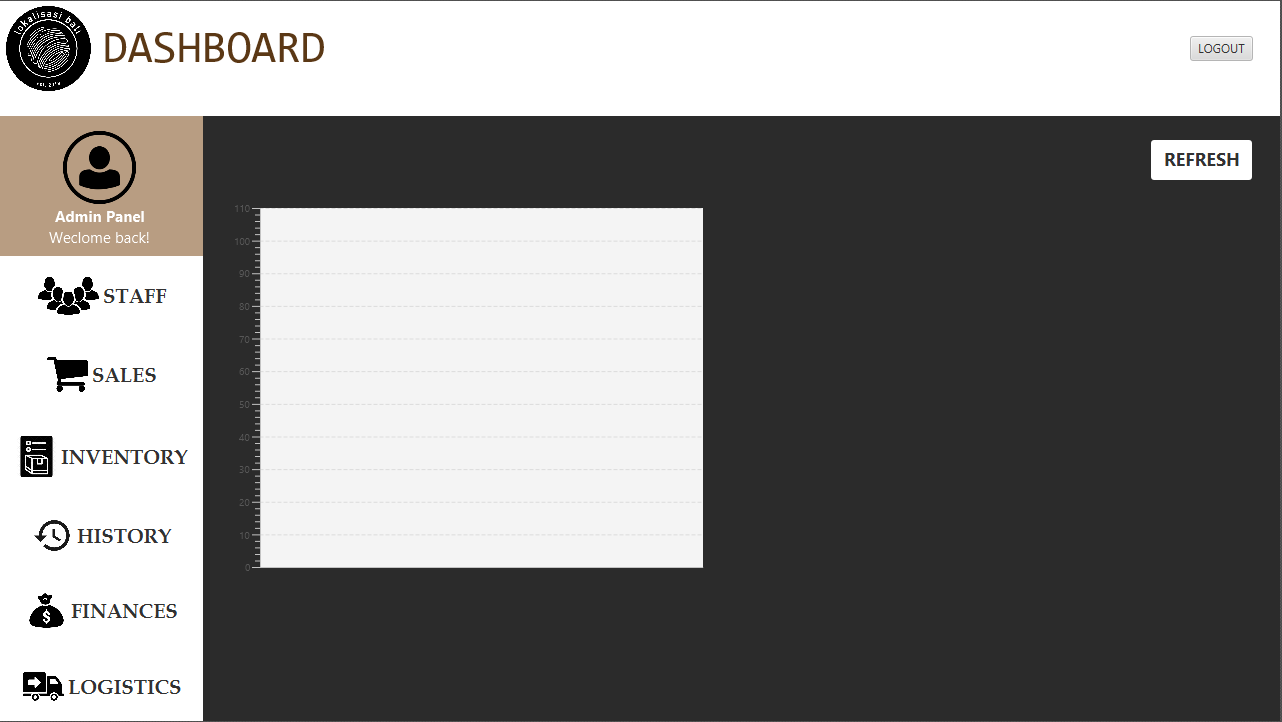
To get data for the logistics table, we use this query:

“*SELECT l.logistic\_id, l.staff\_id, s.staff\_fname, s.staff\_lname, l.cust\_address, p.provider\_name, l.tracking\_number, l.date\_sent FROM Logistics l LEFT JOIN Staff s ON l.staff\_id = s.staff\_id LEFT JOIN LogisticsProvider p ON l.logistic\_provider = p.provider\_id;”*

