VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

DATABASE SYSTEMS (LAB) (CO2014)

Class: CC03 | Group: 2

ASSIGNMENT 2
FABRIC AGENCY DATABASE

Lecturer:	Phan Trọng Nhân, PhD
Students:	Lê Hoàng Duy - 2152040
	Nguyễn Khương Duy - 1952615
	Trần Nhật Duy - 2153258
	Nguyễn Hồ Tiến Đạt - 2152503
	Nguyễn Thịnh Đạt - 2152507

HO CHI MINH CITY, DECEMBER 2023

Contents

1 List of member & workload

ID	**Student ID**	**Full Name**	**Workload**	**Commitment**	**Note**
1	2152040	Lê Hoàng Duy	Physical Design	100%	Leader
2	1952615	Nguyễn Khương Duy		0%	
3	2153258	Trần Nhật Duy	Building Applications Security	100%	
4	2152503	Nguyễn Hồ Tiến Đạt	Data review Indexing Trigger	100%	
5	2152507	Nguyễn Thịnh Đạt	Store Procedure, Function	100%	

2 Recover from Assignment 1

After receiving some feedback from our instructor, we have rechecked the EER diagram and the mapping to make them more logical and meet the requirements better.

The new EER diagram is shown in Figure 1 and can be accessed via link EER Diagram to have a better view.

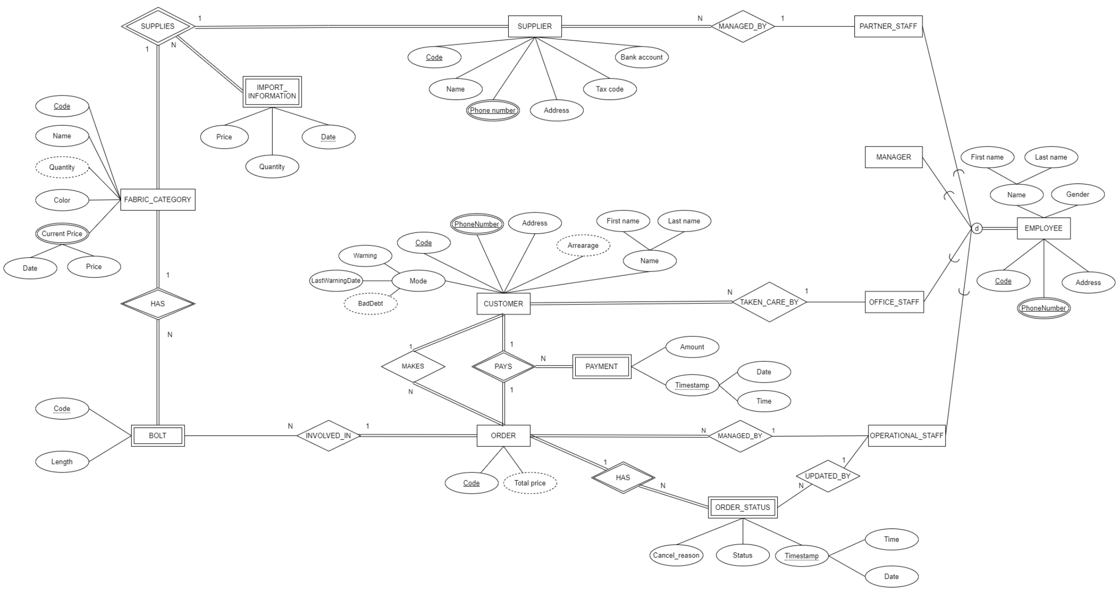


Figure 1: New EER diagram

The new relational mapping is shown in Figure 2 and can be accessed via link Mapping (tab NEW_MAPPING) for better view.

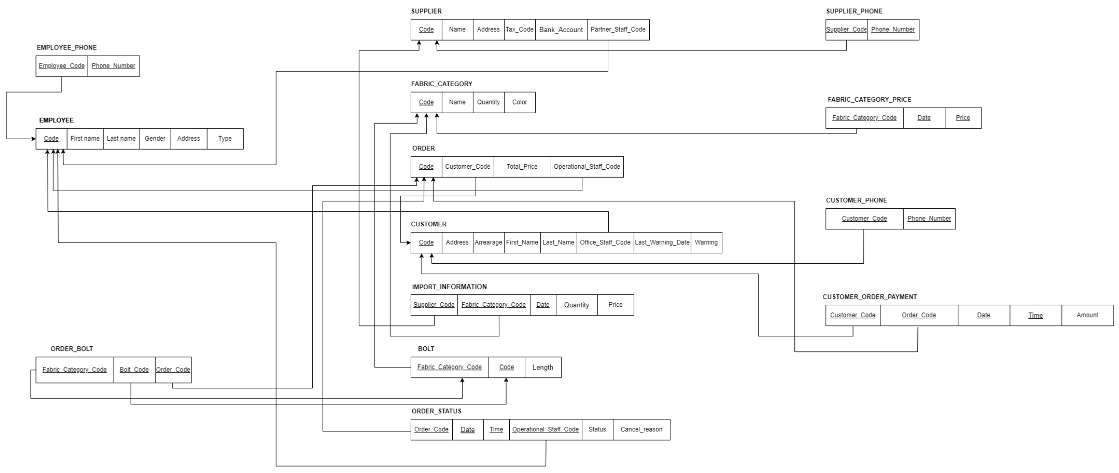


Figure 2: New relational mapping

3 Physical database design

3.1 Implementing the database

In this assignment, we use **SQL Server** as the Database Management System. Here are our choices of data type, data length and the constraints for each table:

EMPLOYEE:

- `E_Code`: string of 4 characters.
- `E_FName`: string of up to 20 characters.
- `E_LName`: string of up to 20 characters.
- `E_Address`: string of up to 50 characters.
- `E_Gender`: contains one of the characters `'F'` (for female) and `'M'` (for male).
- `E_Type`: contains one of the strings: `'Manager'`, `'Operational staff'`, `'Office staff'`, `'Partner staff'`.
- **Primary key:** `E_Code`.
- **Fields to be set `NOT NULL`:** `E_FName`, `E_LName`, `E_Type`.

EMPLOYEE_PHONE:

- `Phone_number`: string of up to 15 digits.
- **Primary key:** (`Employee_Code`, `Phone_number`).
- **Foreign key:**
  - `Employee_Code` references `E_Code` in the table `Employee`.
- **Fields to be set `UNIQUE`:** `Phone_number`.

SUPPLIER:

- `S_Code`: string of 6 characters.
- `S_Name`: string of up to 20 characters.
- `S_Address`: string of up to 50 characters.

- S_Taxcode: string of up to 15 characters.
- S_BankAccount: string of up to 20 characters.
- **Primary key:** S_Code.
- **Foreign key:**
  - S_Pstaff_code references E_Code of type 'Partner staff' in the table Employee.
- **Fields to be set NOT NULL:** S_Name, S_Pstaff_code.

FABRIC_CATEGORY:

- F_Code: string of 5 characters.
- F_Name: string of up to 20 characters.
- F_Quantity: non-negative integer, default as 0.
- F_Color: hexadecimal code of a color adhering to the regular expression: `\^#([A-Fa-f0-9]{6}|[A-Fa-f0-9]{3})$\`, for example #A8B4DD or #C73.
- **Primary key:** F_Code.
- **Fields to be set NOT NULL:** F_Name.

CUSTOMER:

- C_Code: string of 8 characters.
- C_FName: string of up to 20 characters.
- C_LName: string of up to 20 characters.
- C_Address: string of up to 50 characters.
- C_Arrearage: non-negative number with precision 10, scale 2.
- C_Last_Warning_Date: string that adheres to ISO 8601 Date standard.
- C_Warning: contains one of the characters '0' (not in warning mode) and '1' (in warning mode).
- **Primary key:** C_Code.

- **Foreign key:**
  - C_Office_staff_code references E_Code of type 'Office staff' in the table Employee.
- **Fields to be set NOT NULL:** C_FName, C_LName, C_Office_staff_code.

ORDER_CUSTOMER:

- O_Code: string of 10 characters.
- O_Total_Price: non-negative number with precision 10, scale 2; default as 0.
- **Primary key:** O_Code.
- **Foreign key:**
  - O_Customer_Code references C_Code in the table Customer.
  - O_Ostaff_code references E_Code of type 'Operational staff' in the table Employee.
- **Fields to be set NOT NULL:** O_Customer_Code, O_Ostaff_code.

IMPORT_INFORMATION:

- I_Date: string that adheres to ISO 8601 Date standard.
- I_Quantity: positive integer.
- I_Price: positive number with precision 10, scale 2.
- **Primary key:** (I_Supplier_Code, I_Category_Code, I_Date).
- **Foreign key:**
  - I_Supplier_Code references S_Code in the table Supplier.
  - I_Category_Code references F_Code in the table Fabric_Category.
- **Fields to be set NOT NULL:** I_Quantity, I_Price.

BOLT:

- B_Code: string of 7 characters.
- B_Length: positive number with precision 5, scale 2.
- **Primary key:** (B_Code, B_Category_Code).

- **Foreign key:**
  - B_Category_Code references F_Code in the table Fabric_Category.

ORDER_STATUS:

- OS_Date: string that adheres to ISO 8601 Date standard.
- OS_Time: string that adheres to ISO 8601 Time standard.
- OS_Status: contains one of the strings 'New', 'Ordered', 'Partial paid, 'Full paid' and 'Cancelled'; default as 'New'.
- OS_Cancel_Reason: string of up to 255 characters.
- **Primary key:** (OS_Code, OS_Date, OS_Time).
- **Foreign key:**
  - OS_Code references O_Code in the table Order_Customer.
  - OS_Ostaff_code references E_Code of type 'Operational staff' in the table Employee.
- **Fields to be set NOT NULL:** OS_Status.

ORDER_BOLT:

- **Primary key:** (OB_Bolt_Code, OB_Category_Code, OB_Order_Code).
- **Foreign key:**
  - (OB_Bolt_Code, OB_Category_Code) references (B_Code, B_Category_Code) in the table Bolt.
  - OB_Order_Code references O_Code in the table Order_Customer.

SUPPLIER_PHONE:

- Phone_number: string of up to 15 digits.
- **Primary key:** (Supplier_Code, Phone_number).
- **Foreign key:**
  - Supplier_Code references S_Code in the table Supplier.

- **Fields to be set UNIQUE:** Phone_number.

CUSTOMER_PHONE:

- Phone_number: string of up to 15 digits.
- **Primary key:** (Customer_Code, Phone_number).
- **Foreign key:**
  - Customer_Code references C_Code in the table Customer.
- **Fields to be set UNIQUE:** Phone_number.

FABRIC_CATEGORY_PRICE:

- FCP_Date: string that adheres to ISO 8601 Date standard.
- FCP_Price: positive number with precision 10, scale 2.
- **Primary key:** (FCP_Category_Code, FCP_Date, FCP_Price).
- **Foreign key:**
  - FCP_Category_Code references F_Code in the table Fabric_Category.

ORDER_PAYMENT:

- OP_Amount: positive number with precision 10, scale 2.
- OP_Date: string that adheres to ISO 8601 Date standard.
- OP_Time: string that adheres to ISO 8601 Time standard.
- **Primary key:** (OP_Customer_Code, OP_Order_Code, OP_Date, OP_Time).
- **Foreign key:**
  - OP_Customer_Code references C_Code in the table Customer.
  - OP_Order_Code references O_Code in the table Order_Customer.
- **Fields to be set NOT NULL:** OP_Amount.

3.2 Triggers for checking and updating

In order to keep the data inserted or updated logical with the existing data in the database and also update the related data (i.e. derived attributes), we also add some triggers that run after the action of insertion or update of data:

- `check_partner_staff` (table `Supplier`):
  Raise an error and cancel the action if the inserted/updated `S_Pstaff_code` represents the employee not in type `'Partner staff'`.

- `check_office_staff` (table `Customer`):
  Raise an error and cancel the action if the inserted/updated `C_Office_staff_code` represents the employee not in type `'Office staff'`.

- `check_operational_staff1` (table `Order_Customer`):
  Raise an error and cancel the action if the inserted/updated `O_Ostaff_code` represents the employee not in type `'Operational staff'`.

