Sprint 1- Software Engineering

“Bookstore Automation System”

BSCS-VI

Group 2

Submitted to:

Mam Aatka Ali

Submitted by:

Noor us Sabah	161101
Mahnoor Jamil	161105
Fabyha Khan	161113
Rabia Kanwal	161118
Noor Fatima	161137
Maham Javad	161151
Amna Azhar	161366

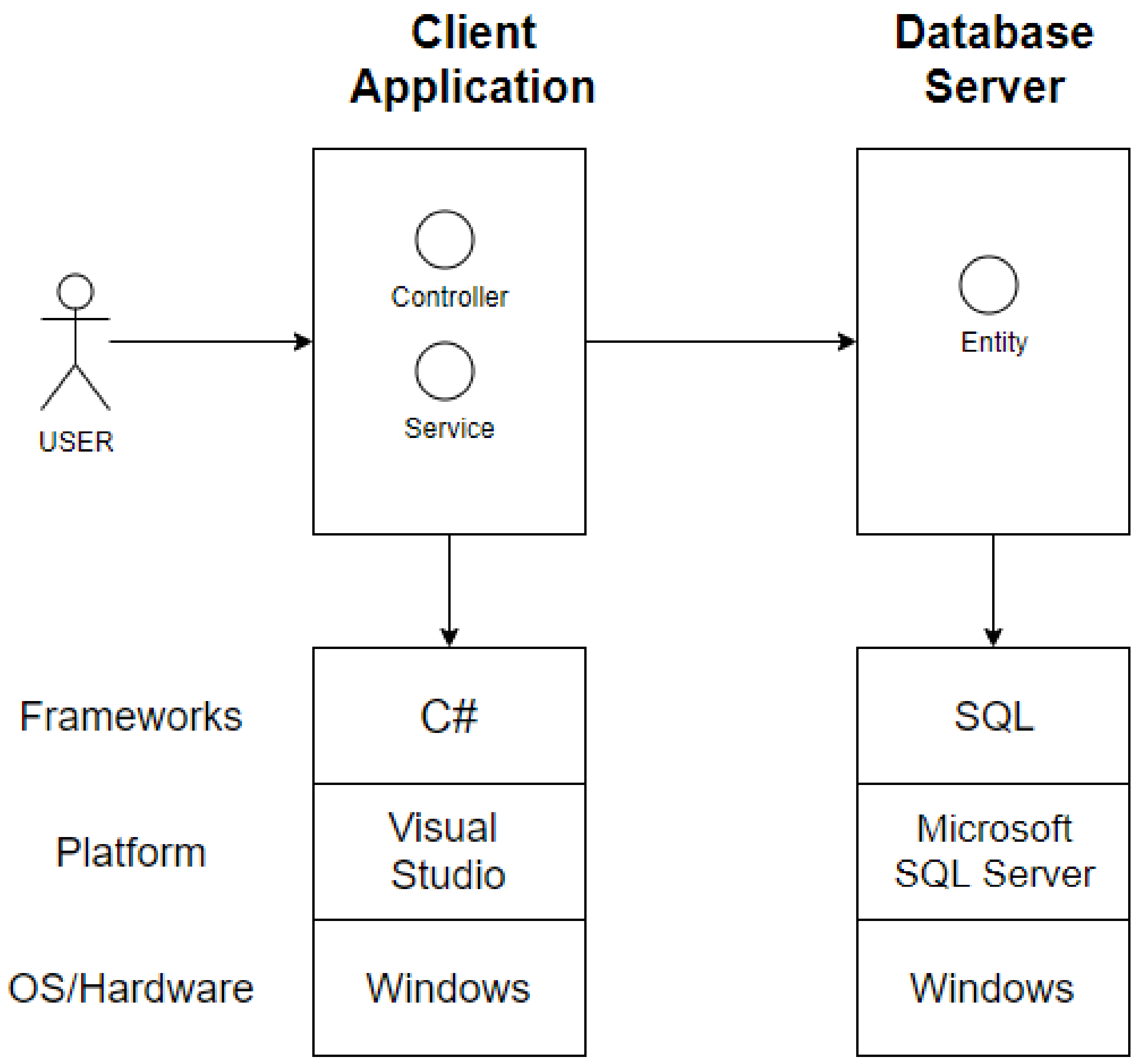
Submission Date:

5th April 2019

Architecture and Development

This section of the document will provide the details of the architecture of Bookstore Automation System and its implementation.