**VII. User Interface Design**

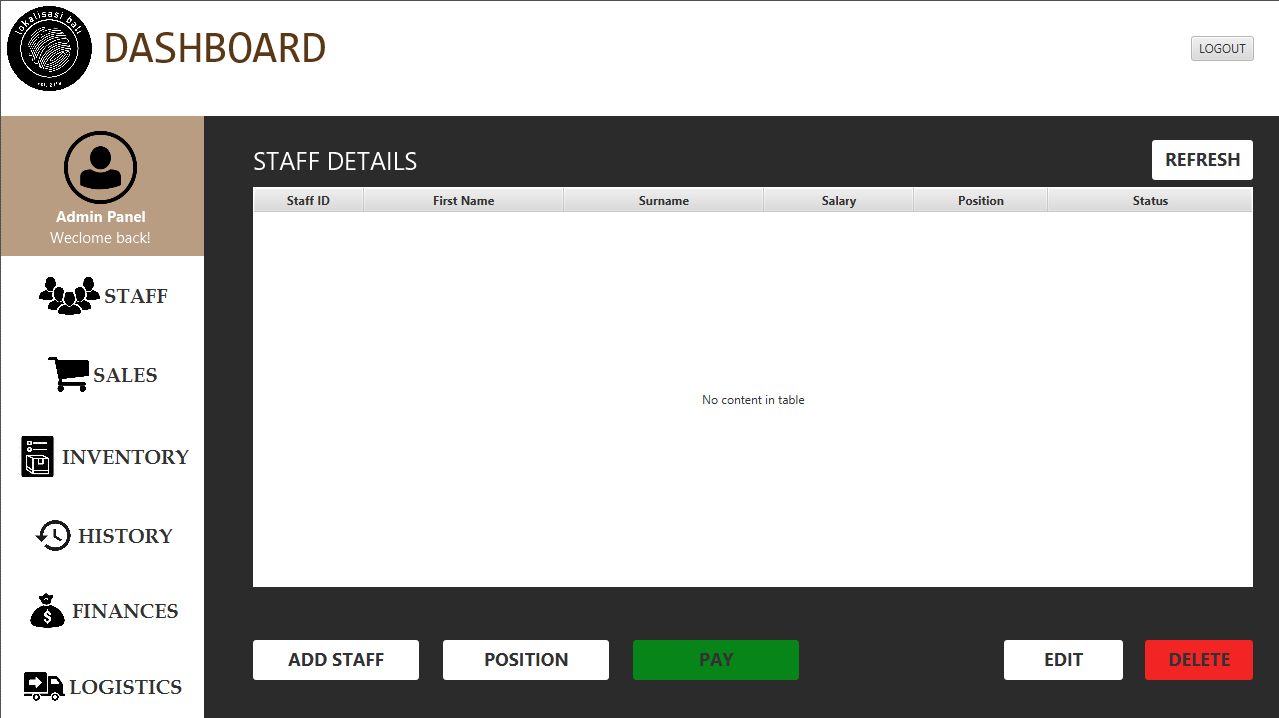
****

This is the login page of the programme. For the authentication, we use the query 🡪 *SELECT \* FROM Auth WHERE staff\_id = ? AND auth\_password = ?;*



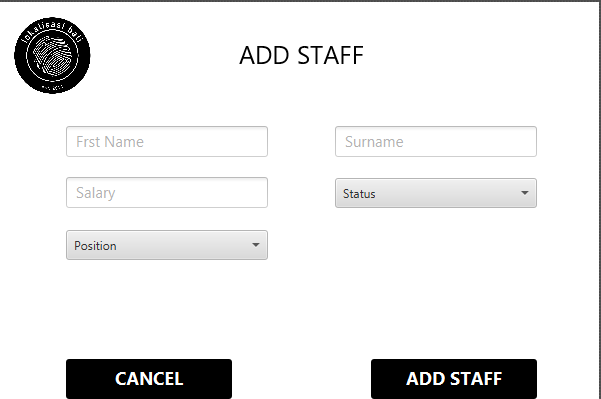
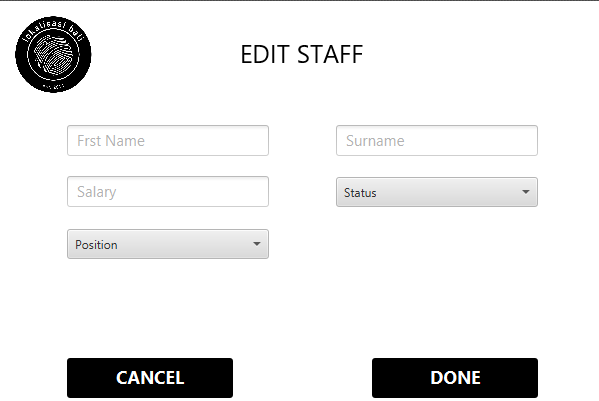
This is the home page. After the user have logged in, this page will appear. There will be displayed a chart containing sales trajectory details.

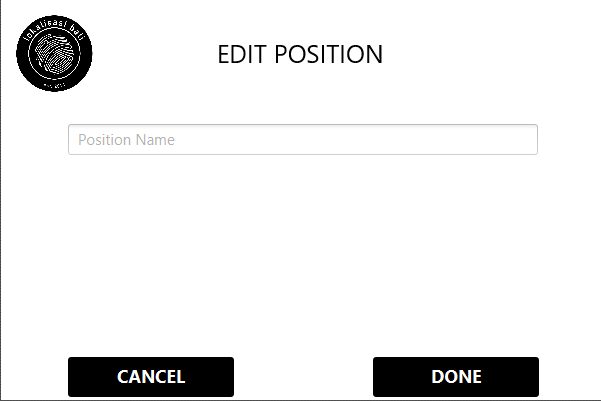
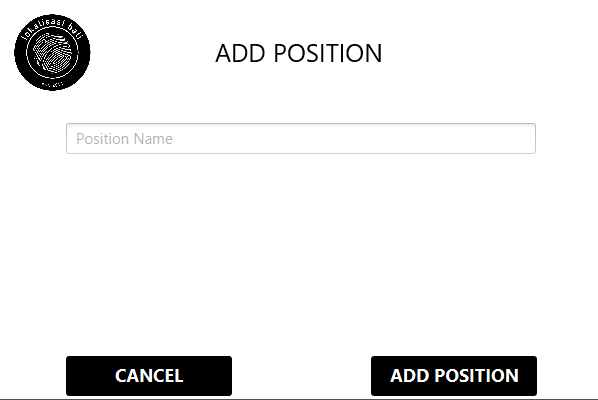
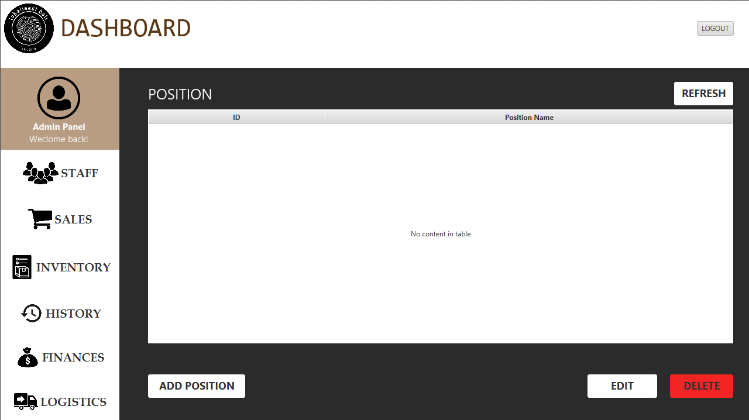
As the home page, the user can choose between features one the left side of the screen. Between staff, sales, inventory, history, finances and logistics.