- `check_unique_supplier` (table `Import_Information`):
  Raise an error and cancel the action if there are more than 1 supplier providing the category in the `inserted` row.

- `update_category_quantity1` (table `Import_Information`):
  Update the category quantity (in the table `Fabric_Category`) with the inserted/updated `I_Quantity`.

- `check_bolt` (table `Order_Bolt`):
  Raise an error and cancel the action if the bolt in the `inserted` row has appeared in an order that has not been cancelled.

- `update_total_price` (table `Order_Bolt`):

  – Get the category's price at the time the order in the `inserted` row is at the status `'New'`.

  – Use that price to update the total price (in the table `Order_Customer`) of the order.

- `check_operational_staff2` (table `Order_Status`):
  Raise an error and cancel the action if the inserted/updated `OS_Ostaff_code` (if exists) represents the employee not in type `'Operational staff'`.

- `update_when_change_status` (table `Order_Status`):

  – If the inserted `OS_Status` is `'Ordered'`:

    * Subtract the amount of bolts for each category in the order from the corresponding `F_Quantity` in the table `Fabric_Category`.
    * Add the total price of the order to the arrearage of the customer who made that order. If that arrearage becomes greater than 2000 and the `C_Warning` is still `‘0’`, set the `C_Warning` to `‘1’` and record the date that the order changes into status `‘Ordered’` into `C_Last_Waning_Date` (table `Customer`).
  - If the inserted `OS_Status` is `‘Cancelled’`:
    * Add the amount of bolts for each category in the order back to the corresponding `F_Quantity` in the table `Fabric_Category`.
    * Subtract the debt of the customer for that order (if exists) from his/her arrearage. If that arrearage becomes less than or equal to 2000 and the `C_Warning` is still `‘1’`, set the `C_Warning` to `‘0’` (table `Customer`).

- `check_correct_customer` (table `Order_Payment`):
  Raise an error and cancel the action if the customer who paid for the order is not the one who made that order.

- `check_payment` (table `Order_Payment`):
  - If the amount of payment exceeds the debt for the order, raise an error and cancel the action.
  - If the amount of payment is equal to the debt for the order, change that order's status into `‘Full paid’`; otherwise change it into `‘Partial paid’` if there have not been any payment for the order.
  - Subtract the amount of payment from the customer's arrearage. If that arrearage becomes less than or equal to 2000 and the `C_Warning` is still `‘1’`, set the `C_Warning` to `‘0’` (table `Customer`).

3.3 Data insertion

Now we will insert the data into the table we have created and make sure they are meaningful data. Some examples of data insertion for each table are as follows.

```sql
-- EMPLOYEE
Insert Into Employee(E_Code, E_FName, E_LName, E_Gender, E_Address, E_Type)
values ('0001','Harry','Wilson','M','731 Fondren, Houston, TX', 'Manager')
    --...
go

-- EMPLOYEE PHONE
Insert Into Employee_Phone(Employee_Code, Phone_Number)
values ('0001','0817000403')
```

```
    --...
go

-- SUPPLIER
Insert Into Supplier(S_Code, S_Name, S_Address, S_Taxcode, S_BankAccount,
    S_Pstaff_code)
values ('000001','John Murtough','271 Jester, Austin, Florida','555666777','
    000410004567', '2001')
    --...
go

-- SUPPLIER PHONE
Insert Into Supplier_Phone(Supplier_Code, Phone_number)
values ('000001','0989599989')
    --...
go

-- FABRIC CATEGORY
Insert Into Fabric_Category(F_Code, F_Name, F_Color)
values ('11111','Silk','#BDB1A8')
    --...
go

-- CUSTOMER
Insert Into Customer(C_Code, C_FName, C_LName, C_Address, C_Office_staff_code)
values ('22222221','Max','James', '403 Queensland, Jackson, Colorado','3001')
    --...
go

-- CUSTOMER PHONE
Insert Into Customer_Phone(Customer_Code, Phone_number)
values ('22222221','0908862194')
    --...
go

-- IMPORT INFORMATION
Insert Into Import_Information(I_Category_Code, I_Supplier_Code, I_Date,
    I_Quantity, I_Price)
values ('11111','000003','2023-03-04',10,150.0)
    --...
go

-- BOLT
Insert Into Bolt(B_Code, B_Category_Code, B_Length) values ('4000001','11111'
    ,100.0)
    --...
go
```

```
-- FABRIC CATEGORY PRICE
Insert Into Fabric_Category_Price(FCP_Category_Code, FCP_Date, FCP_Price)
values ('11111','2022-11-01',250.0)
    --...
go

-- TABLES RELATED TO ORDER
-- Order 1
Insert Into Order_Customer(O_Code, O_Customer_Code, O_Ostaff_code)
values ('3000000001','22222221','1001')

Insert Into Order_Status(OS_Code, OS_Date, OS_Time, OS_Status, OS_Ostaff_code)
values ('3000000001','2020-07-09','16:30:25','New','1001')

Insert Into Order_Bolt(OB_Bolt_Code, OB_Category_Code, OB_Order_Code)
values ('4000001','11115','3000000001')
    --...

Insert Into Order_Status(OS_Code, OS_Date, OS_Time, OS_Status, OS_Ostaff_code)
values ('3000000001','2020-07-12','09:14:37','Ordered','1001')

Insert Into Order_Payment(OP_Order_Code, OP_Amount, OP_Date, OP_Time,
    OP_Customer_Code)
values ('3000000001',700.0,'2020-07-12','09:21:12','22222221')
    --...
go

--...
```

4 Solution to SQL requirements

4.1 Requirement 1

Increase Silk selling price to 10% of those provided by all suppliers from 01/09/2020.

```
INSERT INTO Fabric_Category_Price (FCP_Category_Code, FCP_Date, FCP_Price)
  SELECT FCP_Category_Code, CAST(GETDATE() AS DATE) AS new_date, FCP_Price *
    1.1 AS new_price
  FROM Fabric_Category_Price,
    (SELECT FCP_Category_Code AS Code, MAX(FCP_Date) AS FCP_Latest_Date
    FROM Fabric_Category_Price
    GROUP BY FCP_Category_Code
    HAVING FCP_Category_Code IN
      (SELECT DISTINCT I_Category_Code
      FROM Import_Information
      WHERE I_Date >= '2020-09-01')) AS Latest_Date
  WHERE FCP_Category_Code = Code AND FCP_Date = FCP_Latest_Date;
```

4.2 Requirement 2

Select all orders containing bolt from the supplier named ‘Silk Agency’.

```
SELECT *
FROM Order_Customer
WHERE O_Code IN
    (SELECT DISTINCT OB_Order_Code
    FROM Order_Bolt
    WHERE OB_Category_Code IN
  (SELECT DISTINCT I_Category_Code
  FROM Supplier JOIN Import_Information ON S_Code = I_Supplier_Code
  WHERE S_Name = 'Silk Agency'));
```

4.3 Requirement 3

Write a function to calculate the total purchase price the agency has to pay for each supplier.

Input: Supplier ID

Output: A list of payment

```
CREATE FUNCTION CalculateSupplierPayment(@SupplierID CHAR(6))
RETURNS TABLE
AS
RETURN
(
    SELECT I_Supplier_Code AS SupplierID, SUM(I_Quantity * I_Price) AS
    TotalPayment
    FROM Import_Information
    WHERE I_Supplier_Code = @SupplierID
    GROUP BY I_Supplier_Code
);
```

4.4 Requirement 4

Write a procedure to sort the suppliers in increasing number of categories they provide in a period of time.

Input: Start date, End date.

Output: A list of sorting suppliers.

```
CREATE PROCEDURE SortSuppliersByCategories(
    @StartDate  DATE,
    @EndDate  DATE
)
AS
BEGIN
    SELECT I_Supplier_Code AS SupplierID, COUNT(DISTINCT I_Category_Code) AS
    CategoryCount
    FROM Import_Information
    WHERE I_Date BETWEEN @StartDate AND @EndDate
    GROUP BY I_Supplier_Code
    ORDER BY COUNT(DISTINCT I_Category_Code) ASC;
END;
```

5 Building applications

5.1 Project repositories

We splitted the application into two parts: the frontend (UI) and the backend (data manipulation).
The frontend repository can be found at: https://github.com/hitonichi/dbs-fabric-agency-frontend
The backend repository can be found at: https://github.com/hitonichi/dbs-fabric-agency-backend

We managed to implemented and got the application work under local environment.

5.2 Requirement functions

5.2.1 Log in, log out from the system

Every time an unauthenticated user access the site (on any route, path), that user is redirected to the sign in page (Figure 3, and is required to fill in the credentials to sign in to use other features.

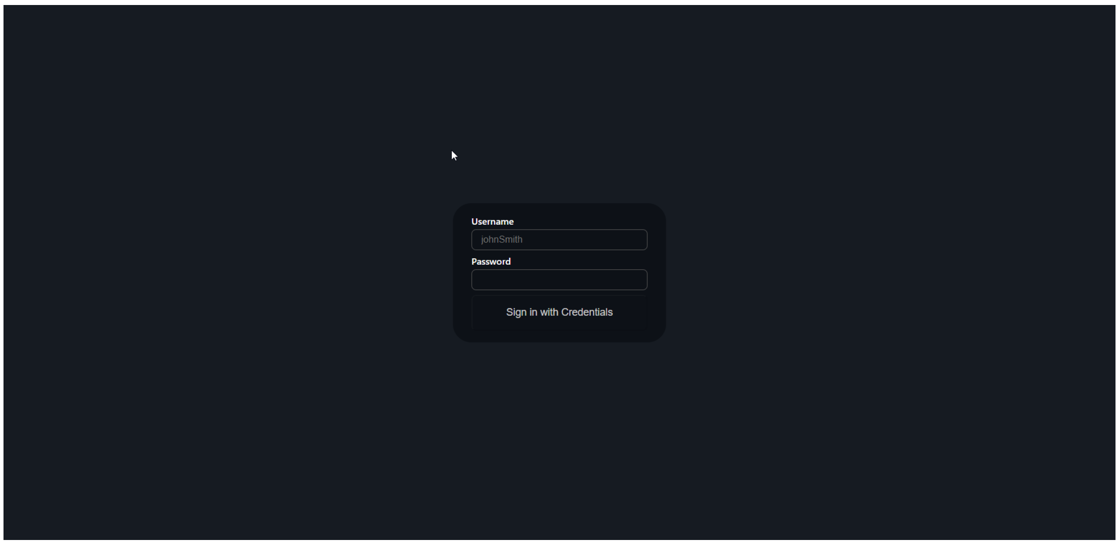


Figure 3: Sign in view

5.2.2 Requirement 1

Search material purchasing information: Search results include the name, phone number of the suppliers and information about the supply.

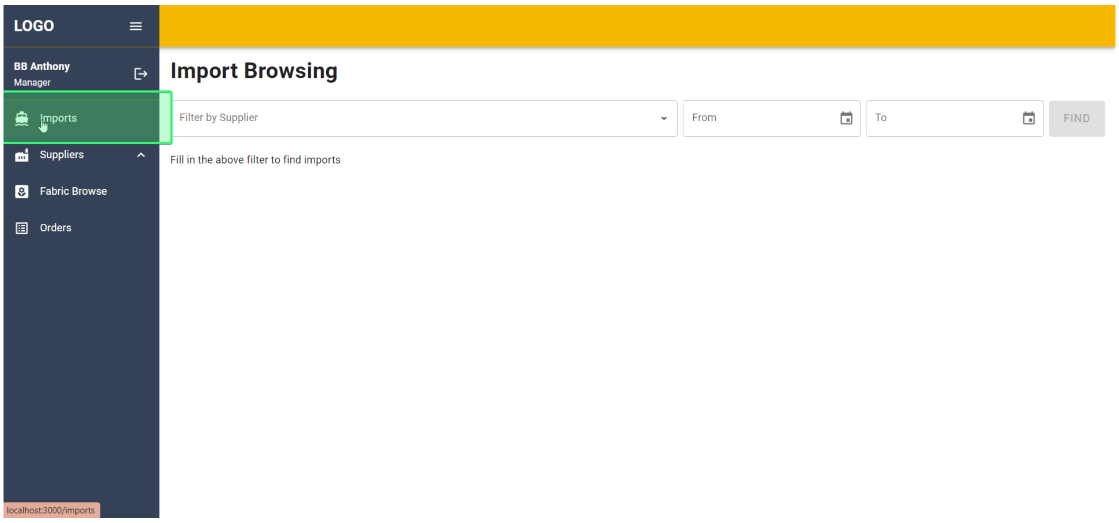


Figure 4: Step 1 - Home screen view

Step 1 (Figure 4): Firstly, the user (Manager) can access the import browsing page from anywhere on the site, by clicking into the 'Imports' button located on the sidebar. The site will then navigate to the import default view.