Project Architecture:



List of component responsibilities:

Client Application	Database Server
• Displays data • Modifies data • Communicate business rules to user • Provides all user interaction • Communicates with database server to process information • Provide system to input information • Display information	• Receive request from user • Manages data interaction • Enforces data integrity • Provide data security • Controlling and monitoring user access to the database • Planning for backup and recovery of database information • Maintaining archived data • Generating various reports by querying from database as per need • Managing and monitoring data replication

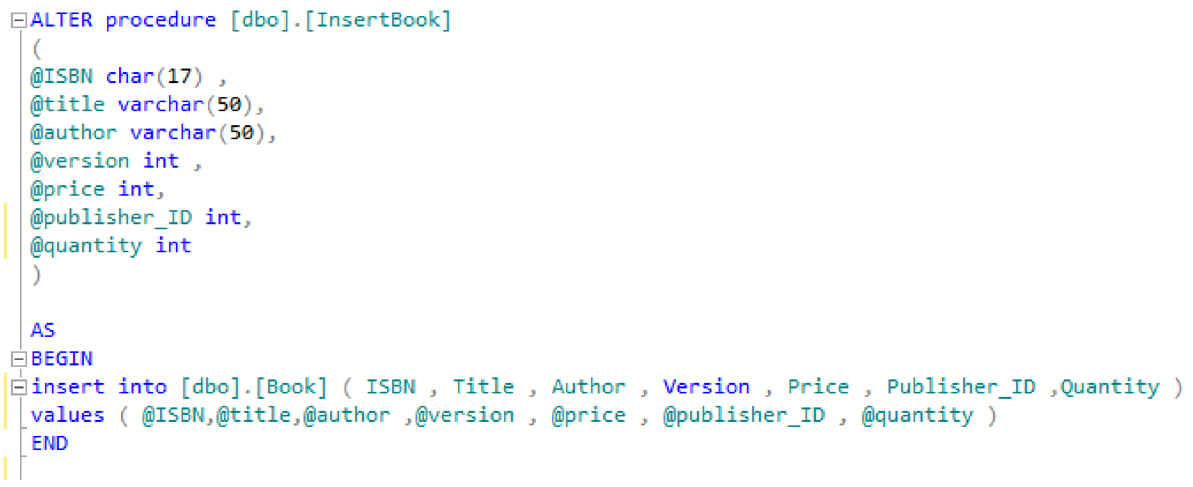
Implementation:

Stored Procedures

For the Book table, the following are stored procedures for insert, delete and updating rows.

Insert:

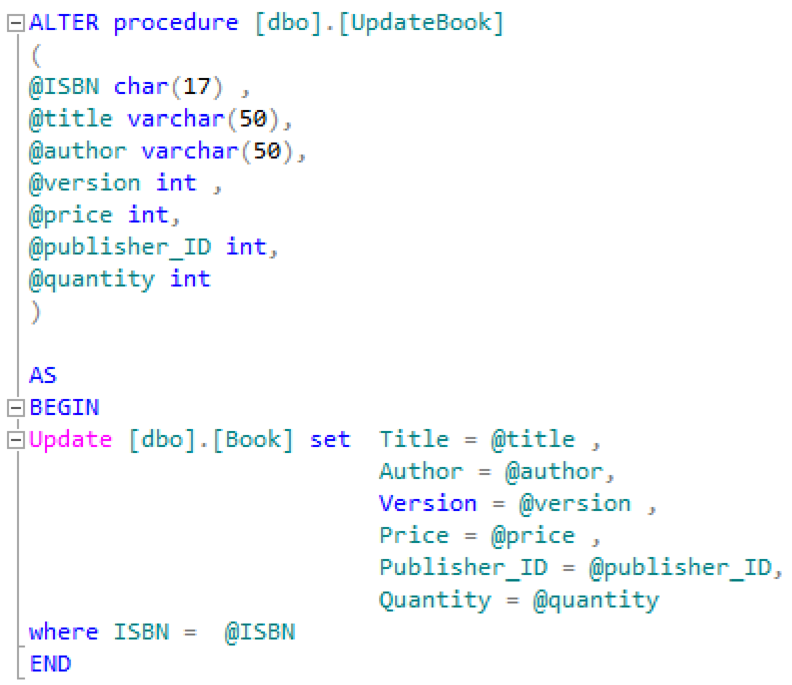
The InsertBook stored procedure takes the ISBN, title, author name, version, price, publisher ID and quantity of a book as its parameter and inserts the details of that book in the Book table.



```
ALTER procedure [dbo].[InsertBook]
(
@ISBN char(17) ,
@title varchar(50),
@author varchar(50),
@version int ,
@price int,
@publisher_ID int,
@quantity int
)

AS
BEGIN
insert into [dbo].[Book] ( ISBN , Title , Author , Version , Price , Publisher_ID ,Quantity )
values ( @ISBN,@title,@author ,@version , @price , @publisher_ID , @quantity )
END
```