This page for staff details uses the table 🡪 [*Staff*], [*StaffPosition*], [*StaffStatus*]. Query 🡪 *SELECT s.staff\_id, s.staff\_fname, s.staff\_lname, s.staff\_salary, p.position\_name, t.status\_name FROM Staff s LEFT JOIN StaffPosition p ON s.staff\_position\_id = p.position\_id LEFT JOIN StaffStatus t ON s.staff\_status\_id = t.status\_id;*

There are 3 buttons available on this page to manipulate the tables mentioned. There is a button to add a staff, to edit a staff details and delete a staff.





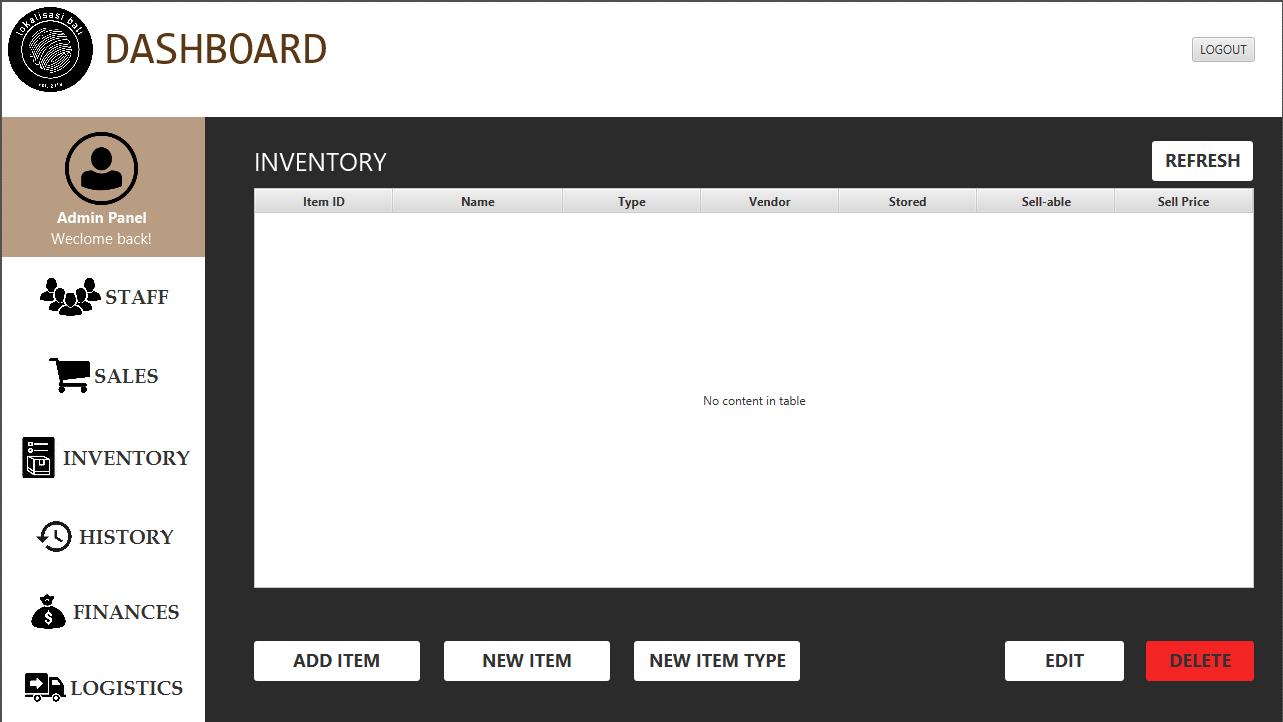
Add staff Query 🡪 *INSERT INTO Staff (staff\_fname, staff\_lname, staff\_salary, staff\_position\_id, staff\_status\_id) VALUES (?, ?, ?, ?, ?)*