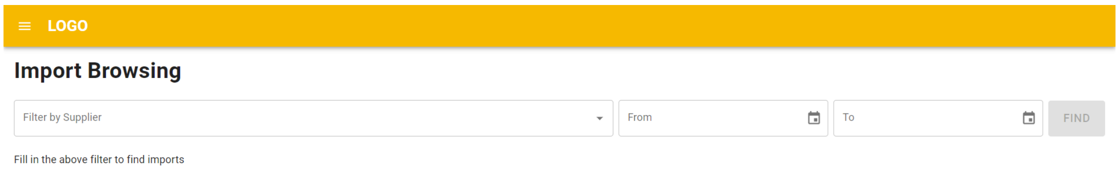
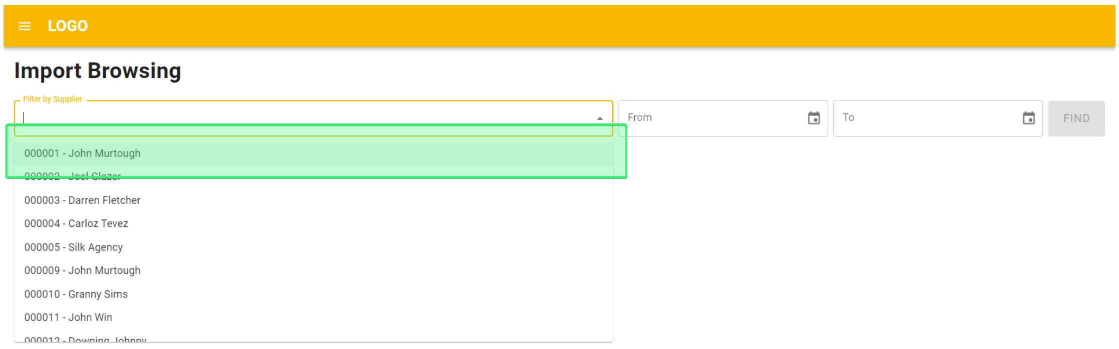


Figure 5: Step 2 - Import Search view (empty)

Step 2 (Figure 5): Initially, there will be no result on this page since the user hasn't insert any search parameters.

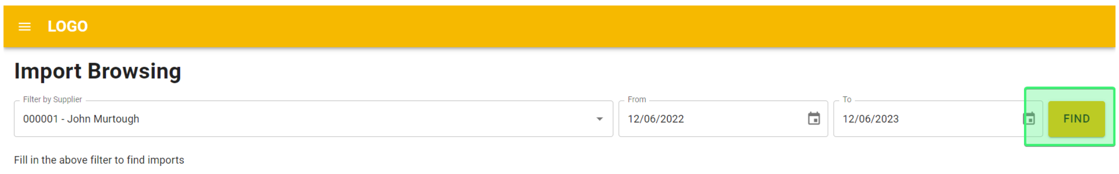


Import Browsing

000001 - John Murtough
000002 - Joel Glazer
000003 - Darren Fletcher
000004 - Carloz Tevez
000005 - Silk Agency
000009 - John Murtough
000010 - Granny Sims
000011 - John Win
000012 - Downing Johnny

Figure 6: Step 3 - Import Search view (selecting supplier)

Step 3 (Figure 6): To search for imports, the user can click onto any of the three input fields, and choose the desired options that the system provides for supplier, start date and end date.



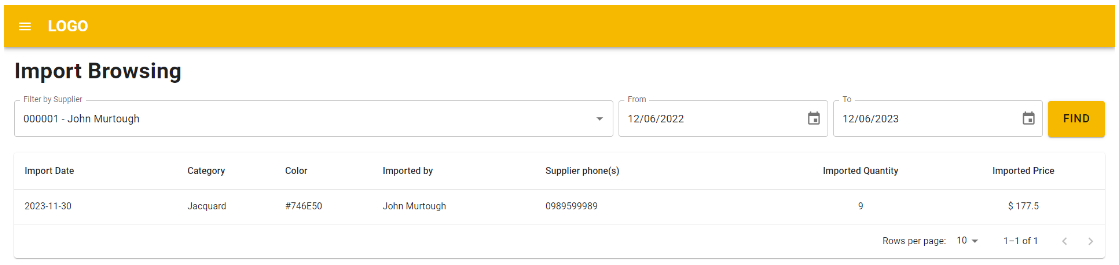
Import Browsing

Filter by Supplier: 000001 - John Murtough
From: 12/06/2022
To: 12/06/2023

Fill in the above filter to find imports

Figure 7: Step 4 - Import Search view (submitting)

Step 4 (Figure 7): User confirmed and select 'Find' to tell the system to fetch data.
Step 5 (Figure 8): The result will be loaded into a tabular form, showing each import's information (date, import category & color, supplier's name and phone(s), imported quantity and price).



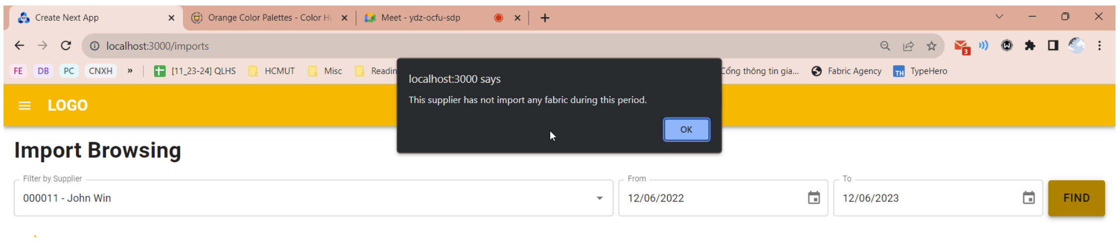
Import Browsing

Filter by Supplier: 000001 - John Murtough
From: 12/06/2022
To: 12/06/2023

Import Date	Category	Color	Imported by	Supplier phone(s)	Imported Quantity	Imported Price
2023-11-30	Jacquard	#746E50	John Murtough	0989599989	9	$ 177.5

Rows per page: 10 1–1 of 1

Figure 8: Step 5 - Import Search view (result loaded)



localhost:3000 says

This supplier has not import any fabric during this period.

Import Browsing

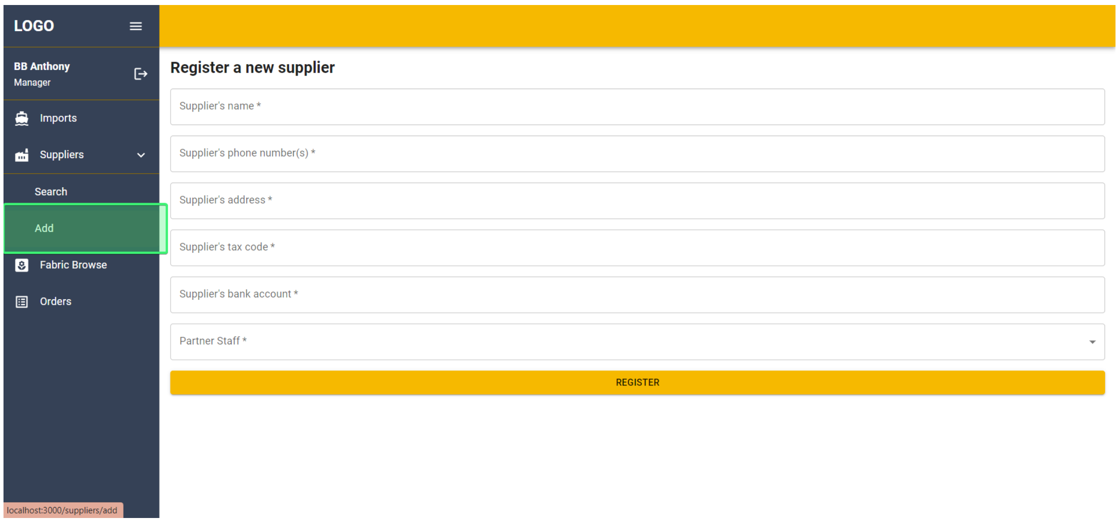
Filter by Supplier: 000011 - John Win
From: 12/06/2022
To: 12/06/2023

Figure 9: Import Search view - empty result

No result (Figure 9): If the resulting query is empty, which means there is no import information for the specified search queries, the system will inform the Manager to try another search.

5.2.3 Requirement 2

Add information for a new supplier.



Register a new supplier

Supplier's name *

Supplier's phone number(s) *

Supplier's address *

Supplier's tax code *

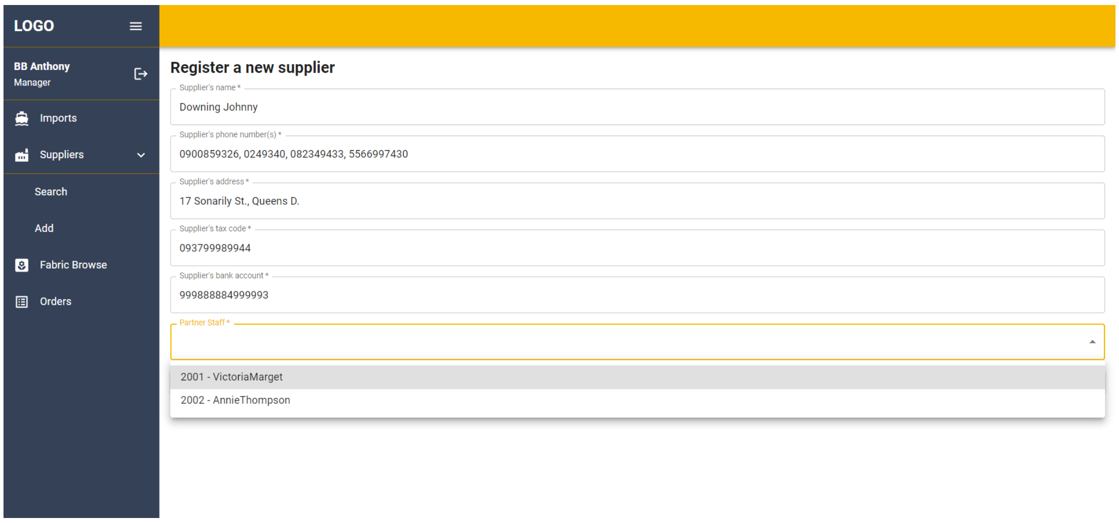
Supplier's bank account *

Partner Staff *

REGISTER

Figure 10: Step 1 - Home screen view

Step 1 (Figure 10): Firstly, the user (Manager) can access the supplier creation page from anywhere on the site, by clicking into the 'Add' option under the 'Supplier' option located on the sidebar. The site will then navigate to the creation form.



Register a new supplier

Supplier's name *: Downing Johnny

Supplier's phone number(s) *: 0900859326, 0249340, 082349433, 5566997430

Supplier's address *: 17 Sonarily St., Queens D.