Update:

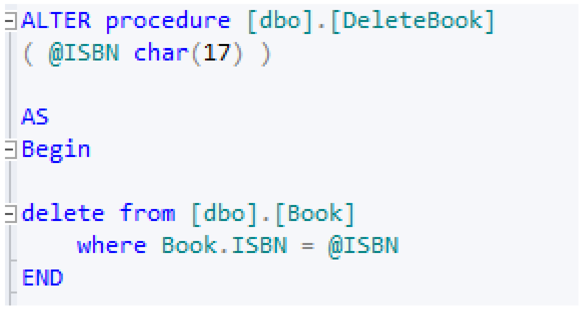
The UpdateBook stored procedure takes the ISBN, title, author name, version, price, publisher ID and quantity of a book as its parameter and updates the details of book for the specified ISBN in the Book table.



```sql
ALTER procedure [dbo].[UpdateBook]
(
@ISBN char(17) ,
@title varchar(50),
@author varchar(50),
@version int ,
@price int,
@publisher_ID int,
@quantity int
)

AS
BEGIN
Update [dbo].[Book] set  Title = @title ,
                         Author = @author,
                         Version = @version ,
                         Price = @price ,
                         Publisher_ID = @publisher_ID,
                         Quantity = @quantity
where ISBN =  @ISBN
END
```

Delete:

The DeleteBook stored procedure takes the ISBN of book as the parameter and deletes that book’s details from Book table.

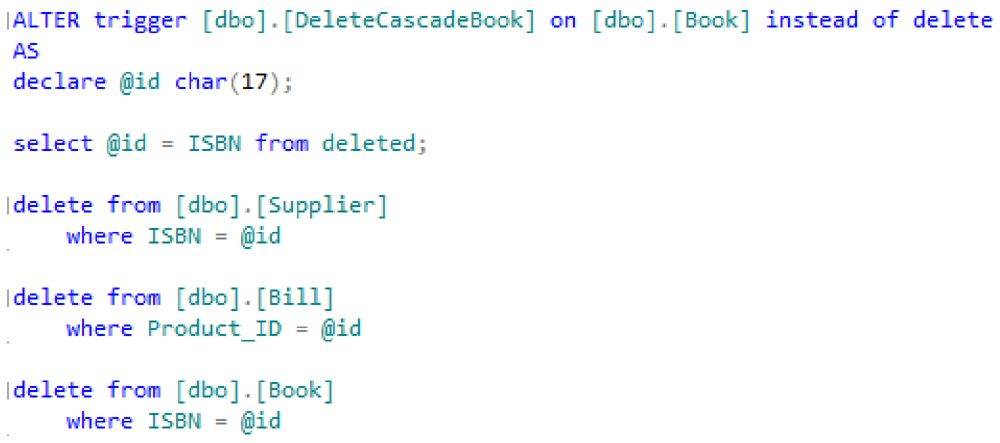


```sql
ALTER procedure [dbo].[DeleteBook]
( @ISBN char(17) )

AS
Begin

delete from [dbo].[Book]
    where Book.ISBN = @ISBN
END
```

Triggers

1) DeleteCascadeBook

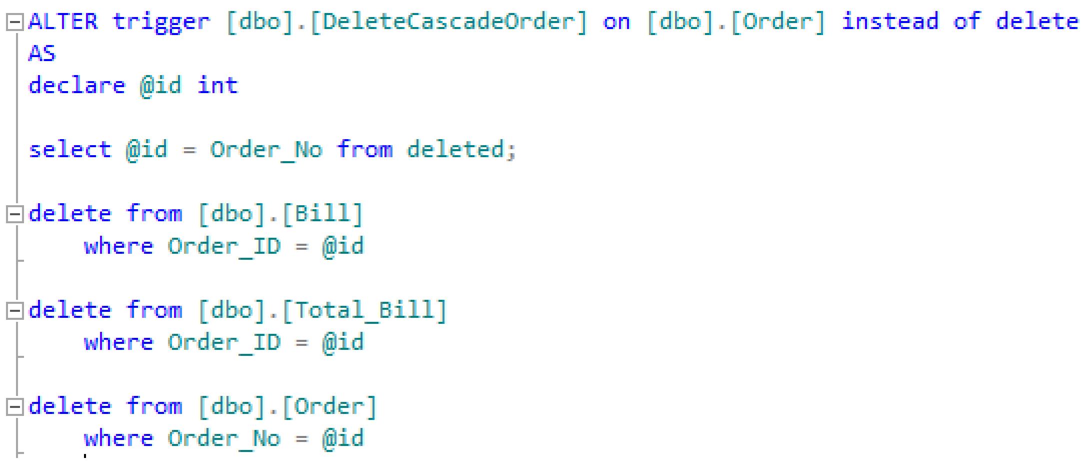
This trigger performs cascade delete on the Book table and maintains the referential integrity. Whenever a record for a book is deleted in the Book table, ‘DeleteCascadeBook’ is fired and deletes the corresponding rows from the tables dependent on Book table (Bill and Supplier table).



```
ALTER trigger [dbo].[DeleteCascadeBook] on [dbo].[Book] instead of delete
AS
declare @id char(17);

select @id = ISBN from deleted;

delete from [dbo].[Supplier]
    where ISBN = @id

delete from [dbo].[Bill]
    where Product_ID = @id

delete from [dbo].[Book]
    where ISBN = @id
```