Delete staff Query 🡪 *DELETE FROM Staff WHERE staff\_id = ?;*

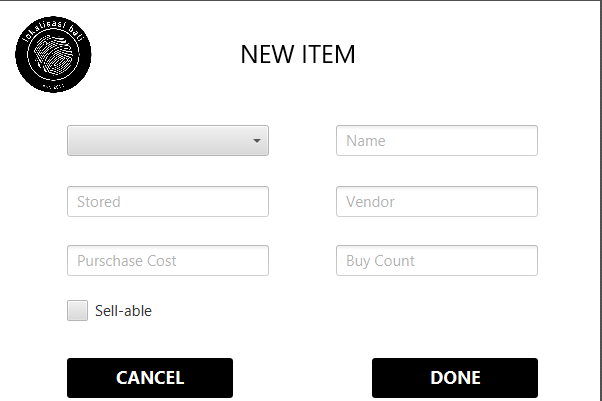
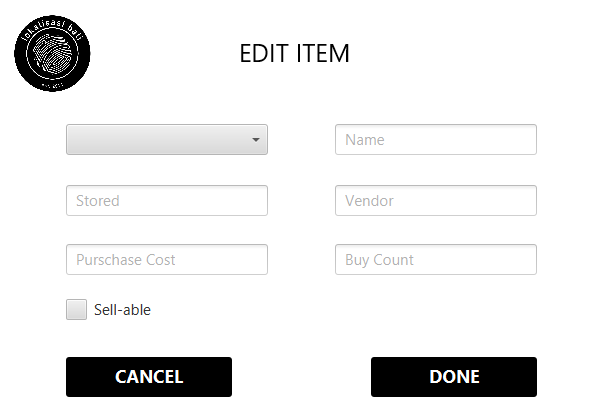
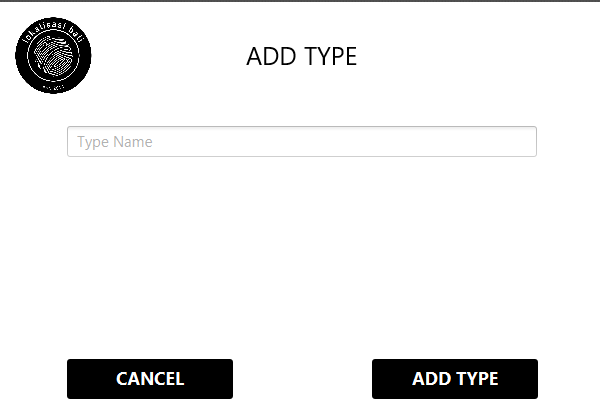
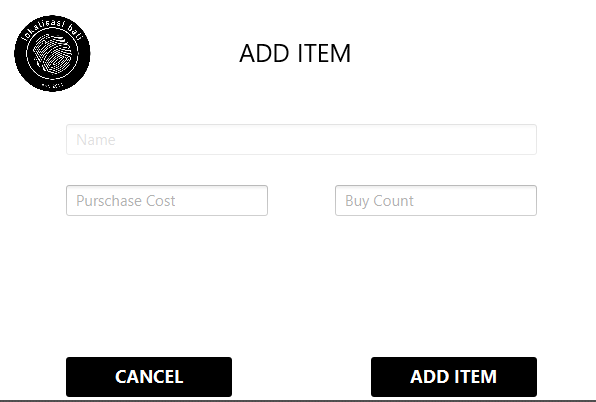
Edit staff Query 🡪 *UPDATE Staff SET staff\_fname = ?, staff\_lname = ?, staff\_salary = ?, staff\_position\_id = ?, staff\_status\_id = ? WHERE staff\_id = ?;*

Add position Query 🡪 *INSERT INTO StaffPosition (position\_name) VALUES (?);*

Edit position Query 🡪 *UPDATE StaffPosition SET position\_name = ? WHERE position\_id = ?;*



This is the inventory page. This page uses the table 🡪 [*Item*], [*ItemType*] and uses the query 🡪 *SELECT item\_id, item\_name, item\_vendor, t.type\_name, item\_stored, item\_price, item\_sellable FROM Item LEFT JOIN ItemType t ON item\_type = t.type\_id*