Supplier's tax code *: 093799989944

Supplier's bank account *: 999888884999993

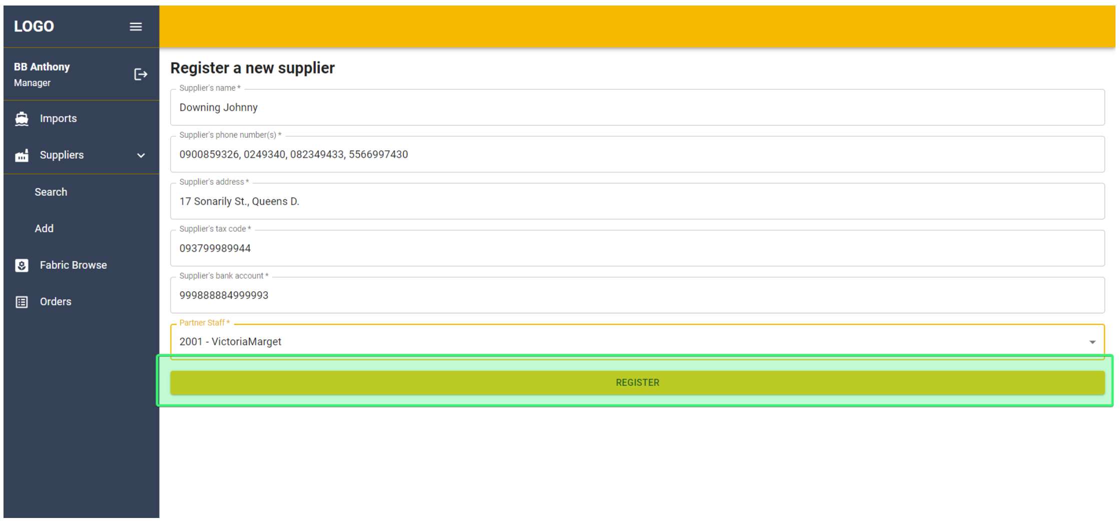
Partner Staff *

2001 - VictoriaMarget

2002 - AnnieThompson

Figure 11: Step 2 - Supplier form view (filling in fields, choosing partner staff)

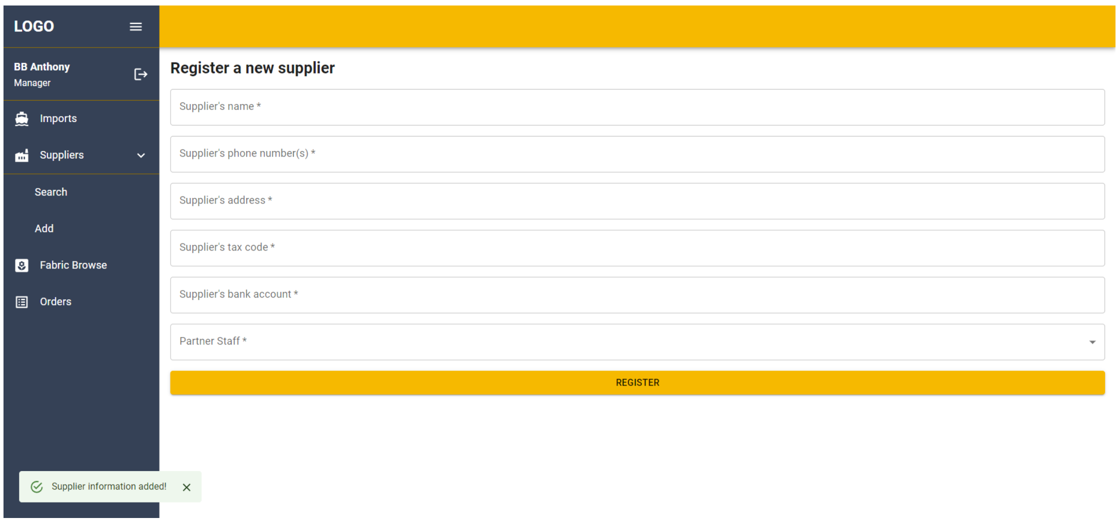
Step 2 (Figure 11): Initially, the form will be empty, showing fields for the new supplier's name, phone number(s), address, tax code, bank account and the selection for partner staff.


Register a new supplier

Supplier's name * : Downing Johnny
Supplier's phone number(s) * : 0900859326, 0249340, 082349433, 5566997430
Supplier's address * : 17 Sonarily St., Queens D.
Supplier's tax code * : 093799989944
Supplier's bank account * : 999888884999993
Partner Staff * : 2001 - VictoriaMarget

Figure 12: Step 3 - Supplier form view (submitting)

Step 3 (Figure 12): The manager fills in all of the fields, and select the desired staff in charge, then he/she can click onto the 'Register' button to send a creation request.

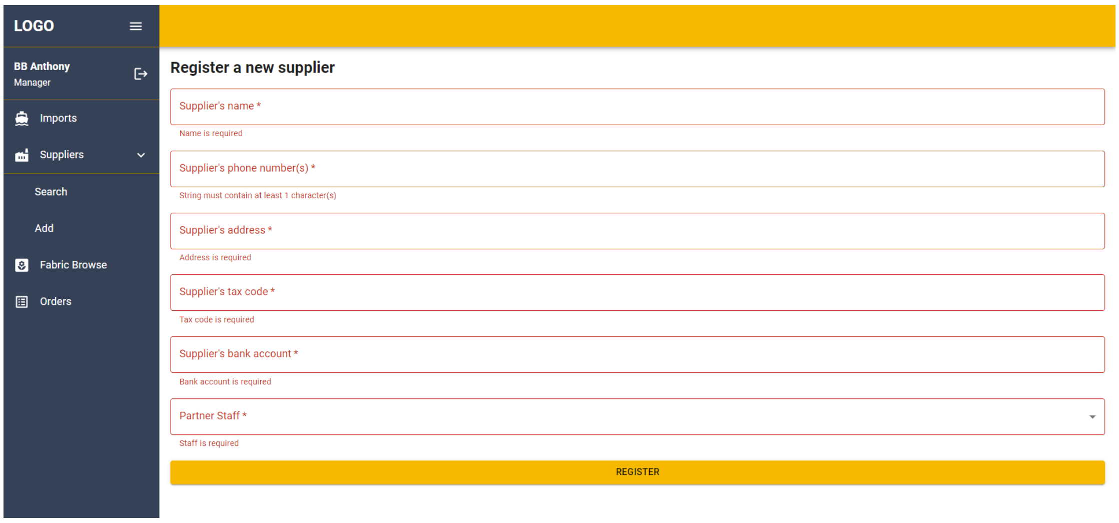

Register a new supplier

Supplier's name *
Supplier's phone number(s) *
Supplier's address *
Supplier's tax code *
Supplier's bank account *
Partner Staff *

Supplier information added!

Figure 13: Step 4 - Supplier form view (submitted)

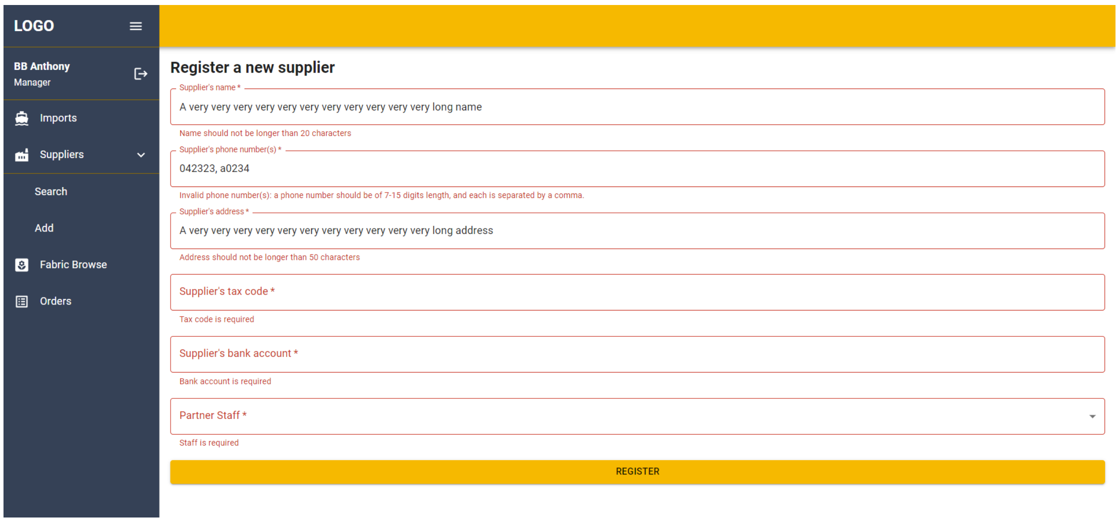
Step 4 (Figure 13): A successful submission will result in a status pop-up at the bottom-left corner of the screen.
Empty fields (Figure 14): Any field that is left empty will be alerted, and the 'Register' button will not function until the errors go away.



Register a new supplier

Supplier's name *
Name is required
Supplier's phone number(s) *
String must contain at least 1 character(s)
Supplier's address *
Address is required
Supplier's tax code *
Tax code is required
Supplier's bank account *
Bank account is required
Partner Staff *
Staff is required

Figure 14: Supplier form view - Invalid input (1)



Register a new supplier

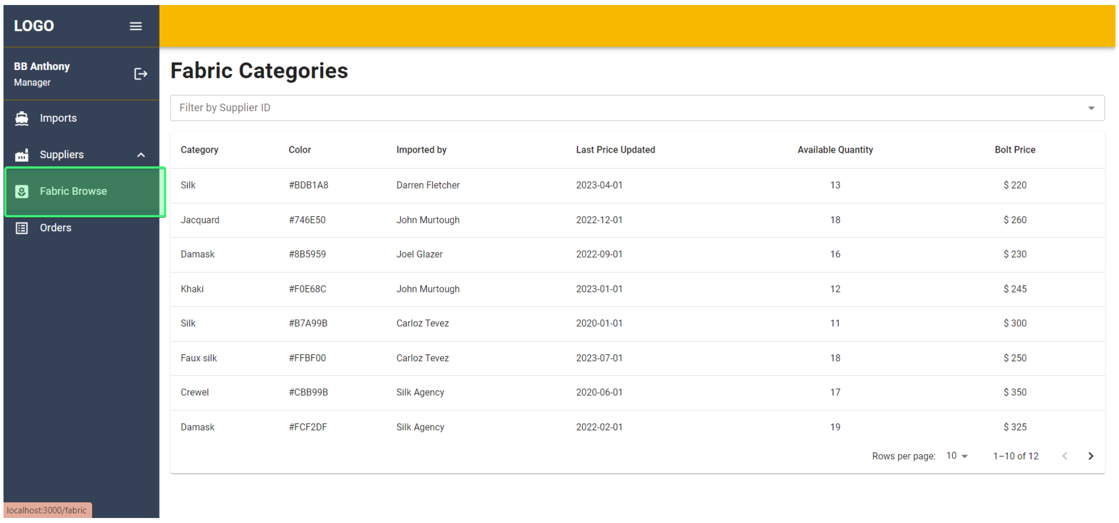
Supplier's name *: A very very very very very very very very very very very very long name
Name should not be longer than 20 characters
Supplier's phone number(s) *: 042323, a0234
Invalid phone number(s): a phone number should be of 7-15 digits length, and each is separated by a comma.
Supplier's address *: A very very very very very very very very very very very very long address
Address should not be longer than 50 characters
Supplier's tax code *
Tax code is required
Supplier's bank account *
Bank account is required
Partner Staff *
Staff is required

Figure 15: Supplier form view - Invalid input (2)

Invalid fields (Figure 14): The system also validates some fields, such as the length constraint and the format of phone numbers, and also prevent user from submitting the creation if the inputs are invalid.

5.2.4 Requirement 3

List details of all categories which are provided by a supplier.