2) DeleteCascadeOrder

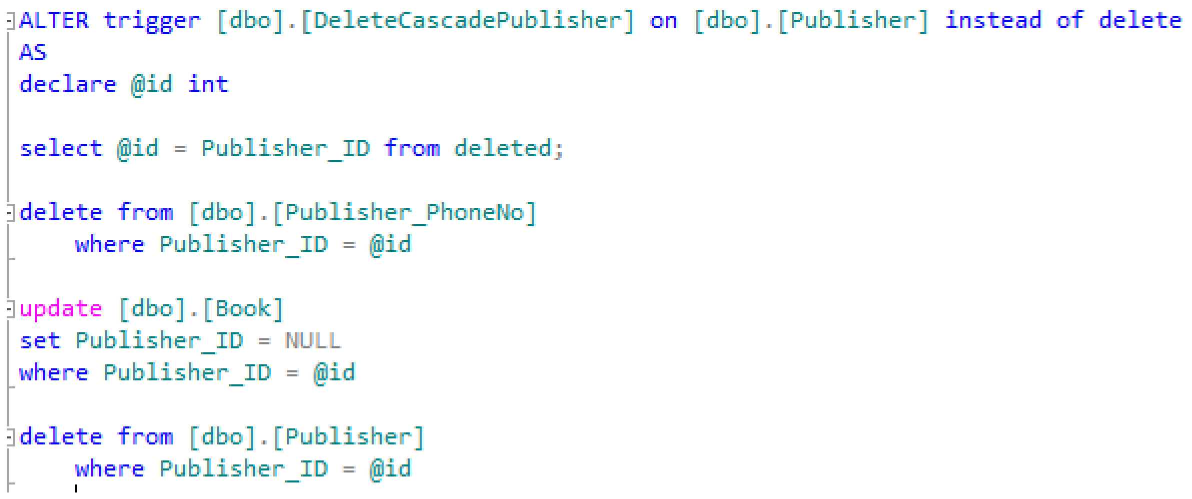
This trigger performs cascade delete on the Order table and maintains the referential integrity. Whenever a record for an order is deleted in the Order table, ‘DeleteCascadeOrder’ is fired and deletes the corresponding rows from the tables dependent on Order table (Bill and Total_Amount table).



```
ALTER trigger [dbo].[DeleteCascadeOrder] on [dbo].[Order] instead of delete
AS
declare @id int

select @id = Order_No from deleted;

delete from [dbo].[Bill]
    where Order_ID = @id

delete from [dbo].[Total_Bill]
    where Order_ID = @id

delete from [dbo].[Order]
    where Order_No = @id
```

3) DeleteCascadePublisher

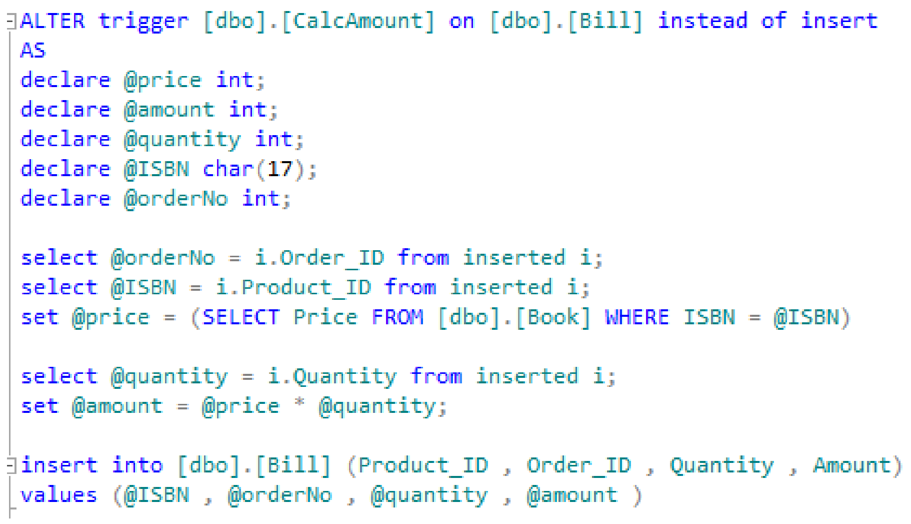
This trigger performs cascade delete on the Publisher table and maintains the referential integrity. Whenever a record for a publisher is deleted in the Publisher table, ‘DeleteCascadePublisher’ is fired and deletes the corresponding rows from the dependent table, Publisher_PhoneNo, and sets the Publisher_ID to NULL in the Book table.



```
ALTER trigger [dbo].[DeleteCascadePublisher] on [dbo].[Publisher] instead of delete
AS
declare @id int

select @id = Publisher_ID from deleted;

delete from [dbo].[Publisher_PhoneNo]
    where Publisher_ID = @id

update [dbo].[Book]
set Publisher_ID = NULL
where Publisher_ID = @id

delete from [dbo].[Publisher]
    where Publisher_ID = @id
```