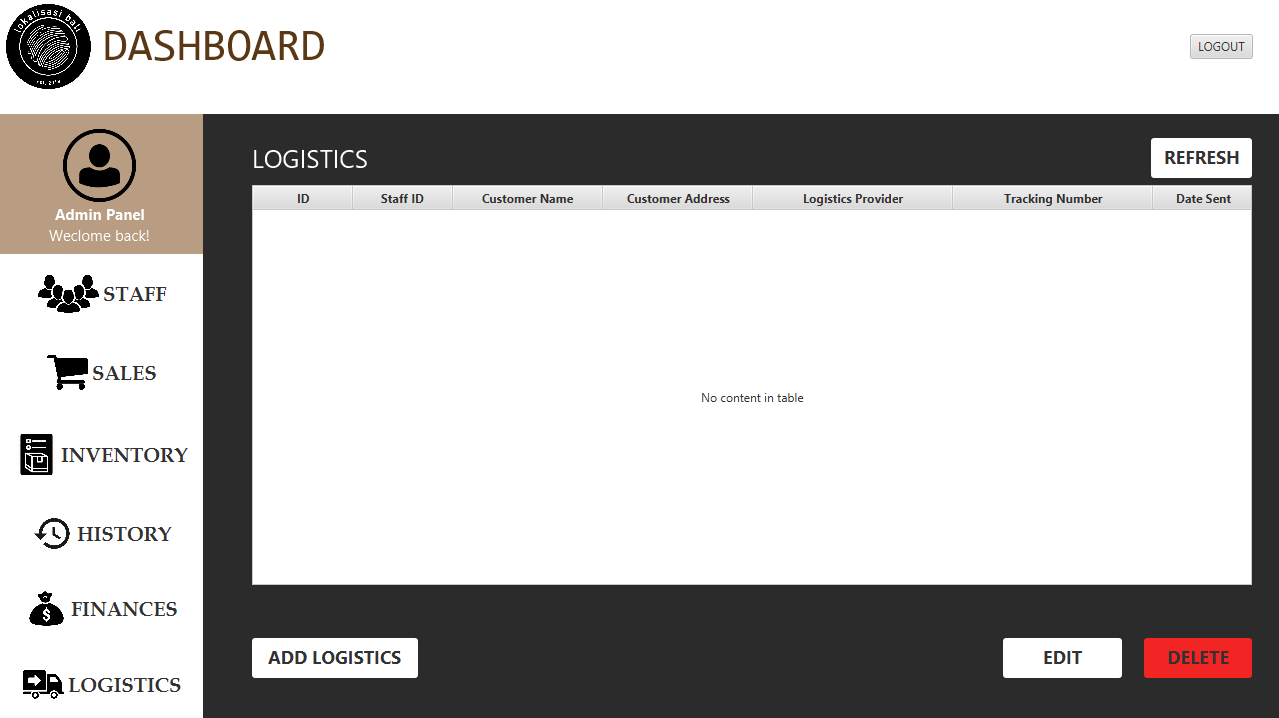
Add item Query 🡪 *UPDATE Item SET item\_stored = item\_stored + ? WHERE item\_id = ?  
PurchaseHistory (purchase\_date, staff\_id, item\_id, purchase\_count, purchase\_price\_total) VALUES (NOW(), ?, ?, ?, ?)*

Edit item Query 🡪 *UPDATE Item SET item\_name = ?, item\_vendor = ?, item\_type = ?, item\_stored = ?, item\_price = ?, item\_sellable = ? WHERE item\_id = ?*

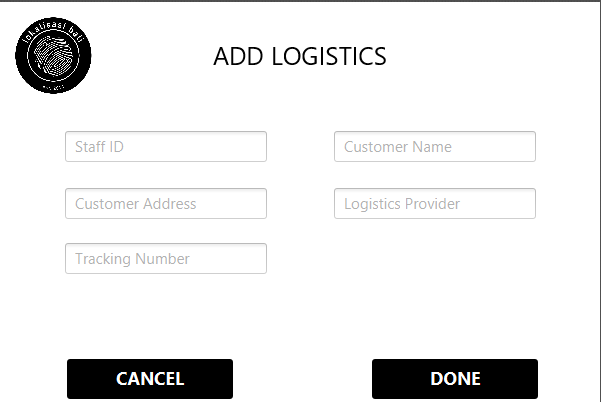
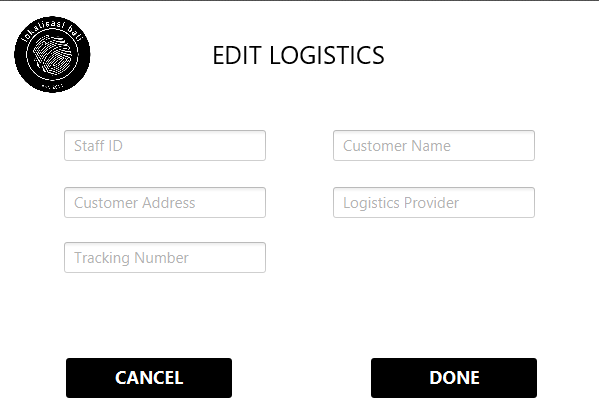
New item Query 🡪 *INSERT INTO Item (item\_name, item\_vendor, item\_type, item\_stored, item\_price, item\_sellable) VALUES (?, ?, ?, ?, ?, ?);  
INSERT INTO PurchaseHistory (purchase\_date, staff\_id, item\_id, purchase\_count, purchase\_price\_total) VALUES (NOW(), ?, ?, ?, ?)*