Fabric Categories

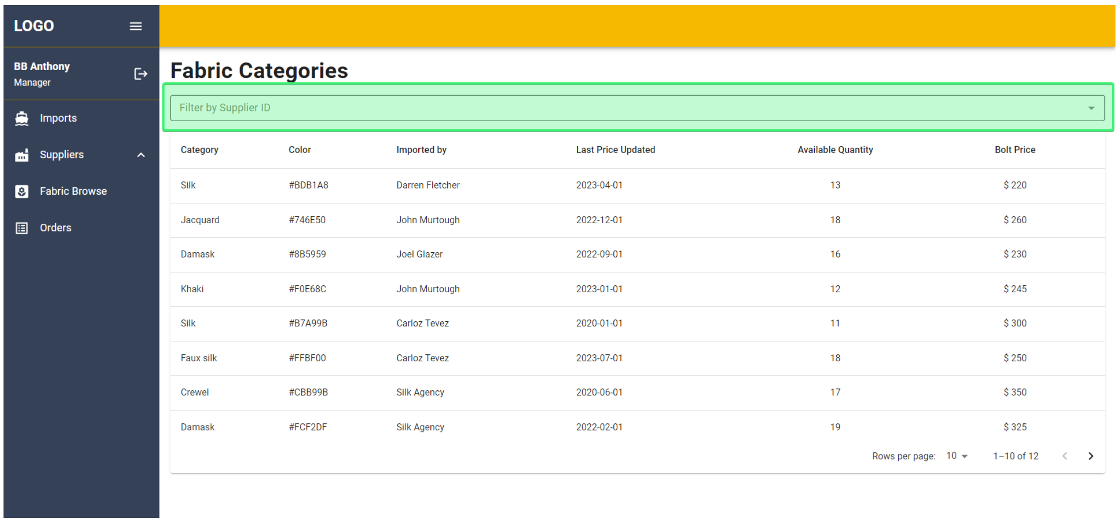
Filter by Supplier ID

Category	Color	Imported by	Last Price Updated	Available Quantity	Bolt Price
Silk	#BDB1A8	Darren Fletcher	2023-04-01	13	$ 220
Jacquard	#746E50	John Murtough	2022-12-01	18	$ 260
Damask	#8B5959	Joel Glazer	2022-09-01	16	$ 230
Khaki	#F0E68C	John Murtough	2023-01-01	12	$ 245
Silk	#B7A99B	Carloz Tevez	2020-01-01	11	$ 300
Faux silk	#FFBF00	Carloz Tevez	2023-07-01	18	$ 250
Crewel	#CBB99B	Silk Agency	2020-06-01	17	$ 350
Damask	#FCF2DF	Silk Agency	2022-02-01	19	$ 325

Rows per page: 10 1–10 of 12

Figure 16: Step 1 - Home screen view

Step 1 (Figure 16): Firstly, the user (Manager) can access the fabric categories page from anywhere on the site, by clicking onto the 'Fabric Browse' option located on the sidebar. The site will then navigate to the fabric view.



Fabric Categories

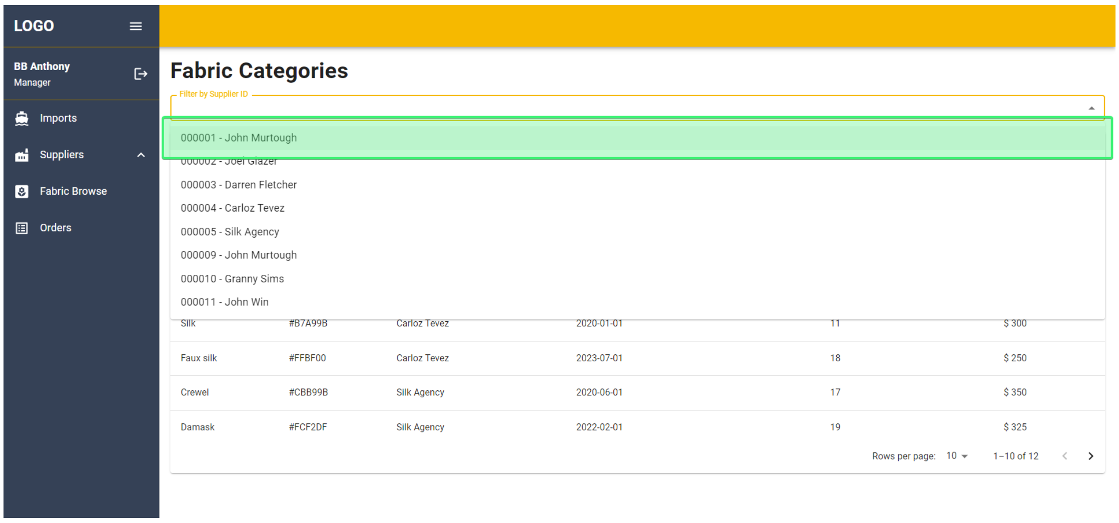
Filter by Supplier ID

Category	Color	Imported by	Last Price Updated	Available Quantity	Bolt Price
Silk	#BDB1A8	Darren Fletcher	2023-04-01	13	$ 220
Jacquard	#746E50	John Murtough	2022-12-01	18	$ 260
Damask	#8B5959	Joel Glazer	2022-09-01	16	$ 230
Khaki	#F0E68C	John Murtough	2023-01-01	12	$ 245
Silk	#B7A99B	Carloz Tevez	2020-01-01	11	$ 300
Faux silk	#FFBF00	Carloz Tevez	2023-07-01	18	$ 250
Crewel	#CBB99B	Silk Agency	2020-06-01	17	$ 350
Damask	#FCF2DF	Silk Agency	2022-02-01	19	$ 325

Rows per page: 10 1–10 of 12

Figure 17: Step 2 - Fabric Category view (un-filtered)

Step 2 (Figure 17): Initially, the system will load all categories available in the system as tabular form, showing some information about the categories, such as name, color, imported supplier, last price update time, available quantity and current price.



Fabric Categories

Filter by Supplier ID

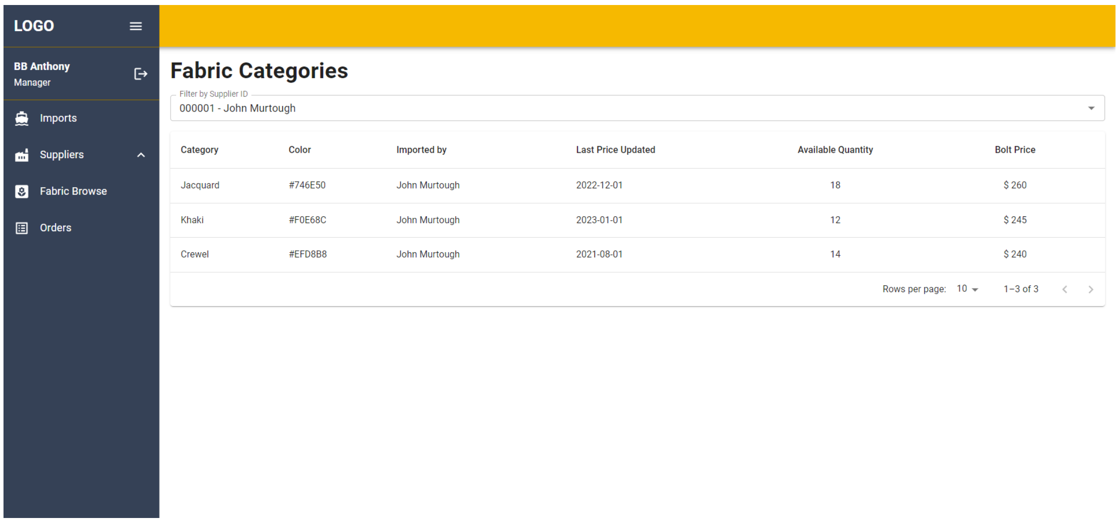
000001 - John Murtough
000002 - Joel Glazer
000003 - Darren Fletcher
000004 - Carloz Tevez
000005 - Silk Agency
000009 - John Murtough
000010 - Granny Sims
000011 - John Win

Silk	#B7A99B	Carloz Tevez	2020-01-01	11	$ 300
Faux silk	#FFBF00	Carloz Tevez	2023-07-01	18	$ 250
Crewel	#CBB99B	Silk Agency	2020-06-01	17	$ 350
Damask	#FCF2DF	Silk Agency	2022-02-01	19	$ 325

Rows per page: 10   1–10 of 12

Figure 18: Step 3 - Fabric Category view (choosing supplier)

Step 3 (Figure 18): The manager can further filter out the original list by selecting a supplier to see categories from that supplier only.



Fabric Categories

Filter by Supplier ID: 000001 - John Murtough

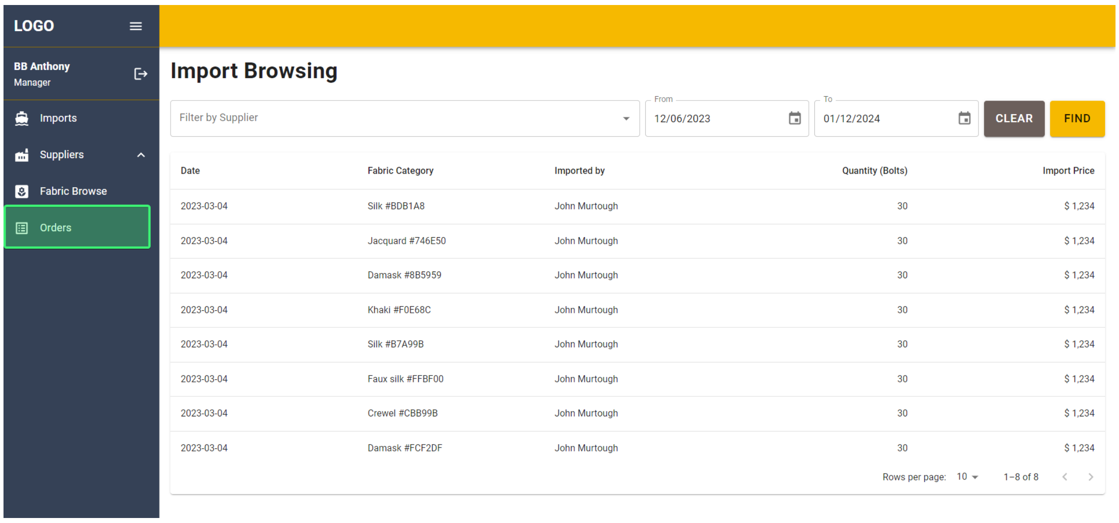
Category	Color	Imported by	Last Price Updated	Available Quantity	Bolt Price
Jacquard	#746E50	John Murtough	2022-12-01	18	$ 260
Khaki	#F0E68C	John Murtough	2023-01-01	12	$ 245
Crewel	#EFD8B8	John Murtough	2021-08-01	14	$ 240

Rows per page: 10   1–3 of 3

Figure 19: Step 4 - Fabric Category view (filtered)

5.2.5 Requirement 4

Make a report that provides full information about the order for each category of a customer.



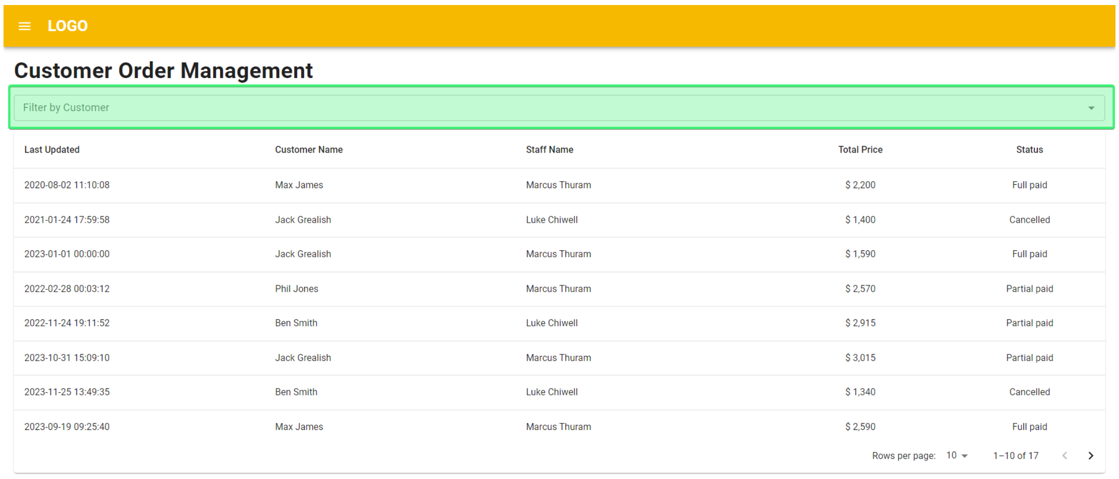
Import Browsing

From 12/06/2023 To 01/12/2024

Date	Fabric Category	Imported by	Quantity (Bolts)	Import Price
2023-03-04	Silk #BDB1A8	John Murtough	30	$ 1,234
2023-03-04	Jacquard #746E50	John Murtough	30	$ 1,234
2023-03-04	Damask #8B5959	John Murtough	30	$ 1,234
2023-03-04	Khaki #F0E68C	John Murtough	30	$ 1,234
2023-03-04	Silk #B7A99B	John Murtough	30	$ 1,234
2023-03-04	Faux silk #FFBF00	John Murtough	30	$ 1,234
2023-03-04	Crewel #CBB99B	John Murtough	30	$ 1,234
2023-03-04	Damask #FCF2DF	John Murtough	30	$ 1,234

Figure 20: Step 1 - Home screen view

Step 1 (Figure 20): Firstly, the user (Manager) can access the order list page from anywhere on the site, by clicking onto the 'Orders' option located on the sidebar. The site will then navigate to the order view.

