# Project Overview

**Problem:**

You have be contracted by The Gallery, an art gallery that specializes in abstract art. The Gallery's clients include Corporate offices. Upscale hotels and individuals art patrons. The Gallery has recently opened offices in London England and Sydney Australia. With the international expansion The Gallery needs an updated Internet web site to manage their data. The Galley has an existing Database with records covering Artists, Customers, Billing, Invoices, Suppliers, and Inventory. You have been requested to design and construct this new Internet web site.

There are three Roles defined for Users of the new system:

- Visitor ← A non-employee who will viewing Artist Profiles, and Inventory items
- Clerk ← An Employee who will manage most database records
- Manager ← An Employee who will set prices for Inventory items, and manage Suppliers

**Use Cases Overview:**

Visitor: View Artist Profile
View Inventory

Clerk: Sign-on to the System
Sell Inventory
Add/Update an Artist Profile
Add/Update an Inventory Item
Add/Update a Customer
View/Update an Invoice

Manager: Add/Update/Delete a Supplier
Price data for Inventory Item

Batch: Receive Inventory

Notes:
Clerks can use all the Visitor use cases, Managers can use all the Clerk use cases.

**Use Case Details:**

## View Artist Profile

Pre-Conditions: The Visitor is viewing a Welcome screen, with a menu option to View Artist Profile
or the Visitor is viewing an Inventory Item for a specific Artist
Artist records exist in database

Post-Conditions: The Visitor is viewing an Artist Profile

Steps:

1. The Visitor selects "**View Artist Profile**" from the main menu or from the **View Inventory** page
2. The System display a list of Artists for the Visitor to select from
3. The Visitor selects an Artist from the displayed list
4. The System displays the Artist Profile, and provides an option to list the Inventory items from the selected Artist
5. If the Visitor selects the option to list Inventory items, invoke the **View Inventory Item** use case.

**View Inventory Item**

Pre-Conditions: The Visitor is viewing a Welcome screen, with a menu option to View Inventory Item,
or the Visitor is viewing a specific Artist Profile
Inventory records exist in database

Post-Conditions: The Visitor is viewing an Inventory Item

Steps:

1. The Visitor selects "**View Inventory Item**" from the main menu, or from the **View Artist Profile** page
2. The System display a list of Inventory Items for the Visitor to select from, if the Visitor comes from the main menu, the Inventory list displays all Inventory Items, if the Visitor comes from the View Artist Profile page, the Inventory list displays items specific to the selected Artist
3. The Visitor selects an Inventory Item from the displayed list
4. The System displays the Inventory Item picture, and details and provides an option to View the Artist Profile
5. If the Visitor selects the option to View Artist Profile, invoke the **View Artist Profile** use case.

**Sign-on to the System**

Pre-Conditions: The Employee has been assigned a UserID and Password

Post-Conditions: The Employee is signed on the system
The Employee Role has been determined

Steps:

1. The Employee enters their UserID and Password
2. The System invokes a one-way hash algorithm on the Password to avoid sending the password in the clear
3. The System compares the UserID and Password hash to the assigned UserID and Password hash
4. On a valid signon the System returns the Employee Role
5. Invalid signon returns an error

**Sell Inventory Items**

Pre-Conditions: The Employee is signed into the system and viewing an Admin screen,
with a menu option to Sell Inventory Item
Inventory records exist in database
Customer Records exits in database

Post-Conditions: The Inventory.Number_in_Stock field has been decremented
An Invoice has been created with Customer identified, and Billing_Address set
Invoice Details has been created and attached to the Invoice

Steps:

1. The Employee selects "**Sell Inventory Item**" from the Admin menu
2. The System Displays a list of Customers
3. The Employee selects the Customer for this purchase, and the Customer Address for shipping
4. The Employee selects the Billing_Information for this Customer
5. The System creates an Invoice for this purchase
6. The System display a list of Inventory Items for the Employee to select from
7. The Employee selects an Inventory Item from the displayed list
8. The System displays the Inventory Item picture, and details and provides an option to confirm the purchase
9. The Employee confirms the Inventory Item to be added to this purchase
10. The System creates a new Invoice_Details for each selected Inventory item, decrementing the Inventory.Number_in_Stock and adding the cost to the Invoice.Total_Cost, if the Inventory.Number_in_Stock = 0, the Invoice_Details.Back_Ordered is set to true.
11. The System asks the Employee if the Customer wishes to add Inventory items to the Invoice, if yes repeat steps 5-10
12. The System computes the Invoice.Total_Cost to include shipping costs
13. The System display the completed Invoice