Delete item Query 🡪 *DELETE FROM Item WHERE item\_id = ?*

Add item type Query 🡪 *INSERT INTO ItemType (type\_name) VALUES (?)*



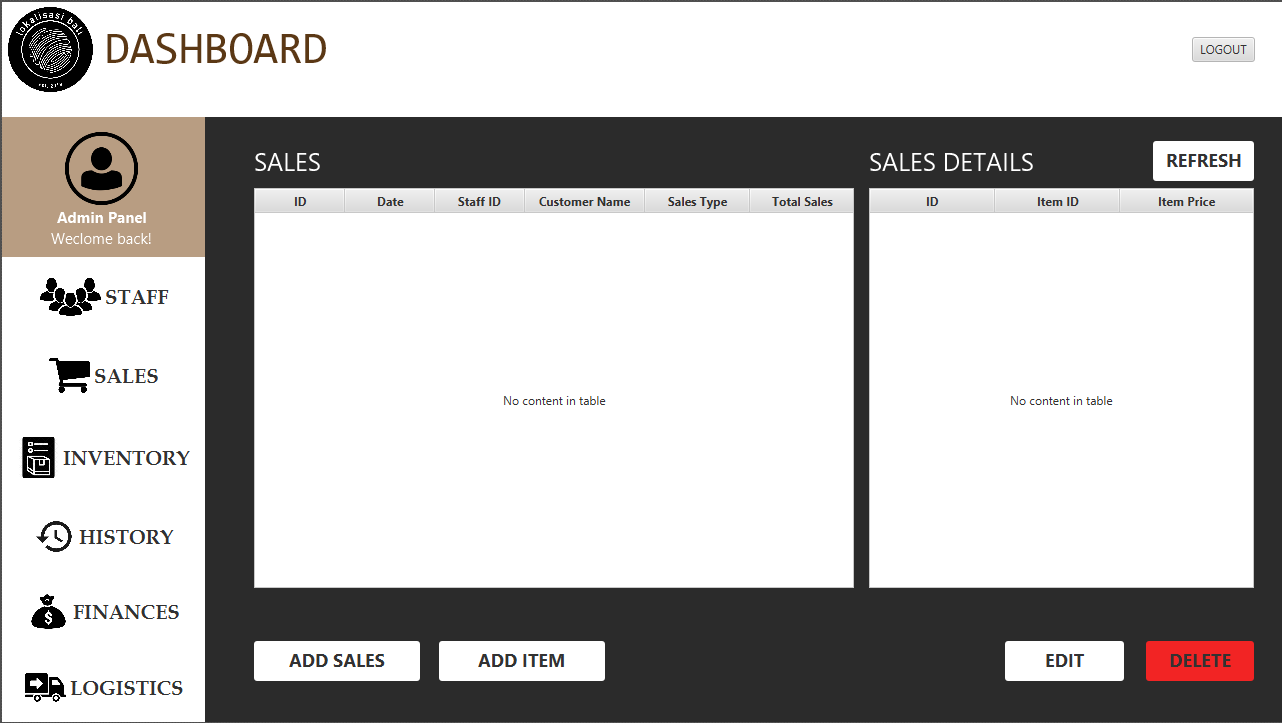
This is the logistics page. This page uses the table 🡪 [*Logistics*], [*LogisticsProvider*] and uses the query 🡪 *SELECT item\_id, item\_name, item\_vendor, t.type\_name, item\_stored, item\_price, item\_sellable FROM Item LEFT JOIN ItemType t ON item\_type = t.type\_id*



Add logistics Query 🡪 *INSERT INTO Logistics (staff\_id, sales\_id, cust\_address, logistics\_provider, tracking\_number, date\_sent) VALUES (?, ?, ?, ?, ?, NOW());*