4) CalcAmount

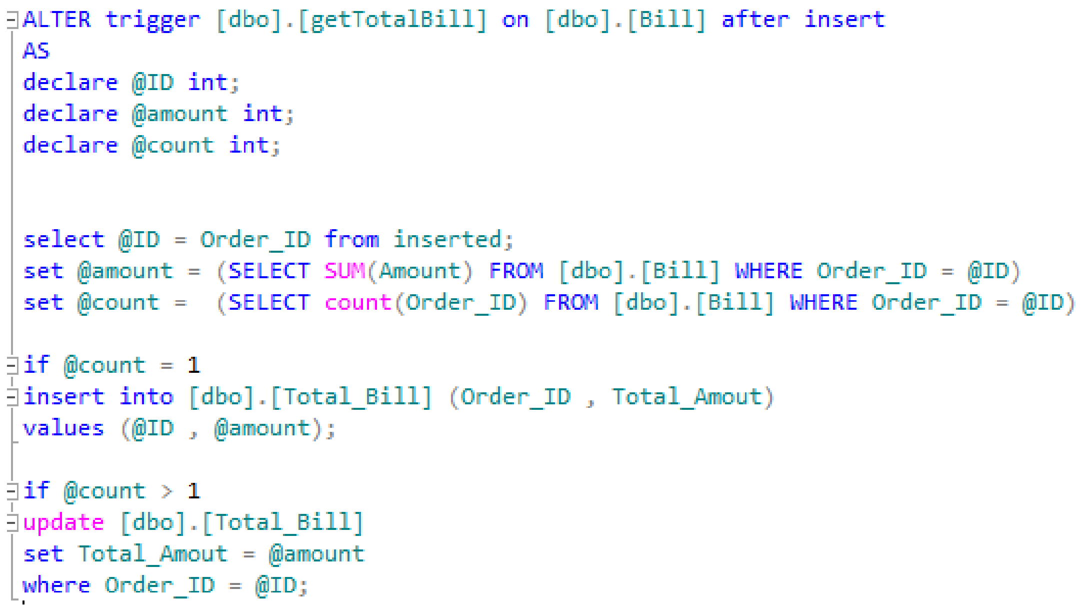
Whenever we insert Order_ID , Product_ID and Quantity for the bill in the Bill table, the CalcAmount trigger is fired and the Amount is calculated by multiplying the Quantity and Price of the book and stored in the Bill table.



```
ALTER trigger [dbo].[CalcAmount] on [dbo].[Bill] instead of insert
AS
declare @price int;
declare @amount int;
declare @quantity int;
declare @ISBN char(17);
declare @orderNo int;

select @orderNo = i.Order_ID from inserted i;
select @ISBN = i.Product_ID from inserted i;
set @price = (SELECT Price FROM [dbo].[Book] WHERE ISBN = @ISBN)

select @quantity = i.Quantity from inserted i;
set @amount = @price * @quantity;

insert into [dbo].[Bill] (Product_ID , Order_ID , Quantity , Amount)
values (@ISBN , @orderNo , @quantity , @amount )
```

5) getTotalBill

After the values for a single entry have been inserted for a particular Order_ID in the Bill table and more items have been added in the Bill for the same Order_ID, ‘getTotalBill’ trigger is fired and the total amount for the corresponding Order_ID is calculated by adding all the amounts and stored in the Total_Amount table. If no entry exists for the Order_ID in the Total_Amount table, a new entry is inserted. Else, the total_amount is updated.



```
ALTER trigger [dbo].[getTotalBill] on [dbo].[Bill] after insert
AS
declare @ID int;
declare @amount int;
declare @count int;


select @ID = Order_ID from inserted;
set @amount = (SELECT SUM(Amount) FROM [dbo].[Bill] WHERE Order_ID = @ID)
set @count =  (SELECT count(Order_ID) FROM [dbo].[Bill] WHERE Order_ID = @ID)

if @count = 1
insert into [dbo].[Total_Bill] (Order_ID , Total_Amout)
values (@ID , @amount);

if @count > 1
update [dbo].[Total_Bill]
set Total_Amout = @amount
where Order_ID = @ID;
```