**Add/Update an Artist Profile**

Pre-Conditions: The Employee is signed into the system and viewing an Admin screen, with a menu option to Add/Update an Artist Profile

Post-Conditions: The Artist Profile has been added or updated

Steps:

1. The Employee selects "**Add/Update an Artist Profile**" from the Admin menu
2. The System display a list of Artist Profiles, and an option to "**Add An Artist Profil**e" for the Employee to select from
3. If the Employee selects an existing Artist Profile, the System displays the current data, if the Employee selects to add a new Artist Profile, the System display a blanks screen to enter the data
4. The Employee edits/enters the Artist Profile data
5. The System updates or adds the Artist Profile data

**Add/Update an Inventory Item**

Pre-Conditions: The Employee is signed into the system and viewing a Admin screen, with a menu option to Add/Update an Inventory Item

Post-Conditions: The Inventory Item has been added or updated

Steps:

1. The Employee selects "**Add/Update an Inventory Item**" from the Admin menu
2. The System display a list of Inventory items, and an option to "**Add A New Inventory Item**" for the Employee to select from
3. If the Employee selects an existing Inventory Item, the System displays the current data, if the Employee selects to add a new Inventory Item, the System display a blanks screen to enter the data, including the ability to upload a new graphic
4. The Employee edits/enters the Inventory Item data
5. The System updates or adds the Inventory Item data

**Add/Update a Customer Record**

Pre-Conditions: The Employee is signed into the system and viewing an Admin screen, with a menu option to Add/Update a Customer Record

Post-Conditions: The Customer has been added or updated

Steps:

1. The Employee selects "**Add/Update a Customer Record**" from the Admin menu
2. The System display a list of Customers, and an option to "**Add A New Customer**" for the Employee to select from
3. If the Employee selects an existing Customer, the System displays the current data, if the Employee selects to add a new Customer, the System display a blanks screen to enter the data
4. The Employee edits/enters the Customer data
5. The System updates or adds the Customer data

**View/Update Invoice Record**

Pre-Conditions: The Employee is signed into the system and viewing an Admin screen,
with a menu option to View/Update Invoice Record
Invoice and Invoice_Details records exist

Post-Conditions: The Invoice and/or Invoice_Detail Records have been viewed or updated

Steps:

1. The Employee selects "**View/Update Invoice Record**" from the Admin menu
2. The System display a list of Invoices for the Employee to select from
3. The Employee selects an Invoice
4. The System displays the current Invoice data, and the Invoice_Details data
5. The Employee edits the Invoice and/or Invoice_Details data
6. The System updates the Database with the new data

**Add/Update/Delete a Supplier**

Pre-Conditions: The Employee is signed into the system and viewing an Admin screen, with a menu option to Add/Update/Delete a Supplier

Post-Conditions: The Supplier has been added/updated/Deleted

Steps:

1. The Employee selects "**Add/Update/Delete a Supplier**" from the Admin menu
2. The System display a list of Suppliers, and an option to "**Add A Supplier**" for the Employee to select from
3. If the Employee selects an existing Supplier, the System displays the current data, if the Employee selects to add a new Supplier, the System display a blanks screen to enter the data
4. The Employee edits/enters the Supplier data, or selects "**Delete Supplier**" option
5. If the Employee selected the Delete Supplier option, the System asks for the Employee to confirm the deletion
6. If the Employee is adding or updating a Supplier, the System updates or adds the Supplier data

**Update Price Data for Inventory Item**

Pre-Conditions: The Employee is signed into the system and viewing an Admin screen, with a menu option to Update Price Data for Inventory Item

Post-Conditions: The Inventory.Unit_Cost has been updated

Steps:

1. The Employee selects "**Update Price Data for Inventory Item**" from the Admin menu
2. The System display a list of Inventory items for the Employee to select from
3. The Employee selects an Inventory item,
4. The System displays the current data
5. The Employee edits the Inventory Item Price data
6. The System updates the Inventory.Unit_Price data

**Receive Inventory**

Pre-Conditions: A data containing the incoming Inventory details in the following format:
ItemNumber,NumberOfItems
A results directory to place the output reports

Post-Conditions: The Inventory.Number_In_Stock has been updated
A ReceiveInventoryResults.xml file is created in the results directory
An optional ReceiveInventoryError.txt is created if need to hold processing errors

Steps:

1 The program opens the input file
2 For each record in the input file
  2.1 The program finds the Inventory record
  2.2 The Inventory record is update to add the incoming NumberOfItems
  2.3 An entry is added to the results file showing the original Number_in_Stock and the updated Number_In_Stock
3 Any Errors are written to the error file
4 The results and error files are written to the results directory