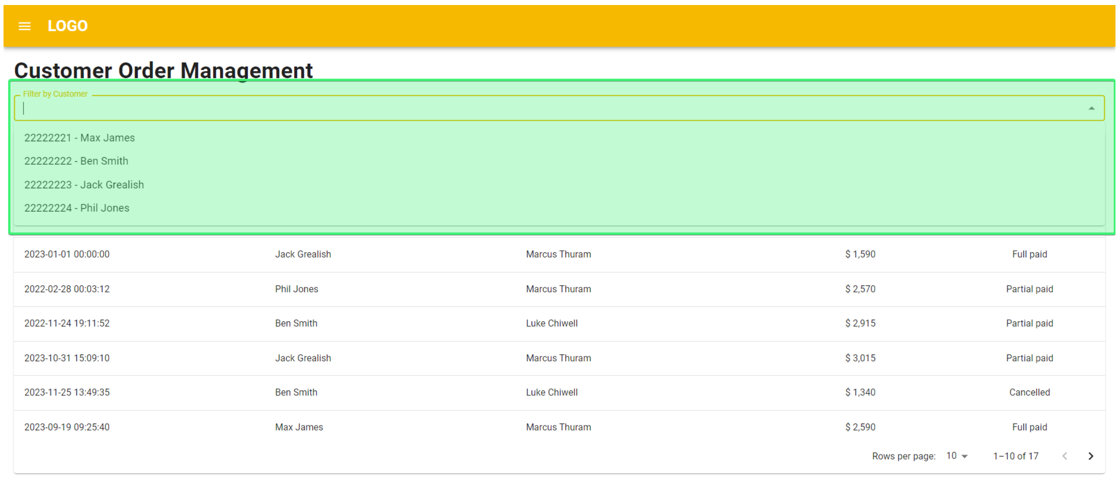


Customer Order Management

Last Updated	Customer Name	Staff Name	Total Price	Status
2020-08-02 11:10:08	Max James	Marcus Thuram	$ 2,200	Full paid
2021-01-24 17:59:58	Jack Grealish	Luke Chiwell	$ 1,400	Cancelled
2023-01-01 00:00:00	Jack Grealish	Marcus Thuram	$ 1,590	Full paid
2022-02-28 00:03:12	Phil Jones	Marcus Thuram	$ 2,570	Partial paid
2022-11-24 19:11:52	Ben Smith	Luke Chiwell	$ 2,915	Partial paid
2023-10-31 15:09:10	Jack Grealish	Marcus Thuram	$ 3,015	Partial paid
2023-11-25 13:49:35	Ben Smith	Luke Chiwell	$ 1,340	Cancelled
2023-09-19 09:25:40	Max James	Marcus Thuram	$ 2,590	Full paid

Figure 21: Step 2 - Order list view (un-filtered)

Step 2 (Figure 21): Initially, the system will load all orders available in the system as tabular form, showing some information about the categories, such as last status update timestamp, customer name, in-charge staff name, total price and status.



Customer Order Management

Filter by Customer

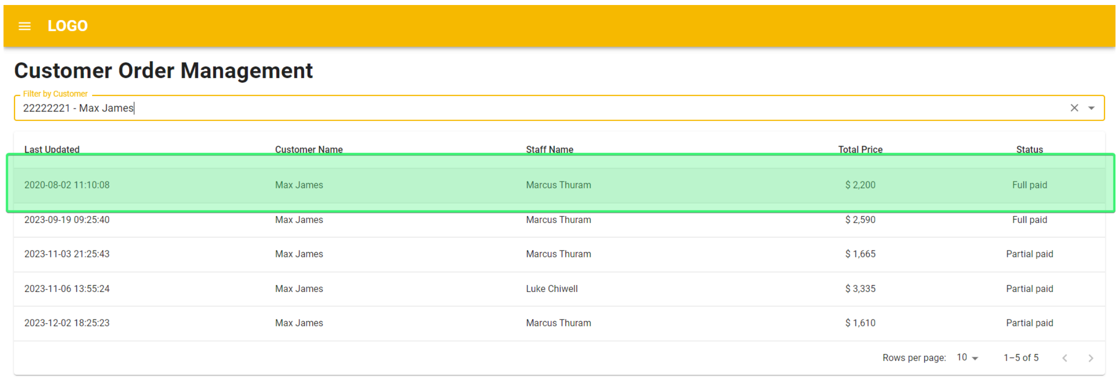
22222221 - Max James
22222222 - Ben Smith
22222223 - Jack Grealish
22222224 - Phil Jones

2023-01-01 00:00:00	Jack Grealish	Marcus Thuram	$ 1,590	Full paid
2022-02-28 00:03:12	Phil Jones	Marcus Thuram	$ 2,570	Partial paid
2022-11-24 19:11:52	Ben Smith	Luke Chiwell	$ 2,915	Partial paid
2023-10-31 15:09:10	Jack Grealish	Marcus Thuram	$ 3,015	Partial paid
2023-11-25 13:49:35	Ben Smith	Luke Chiwell	$ 1,340	Cancelled
2023-09-19 09:25:40	Max James	Marcus Thuram	$ 2,590	Full paid

Rows per page: 10 1–10 of 17

Figure 22: Step 3 - Order list view (choosing customer)

Step 3 (Figure 22): The manager can further filter out the original list by selecting a customer to see orders from that customer only.



Customer Order Management

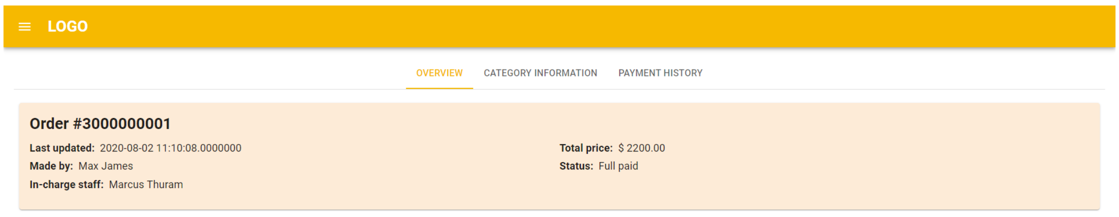
Filter by Customer: 22222221 - Max James

Last Updated	Customer Name	Staff Name	Total Price	Status
2020-08-02 11:10:08	Max James	Marcus Thuram	$ 2,200	Full paid
2023-09-19 09:25:40	Max James	Marcus Thuram	$ 2,590	Full paid
2023-11-03 21:25:43	Max James	Marcus Thuram	$ 1,665	Partial paid
2023-11-06 13:55:24	Max James	Luke Chiwell	$ 3,335	Partial paid
2023-12-02 18:25:23	Max James	Marcus Thuram	$ 1,610	Partial paid

Rows per page: 10 1–5 of 5

Figure 23: Step 4 - Order list view (selecting order)

Step 4 (Figure 23): The manager can also click on a specific row to discover more information about an order, which will navigate he/she to the order information page.

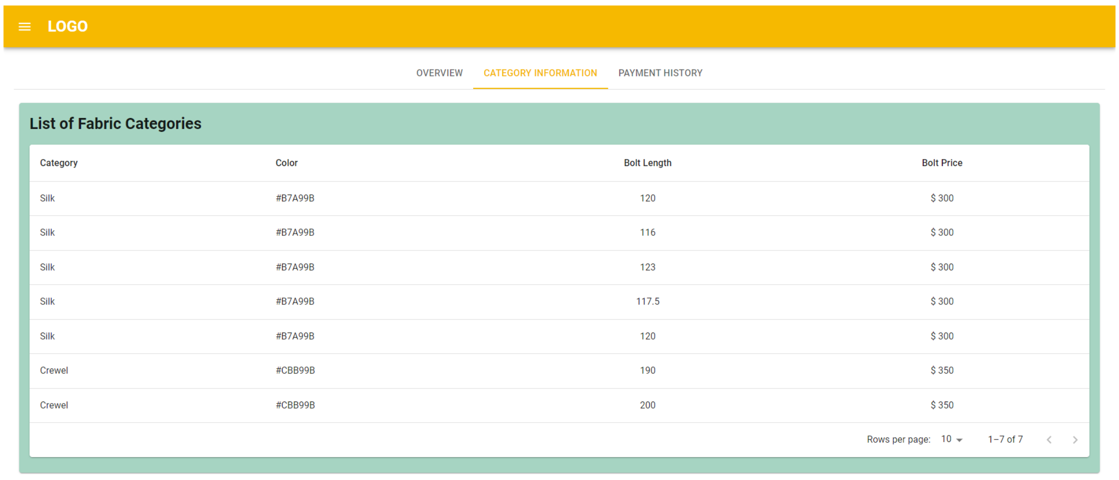


Order #3000000001

Last updated: 2020-08-02 11:10:08.0000000
Made by: Max James
In-charge staff: Marcus Thuram
Total price: $ 2200.00
Status: Full paid

Figure 24: Step 5 - Order view (Overview)

Step 5 (Figure 24): Inside the view, there will be three main tabs: Overview, Category information and Payment history. The overview will shows information about the order itself.



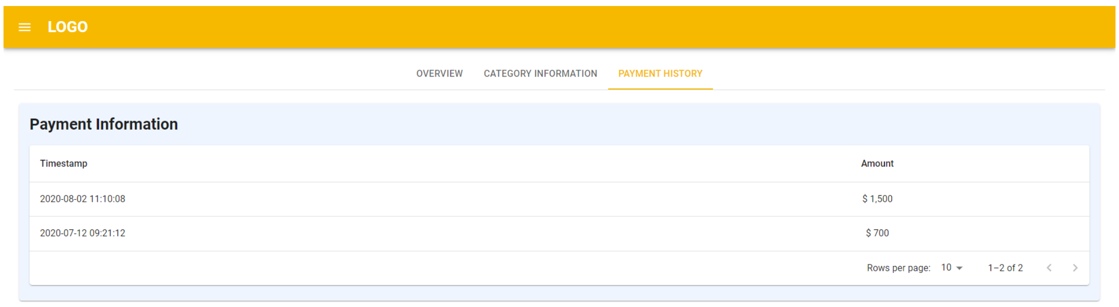
List of Fabric Categories

Category	Color	Bolt Length	Bolt Price
Silk	#B7A99B	120	$ 300
Silk	#B7A99B	116	$ 300
Silk	#B7A99B	123	$ 300
Silk	#B7A99B	117.5	$ 300
Silk	#B7A99B	120	$ 300
Crewel	#CBB99B	190	$ 350
Crewel	#CBB99B	200	$ 350

Rows per page: 10 1–7 of 7

Figure 25: Step 6 - Order view (Bolts)

Step 6 (Figure 25): In the category information section, there will be a table listing all bolts that is ordered in this order.
Step 7 (26): And in the payment history, there will also be a table showing the payments that the customer has made for this order.



Payment Information

Timestamp	Amount
2020-08-02 11:10:08	$ 1,500
2020-07-12 09:21:12	$ 700

Rows per page: 10 1–2 of 2

Figure 26: Step 7 - Order view (Payment)

6 Database management

6.1 Proving one use-case of indexing efficiency

In this section, we will check the indexing efficiency of the use-case of searching all bolts with a given bolt code. The comparison between using index and not using index will be shown in 3 ways: calculating the number of page accesses (as page is the fundamental unit of data storage in SQL Server), checking the time to execute the query and checking the estimated execution plan.