6) updateQuantity

Whenever a purchase is made for a particular book, the quantity of books bought is subtracted from the quantity of books in stock and the quantity of that book is updated in the Book table.

```sql
ALTER trigger [dbo].[updateQuantity] on [dbo].[Bill] after insert
AS
declare @old int;
declare @new int;
declare @updated_quantity int;
declare @ISBN char(17);

select @ISBN = i.Product_ID from inserted i;
set @old = (SELECT Quantity FROM [dbo].[Book] WHERE ISBN = @ISBN)
select @new = i.Quantity from inserted i;
set @updated_quantity = @old - @new


update [dbo].[Book]
set Quantity = @updated_quantity
where ISBN = @ISBN
```

7) InsertQuantity

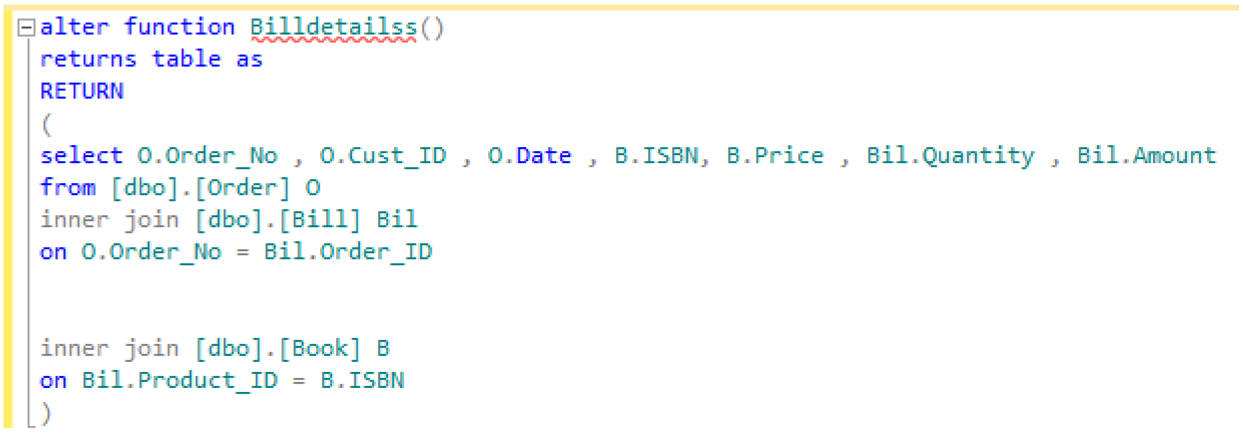
For the following Supplier and Book tables, whenever we insert or update a row for a particular book (identified by ISBN) in the Supplier table, the quantity of that book in Book table is added and updated.

```sql
ALTER trigger [dbo].[InsertQuantity] on [dbo].[Supplier] after insert , update
AS
declare @old int;
declare @new int;
declare @updated_quantity int;
declare @ISBN char(17);
declare @ID int;


select @ISBN = i.ISBN from inserted i;
select @ID = i.Vendor_ID from inserted i;
set @old = (SELECT Quantity FROM [dbo].[Book] WHERE ISBN = @ISBN)
select @new = i.Quantity from inserted i;
set @updated_quantity = @old + @new


update [dbo].[Book]
set Quantity = @updated_quantity
where ISBN = @ISBN
```

Stored Functions

1) Billdetailss()

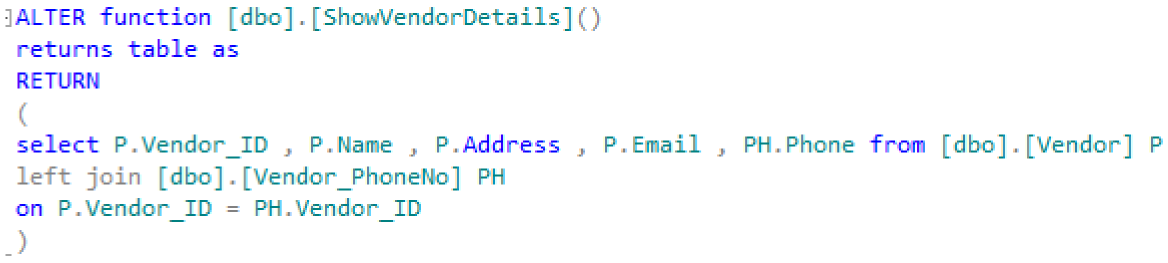
This function joins the Bill and Book table to show the bill detail.



```
alter function Billdetailss()
returns table as
RETURN
(
select O.Order_No , O.Cust_ID , O.Date , B.ISBN, B.Price , Bil.Quantity , Bil.Amount
from [dbo].[Order] O
inner join [dbo].[Bill] Bil
on O.Order_No = Bil.Order_ID


inner join [dbo].[Book] B
on Bil.Product_ID = B.ISBN
)
```