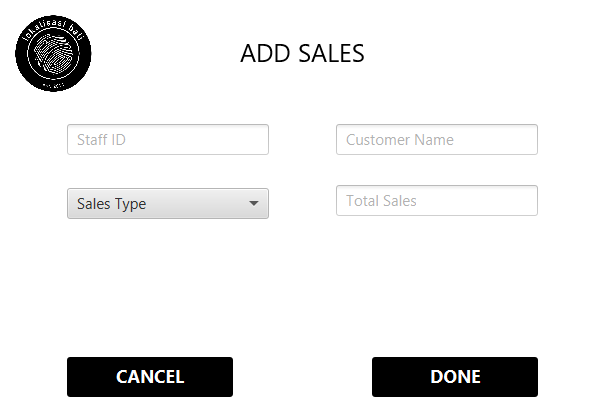
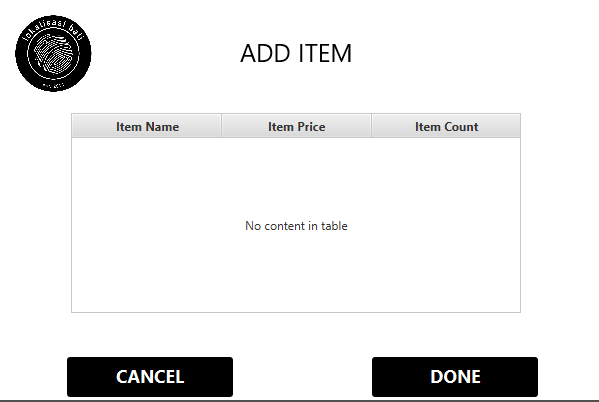
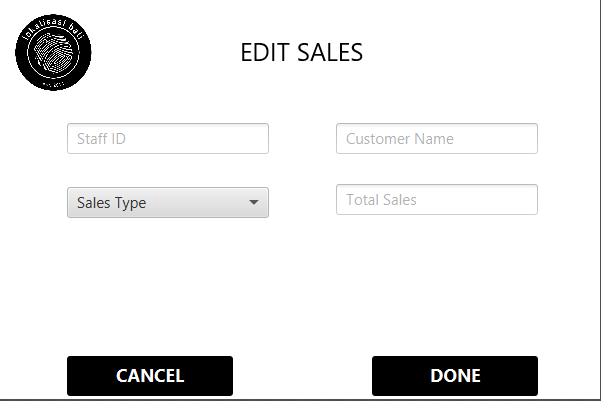
Edit logistics Query 🡪 *UPDATE Logistics SET staff\_id = ?, sales\_id = ?, cust\_address = ?, logistics\_provider = ?, tracking\_number = ?, date\_sent = ? WHERE logistics\_id = ?;*

Delete logistics Query 🡪 *DELETE FROM Logistics WHERE logistics\_id = ?*



This is the sales page. This page uses the table 🡪 [*Sales*], [*SalesDetails*] and uses the query 🡪 *SELECT s.sales\_id, s.sales\_datetime, s.staff\_id, s.cust\_name, i.staff\_fname, i.staff\_lname, t.type\_name FROM Sales s LEFT JOIN Staff i ON s.staff\_id = i.staff\_id LEFT JOIN SalesType t ON s.sales\_type = t.type\_id;*

*SELECT d.item\_id, i.item\_name, i.item\_price FROM SalesDetails d LEFT JOIN Item ON d.item\_id = i.item\_id WHERE d.sales\_id = ?*

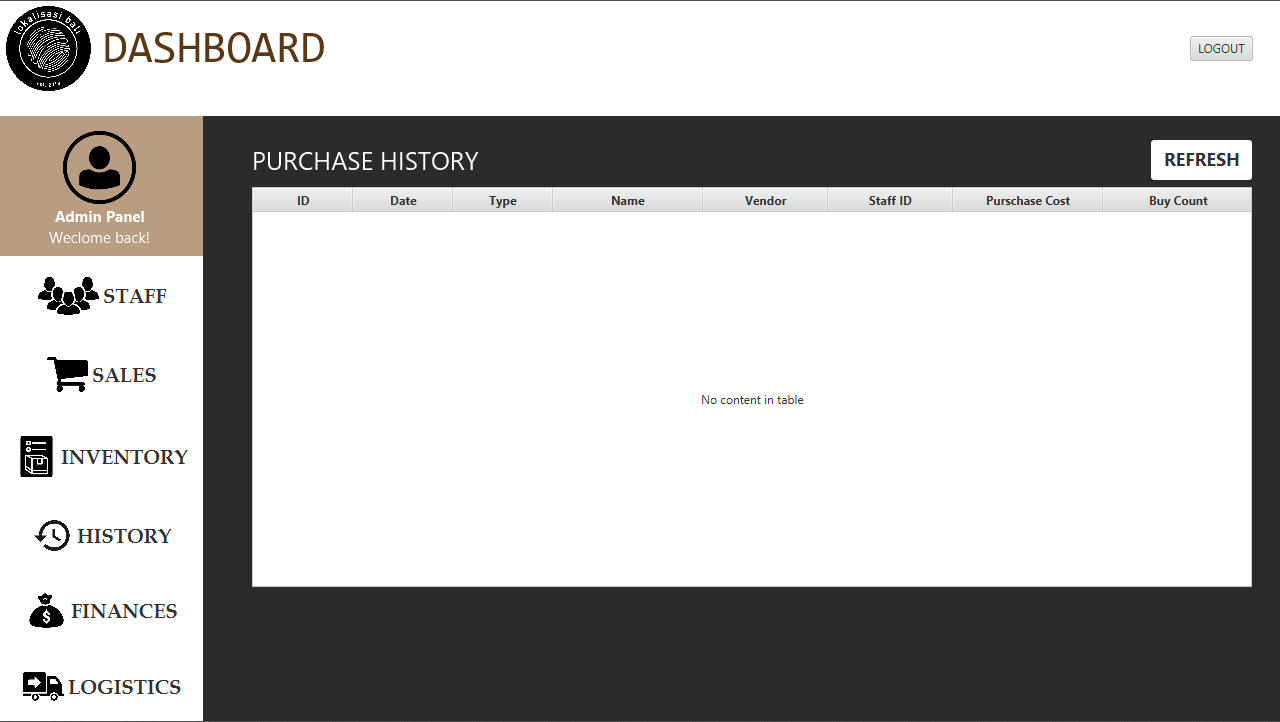


Edit sales Query 🡪 *UPDATE Sales SET sales\_datetime = ?, staff\_id = ?, cust\_name = ?, sales\_type = ?;*