To make the proof more clearly, first of all, we would like to add more records to the table `Bolt` so that each category has 10 million bolts, which means the table `Bolt` will contain 120 million records. We use the following procedure to achieve that:

```
CREATE PROCEDURE add_bolt
  (@category CHAR(5), @start CHAR(7), @end INT)
AS
BEGIN
    DECLARE @count INT;
    SET @count = CONVERT(INT, @start);
    WHILE @count < @end
    BEGIN
        INSERT INTO Bolt(B_Category_Code, B_Code, B_Length)
        VALUES (
            @category,
            -- create B_code with 7 characters from @count
            REPLICATE('0', 7 - LEN(@count)) + CAST(@count AS VARCHAR(7)),
            -- create random B_Length with the value in [100,200], scale 2
            100 + ABS(CAST(CHECKSUM(NEWID()) % 10001 AS FLOAT)) / 100
        );
        SET @count = @count + 1;
    END;
END;
```

And then we execute the procedure, for example:

```
EXECUTE add_bolt '11111', '0000000', 4000001;
EXECUTE add_bolt '11111', '4000021', 10000000;
go
```

Here we consider using cluster index on the column `B_Code` and assume that the file records are unspanned.

6.1.1 Calculating the number of page accesses

Page size: $P = 8192$ bytes.
(The data pages in SQL Server are all the same size: 8KB or 8192 bytes)

Total number of records: $r = 120\,000\,000$ records.

Record length: $R = 17$ bytes.

- `B_Code` (type `CHAR(7)`): 7 bytes.
- `B_Category_Code` (type `CHAR(5)`): 5 bytes.
- `B_Length` (type `DECIMAL(5,2)`): 5 bytes.

Number of records in each page: $pfr = \left\lfloor \frac{P}{R} \right\rfloor = \left\lfloor \frac{8192}{17} \right\rfloor = 481$ records.

Number of pages needed for the table: $p = \left\lceil \frac{r}{pfr} \right\rceil = \left\lceil \frac{120\,000\,000}{481} \right\rceil = 249\,481$ pages.

Without indexing:

The approximate number of page accesses for a binary search on the data file:
$N_1 = \lceil \log_2 p \rceil = \lceil \log_2(249\,481) \rceil = 18$ page accesses.

Size of each index entry: $R_i = 11$ bytes.

- Key field (`B_Code`): 7 bytes.
- Pointer to data page (assuming for 32-bit systems): 4 bytes.

Total number of index entries: $r_i = 10\,000\,000$ entries.
(There are 10 million different values for `B_Code`, so the cluster index will have 10 million entries in total)

Number of index entries in each page: $pfr_i = \left\lfloor \frac{P}{R_i} \right\rfloor = \left\lfloor \frac{8192}{11} \right\rfloor = 744$ entries.

Number of index pages: $p = \left\lceil \frac{r_i}{pfr_i} \right\rceil = \left\lceil \frac{10\,000\,000}{744} \right\rceil = 13\,441$ pages.

With cluster indexing:

The approximate total number of page accesses if a binary search is used on the index:
$N_2 = \lceil \log_2 p \rceil + 1 = \lceil \log_2(13\,441) \rceil + 1 = 15$ page accesses.

Comparing N_1 with N_2, it can be seen that the cluster index optimizes the searching of data.

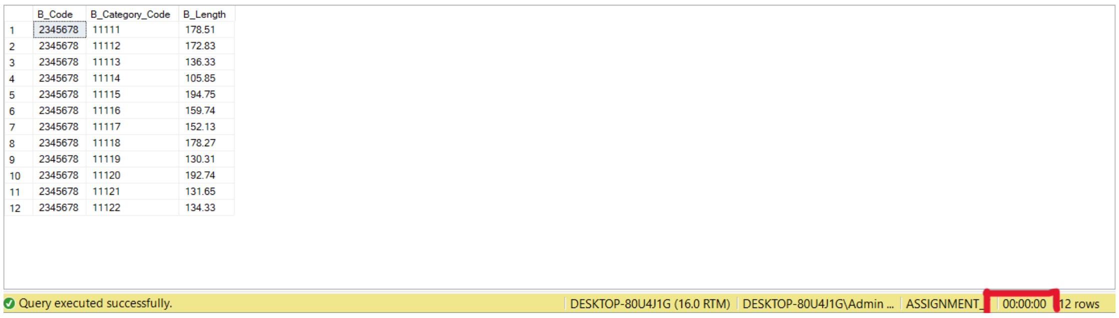
6.1.2 Checking the executing time

In SQL Server, the cluster index for the primary key is created by default. Therefore, in order to run a query with the cluster index, just simply do as usual:

```sql
SELECT * FROM Bolt WHERE B_Code = '2345678';
```

The result for that query with the execution time is shown in Figure 27.

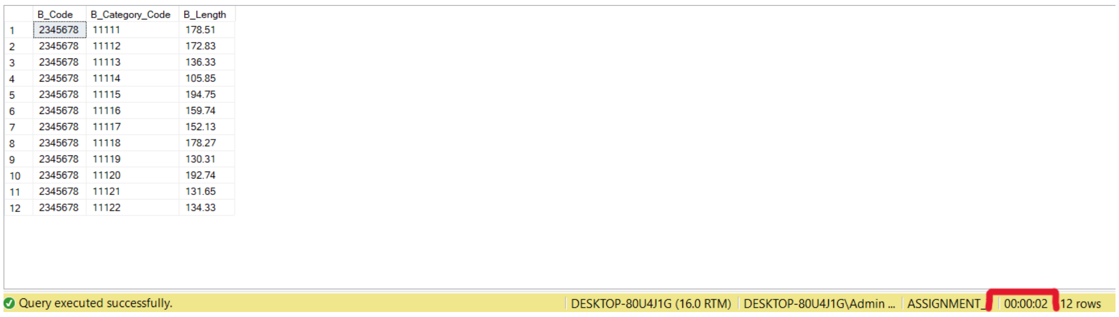
Now to force the table scan in the table, we run the following query:



	B_Code	B_Category_Code	B_Length
1	2345678	11111	178.51
2	2345678	11112	172.83
3	2345678	11113	136.33
4	2345678	11114	105.85
5	2345678	11115	194.75
6	2345678	11116	159.74
7	2345678	11117	152.13
8	2345678	11118	178.27
9	2345678	11119	130.31
10	2345678	11120	192.74
11	2345678	11121	131.65
12	2345678	11122	134.33

```
Query executed successfully.    DESKTOP-80U4J1G (16.0 RTM) | DESKTOP-80U4J1G\Admin ... | ASSIGNMENT_ | 00:00:00 | 12 rows
```

Figure 27: Execution time with cluster index - less than 1 second



	B_Code	B_Category_Code	B_Length
1	2345678	11111	178.51
2	2345678	11112	172.83
3	2345678	11113	136.33
4	2345678	11114	105.85
5	2345678	11115	194.75
6	2345678	11116	159.74
7	2345678	11117	152.13
8	2345678	11118	178.27
9	2345678	11119	130.31
10	2345678	11120	192.74
11	2345678	11121	131.65
12	2345678	11122	134.33

```
Query executed successfully.    DESKTOP-80U4J1G (16.0 RTM) | DESKTOP-80U4J1G\Admin ... | ASSIGNMENT_ | 00:00:02 | 12 rows
```

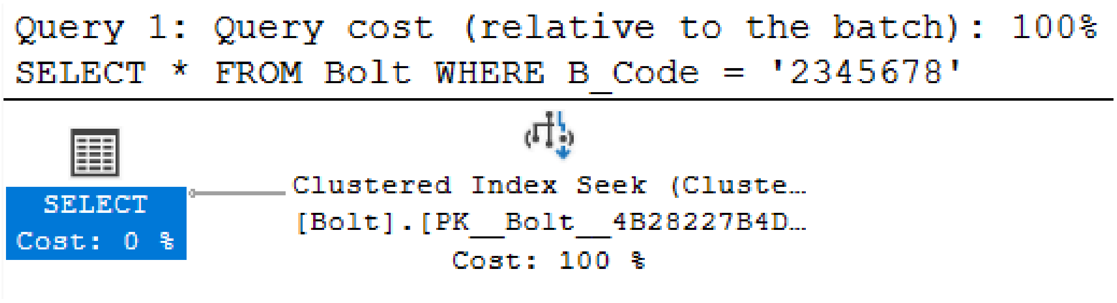
Figure 28: Execution time with table scan force - about 2 seconds

```
SELECT * FROM Bolt WITH (INDEX(0)) WHERE B_Code = '2345678';
```

The result for that query with the execution time is shown in Figure 28. It can be seen that the cluster index make the query executed faster.

6.1.3 Checking the estimated execution plan

Here are the estimated execution plan for the first query with cluster index (Figure 29) and the second one with table scan force (Figure 30).



```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM Bolt WHERE B_Code = '2345678'
```

SELECT
Cost: 0 %
Clustered Index Seek (Cluste...
[Bolt].[PK__Bolt__4B28227B4D...
Cost: 100 %

Figure 29: Estimated execution plan with cluster index

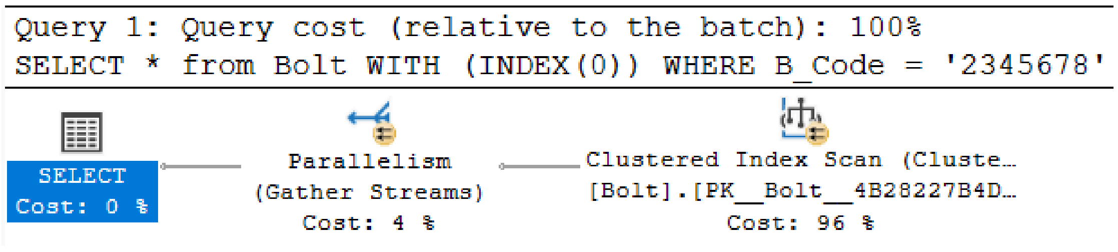

```
Query 1: Query cost (relative to the batch): 100%
SELECT * from Bolt WITH (INDEX(0)) WHERE B_Code = '2345678'
```

Figure 30: Estimated execution plan with table scan force

In all 3 ways, it is proven that the cluster index optimizes the query efficiency.

6.2 Solving one use-case of database security

For this section, we've decided to choose SQL injection as the primary security risk. SQL injection (SQLi) remains one of the most prevalent and impactful security risks in database management. Its enduring threat stems from its ability to exploit vulnerabilities in web applications and manipulate underlying database queries, potentially leading to devastating consequences.

6.2.1 SQLi Recreation

For our application, most of the queries are sanitized and protected, except for the Supplier Creation Form (Screen as in Figure 11), with the query as below (Figure 31).

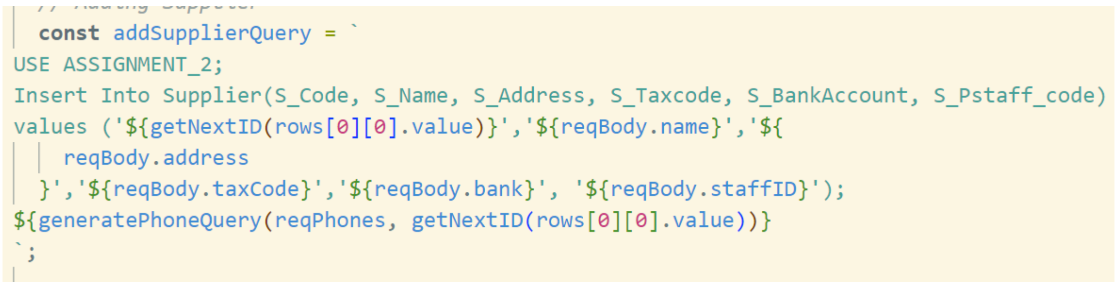
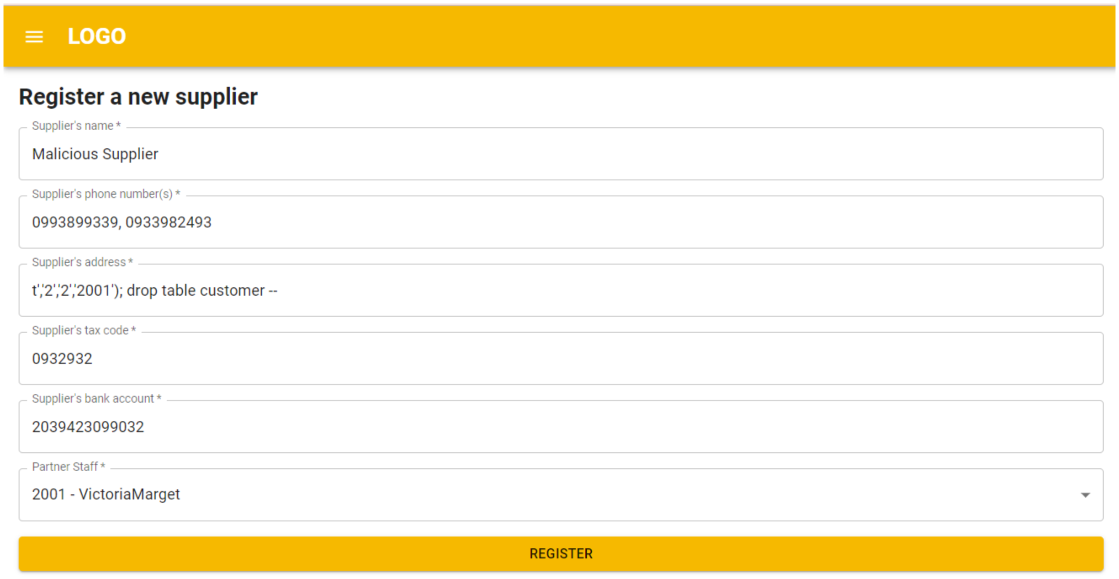

```
  const addSupplierQuery = `
USE ASSIGNMENT_2;
Insert Into Supplier(S_Code, S_Name, S_Address, S_Taxcode, S_BankAccount, S_Pstaff_code)
values ('${getNextID(rows[0][0].value)}','${reqBody.name}','${
    reqBody.address
  }','${reqBody.taxCode}','${reqBody.bank}', '${reqBody.staffID}');
${generatePhoneQuery(reqPhones, getNextID(rows[0][0].value))}
`;
```

Figure 31: A vulnerable query

In this query, the parameters are directly replaced into the query string, with no sanitize process.