2) ShowVendorDetails()

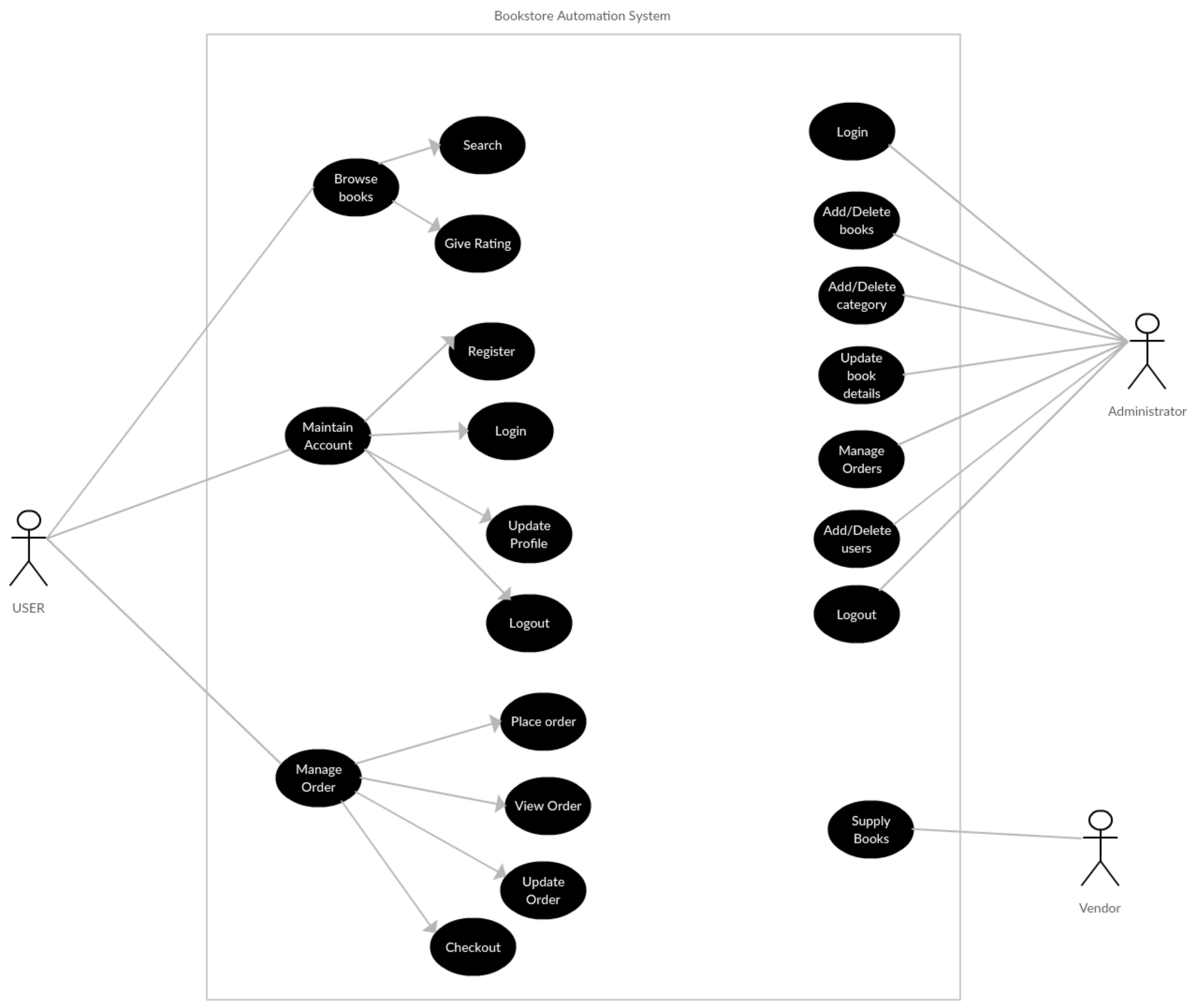
This function joins the Vendor table with Vendor_PhoneNo table to display the details of vendor in a single table.



```
ALTER function [dbo].[ShowVendorDetails]()
returns table as
RETURN
(
select P.Vendor_ID , P.Name , P.Address , P.Email , PH.Phone from [dbo].[Vendor] P
left join [dbo].[Vendor_PhoneNo] PH
on P.Vendor_ID = PH.Vendor_ID
)
```