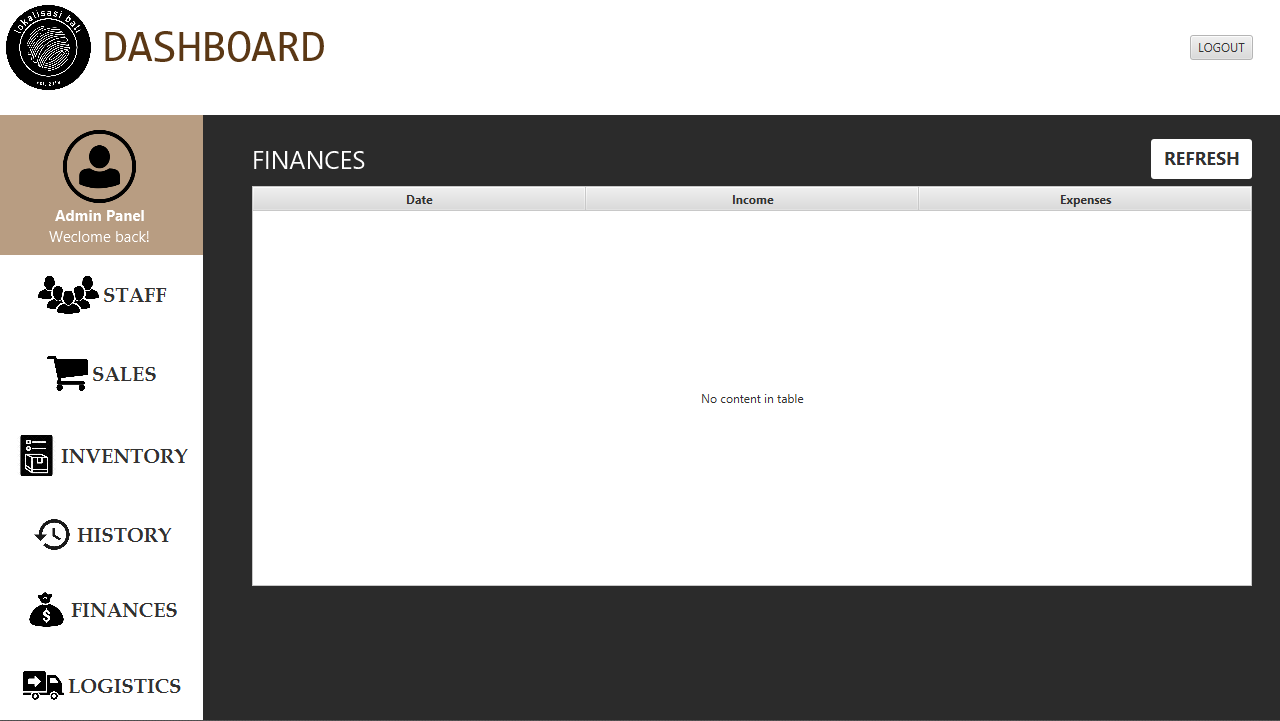
Add sales Query 🡪 *INSERT INTO Sales (sales,datetime, staff\_id, cust\_name, sales\_type) VALUES (NOW(), ?, ?, ?);*

Add item Query 🡪 *INSERT INTO SalesDetails (sales\_id, item\_id) VALUES (?, ?);*

Delete sales Query 🡪 *DELETE FROM Sales WHERE sales\_id = ?*



This is the history page. The table used are 🡺 [*PurchaseHistory*], [*Item*], [*Staff*] and use the query 🡪 *SELECT p.purchase\_id, p.purchase\_date, p.staff\_id, s.staff\_fname, s.staff\_lname, i.item\_name, t.type\_name, p.purchase\_count, p.purchase\_price\_total FROM PurchaseHistory p LEFT JOIN Staff s ON p.staff\_id = s.staff\_id LEFT JOIN Item i on p.item\_id = i.item\_id LEFT JOIN ItemType t ON i.item\_type = t.type\_id;*”



This is the finance page. The table used are 🡺 [*PurchaseHistory*] and use the query 🡪 *SELECT DATE(purchase\_date), SUM(purchase\_price\_total) as sum FROM PurchaseHistory GROUP BY DATE(purchase\_date)*

**VIII. Database Security**

Before the user can access all the features, they have to login. Each user has their own unique staff ID and password. To login, they should enter their staff ID along with their password. If they cannot provide the following, they will not be able to access the programme. For this programme, there are operations, cashier, barista, sales and social media as users. Only those in sales and operations can access all the features of the programme.