On our application interface, if a malicious input in submitted, as in Figure 32, the operation on submit is still successful (Figure 33), however, when checking the database, one of our table is missing (Figure 34).

That is one of the potential impacts that can be caused by SQLi.



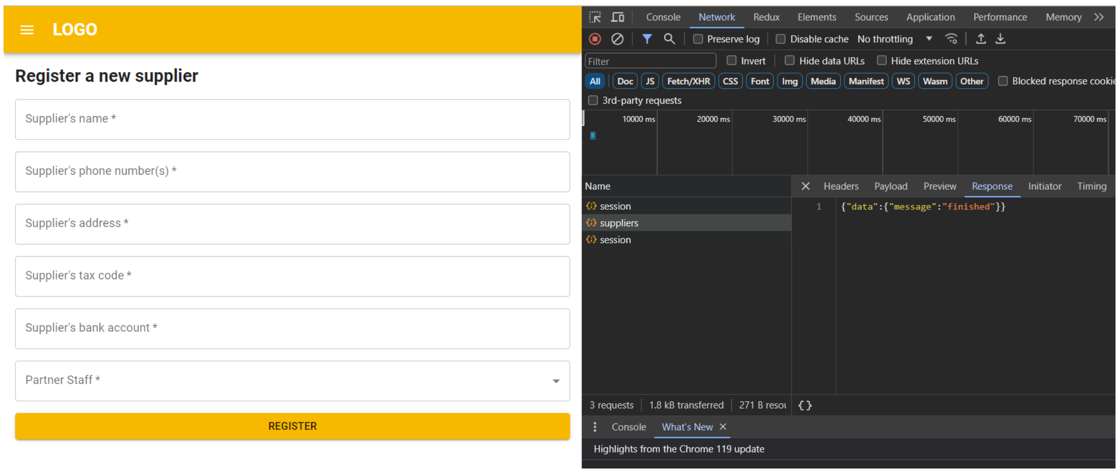
LOGO

Register a new supplier

Supplier's name *	Malicious Supplier
Supplier's phone number(s) *	0993899339, 0933982493
Supplier's address *	t','2','2','2001'); drop table customer --
Supplier's tax code *	0932932
Supplier's bank account *	2039423099032
Partner Staff *	2001 - VictoriaMarget

REGISTER

Figure 32: Malicious Form input

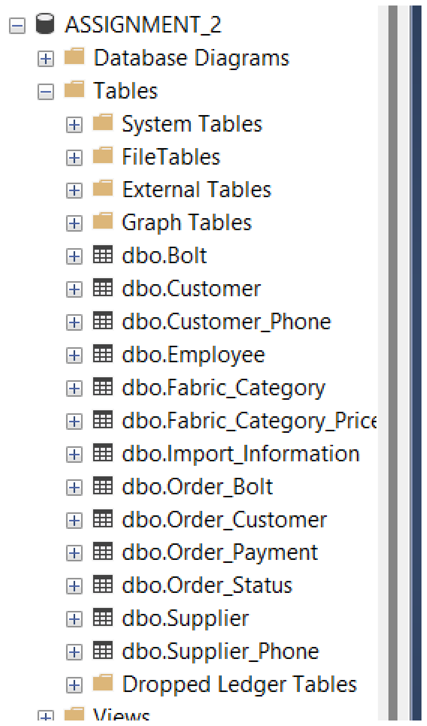


```
{"data":{"message":"finished"}}
```

Figure 33: Response status

6.2.2 Other forms of SQLi

Our use case demonstrated as above is on form of SQL injection, more specifically, a form of SQL Manipulation. Besides, there are several other methods that malicious users can do harm to our system such as Code injection, Function call injection, etc.



```
ASSIGNMENT_2
  Database Diagrams
  Tables
    System Tables
    FileTables
    External Tables
    Graph Tables
    dbo.Bolt
    dbo.Customer
    dbo.Customer_Phone
    dbo.Employee
    dbo.Fabric_Category
    dbo.Fabric_Category_Price
    dbo.Import_Information
    dbo.Order_Bolt
    dbo.Order_Customer
    dbo.Order_Payment
    dbo.Order_Status
    dbo.Supplier
    dbo.Supplier_Phone
    Dropped Ledger Tables
  Views
```

Figure 34: Missing table in database

6.2.3 Impacts

Once our system database is exposed to the attackers, there will be immeasurable impacts that can ruin our service and database. Some of them are:

- **Database fingerprinting**: By injecting specific code into database queries, attackers can gather information about the database schema, version, and even the operating system it runs on. This information can be valuable for planning further attacks and exploiting additional vulnerabilities.

- **Denial of Service (DOS)**: Attackers can trigger resource exhaustion on the database server by crafting malicious queries that consume excessive CPU, memory, or network bandwidth. This can render the database inaccessible to legitimate users, effectively disrupting website functionality and service availability.

- **Bypassing authentication**: By manipulating login credentials or injecting malicious code into authentication queries, attackers can gain unauthorized access to the database even without valid credentials. This can allow them to view sensitive information, modify data, or even take control of the server entirely.

- **Identifying injectable parameters**: Attackers often rely on automated tools to identify vulnerable input fields within web applications where they can inject malicious code. This allows them to efficiently target and exploit multiple applications with minimal effort.
- **Executing remote commands**: In some instances, SQL injection vulnerabilities can be exploited to execute arbitrary commands on the underlying operating system. This grants attackers significant control over the system, allowing them to install malware, steal data, or launch further attacks.
- **Privilege escalation**: By exploiting vulnerabilities within the database's access control mechanisms, attackers can escalate their privileges and gain administrative access to the entire system. This allows them to perform any action they wish, including modifying sensitive data, deleting critical files, or even taking complete control of the database server.

6.2.4 Precautions and mitigation

Some notable protections to SQLi can be:

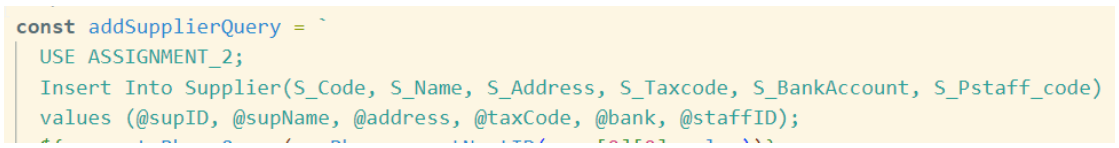
- **Binding variables**: Binding variables, also known as parameterized queries, are a powerful technique for preventing SQL injection. Instead of directly embedding user input into SQL statements, developers use placeholders for the data, which are then bound to specific values during query execution. This ensures that user input is treated as data, preventing it from being interpreted as malicious code.
- **Filtering inputs**: Filtering user input before passing it to the database can effectively remove potentially harmful characters and code snippets. By implementing whitelisting or blacklisting techniques, developers can restrict the type of data allowed in a specific field, significantly reducing the risk of SQL injection vulnerabilities.
- **Function security**: Using secure database functions and stored procedures can further enhance protection against SQL injection. These functions are designed to accept specific data formats and handle user input securely, preventing malicious code from being injected into the database. Additionally, limiting the privileges of database accounts and restricting the use of potentially dangerous functions can further minimize the risk of successful attacks.

6.2.5 Approach to the SQLi of our system

To solve our issue, we can perform a variable binding by utilizing the `tedious` library's capability. The new query Request can be re-written as in Figure 35.

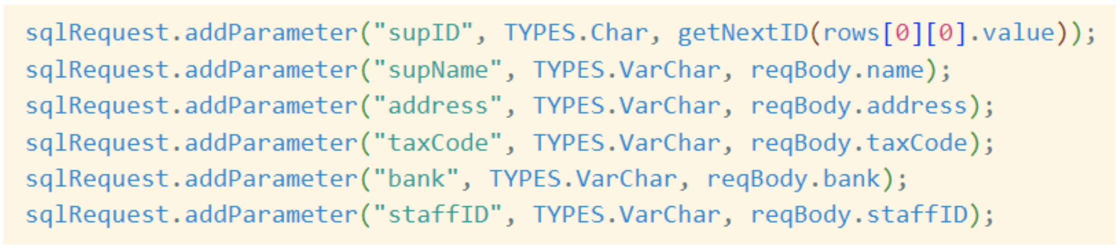
Each variable (parameter) is replaced with a special string having an '@' right before them. After that, we can bind our actual data to them as in Figure 36.

As the result, all malicious input will be treated as string, leaving almost no door for attackers to bypass this setting. For example, if the attacker tries to input the same input as earlier, our



```
const addSupplierQuery = `
  USE ASSIGNMENT_2;
  Insert Into Supplier(S_Code, S_Name, S_Address, S_Taxcode, S_BankAccount, S_Pstaff_code)
  values (@supID, @supName, @address, @taxCode, @bank, @staffID);
```

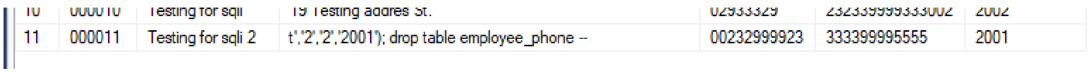
Figure 35: New string query



```
sqlRequest.addParameter("supID", TYPES.Char, getNextID(rows[0][0].value));
sqlRequest.addParameter("supName", TYPES.VarChar, reqBody.name);
sqlRequest.addParameter("address", TYPES.VarChar, reqBody.address);
sqlRequest.addParameter("taxCode", TYPES.VarChar, reqBody.taxCode);
sqlRequest.addParameter("bank", TYPES.VarChar, reqBody.bank);
sqlRequest.addParameter("staffID", TYPES.VarChar, reqBody.staffID);
```

Figure 36: Variable binding

database treats them as normal string and the flow is normal (new supplier added to the table) (Figure 37).



10	000010	Testing for sqli	19 Testing addres St.	02933329	2323339999333002	2002
11	000011	Testing for sqli 2	t','2','2','2001'); drop table employee_phone --	00232999923	333399995555	2001

Figure 37: Failed attach attempt

6.2.6 Reasoning

There are a few reasons why we choose this approach instead of input validation, or function security:

- **Reliable library support**: Instead of risking ourselves on input validation from scratch, we can make use of the well-made library's capability to implement our security measures. However, this action still need to be considered thoroughly, and backed by enough assessment in the library's safety. In other words, our database is foolproof only when the library secured.
- **Enhanced user experience**: Specifically in our use case, the SQLi occurs in the supplier creation form, where the manager has to fill in information about a new partner, we reckon that limiting what the user can insert may take a big toll on the application experience. It is not rare to see the supplier's name or streets in addresses contains some single quotes or special characters. Therefore, taking this into account, we refrained ourselves from restricting the input.