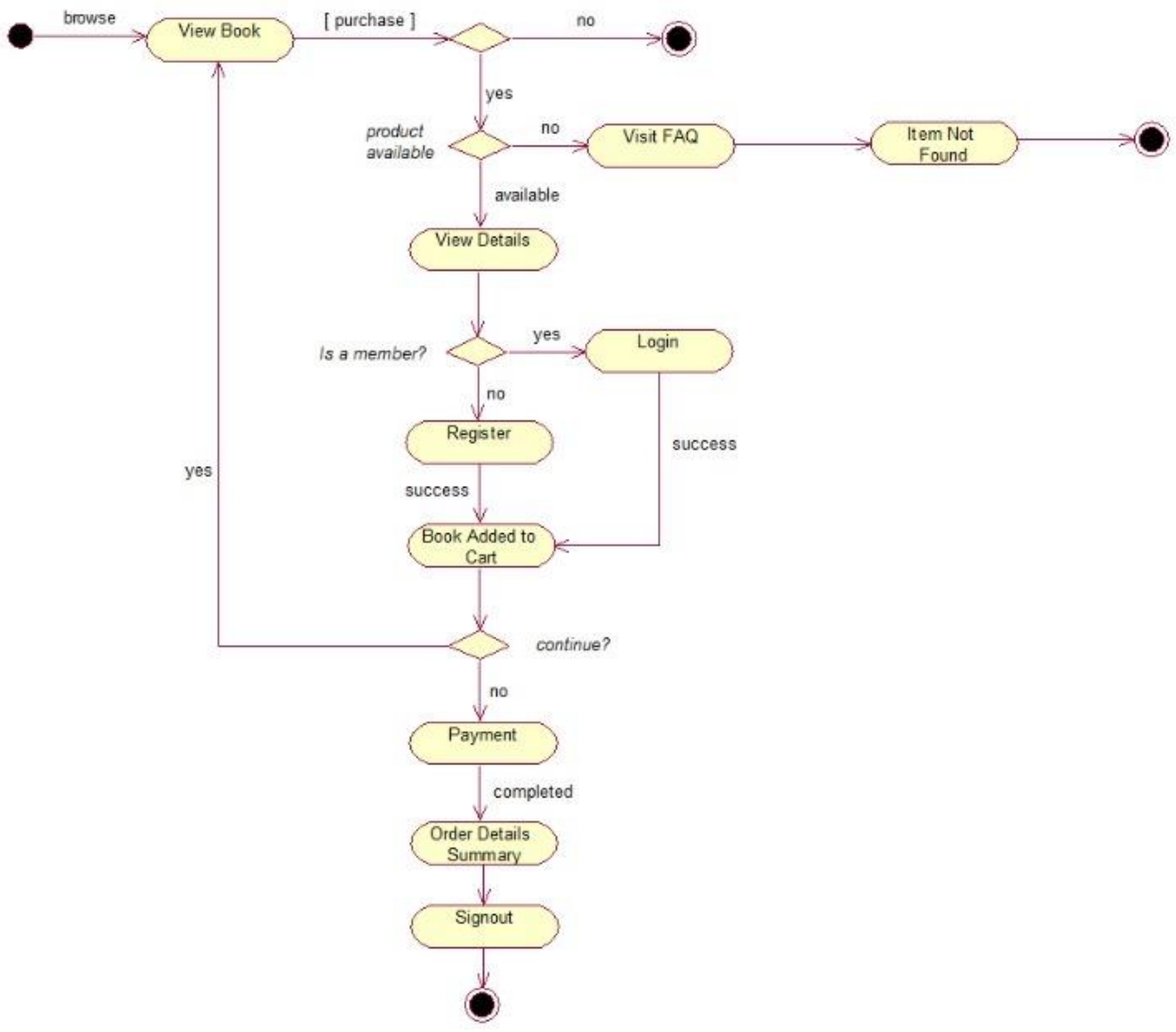
Domain Driven Design

The following diagram gives the use case for the system:



State Based Behavior

Statechart for user processes:


browse
View Book
[purchase]
no
yes
product available
no
Visit FAQ
Item Not Found
available
View Details
Is a member?
yes
Login
no
Register
success
success
yes
Book Added to Cart
continue?
no
Payment
completed
Order Details Summary
Signout