**Business Rules:**

- Within the database: true = 1, false = 0
- Suppliers with Inventory records cannot be deleted.
- Inventory Items that have been sold (Invoices exist) cannot be deleted.
- Invoices.Total_Cost = sum((Inventory.Unit_Cost * Invoice_Details.Quantity_Ordered) for each Inventory Item on Invoice)
- An Invoice must have at least one Invoice_Details record, but may have multiple Invoice_Details records
- Shipping Cost (Added to Invoice.Total_Cost):
  - United States of America Customers is 2% of Invoice.Total_Cost
  - Canada or Mexico is 3% of Invoice.Total_Cost
  - Australia or New Zealand is 10% of Invoice.Total_Cost
  - United Kingdom is 7% of Invoice.Total_Cost
- Invoice.Order_Placed defaults to Invoice creation date
- Customers may have multiple Customer Addresses, but must have at least 1
- Customers may have multiple Billing Information records, but must have at least 1
- The Customer Address on a Billing Information records must belong the same Customer as the Billing Information record
- If the Invoice_Details.Quantity_Ordered > Inventory.Number_In_Stock, then Invoice_Details.Back_Ordered and Invoice.Back_Ordered are set to true
- Inventory.Type = painting or print (note the lowercase)
- Invoices.Paid is initialized to false
- The Gallery does business (Artists, Customers and Suppliers) within the following countries:
  - United States of America
  - Canada
  - Mexico
  - United Kingdom
  - Australia
  - New Zealand
- Billing_Information.Card_Number must be a valid Credit Card format (15 or 16 digits)
- All Costs are in US Dollars
- New Inventory items will have an initial Unit_Price set to -1.0 until a Manager prices the item
- Inventory Items with a price of -1.0 can not be sold or viewed by Visitors
- Postal Codes (A – Alpha, 9 – Numeric, _ – Space):
  - Canada: A9A_9A9 (7 chars)
  - Australia: 9999 (4 chars)
  - New Zealand: 9999 (4 chars)
  - Mexico: 99999 (5 chars)
  - US: 99999 (5 chars)
  - UK:
    - A9_9AA (6 chars)
    - A99_9AA (7 chars)
    - AA99_9AA (8 chars)
    - A9A_9AA (7 chars)
    - AA9A_9AA (8 chars)

**Gallery Data:**

The Gallery Database contains the following schema:

```
create table ARTISTS
(       ARTIST_ID int generated by default as identity(start with 1) not null primary key,
        FIRST_NAME varchar(25) not null,
        LAST_NAME  varchar(25) not null,
        PROFILE text,
        ADDRESS_LINE_1 varchar(40) not null,
        ADDRESS_LINE_2 varchar(40),
        CITY varchar(25) not null,
        REGION_OR_STATE varchar(20) not null,
        COUNTRY varchar(25) not null,
        POSTAL_CODE character(10) not null
        EMAIL varchar(40),
        PHONE varchar(15) not null,
        WEBSITE varchar(40)
);

create table BILLING_INFORMATION
(       BILLING_ID int generated by default as identity(start with 1) not null primary key,
        CUSTOMER_ID int not null,
        BILLING_ADDRESS_ID int not null,
        CARD_NUMBER char(16) not null,
        CARD_HOLDER varchar(60) not null,
        SECURITY_CODE character(4) not null
);

create table CUSTOMER_ADDRESSES
(       ADDRESS_ID int generated by default as identity(start with 1) not null primary key,
        CUSTOMER_ID int not null,
        ADDRESS_LINE_1 varchar(40) not null,
        ADDRESS_LINE_2 varchar(40),
        CITY varchar(25) not null,
        REGION_OR_STATE varchar(20) not null,
        COUNTRY varchar(25) not null,
        POSTAL_CODE character(10) not null
);

create table CUSTOMERS
(       CUSTOMER_ID int generated by default as identity(start with 1) not null primary key,
        TITLE char(10),
        FIRST_NAME varchar(25) not null,
        MIDDLE_INITIAL character(1),
        LAST_NAME varchar(25) not null,
        COMPANY varchar(25),
        EMAIL varchar(40),
        PHONE varchar(15) not null
```

```
);

create table INVENTORY
(       ITEM_ID int generated by default as identity(start with 1) not null primary key,
        SUPPLIER_ID int not null,
        ARTIST_ID int not null,
        NAME varchar(25) not null,
        DESCRIPTION varchar(250) not null,
        IMAGE blob,
        TYPE varchar(12) not null,
        UNIT_COST decimal(7,2) not null,
        NUMBER_IN_STOCK bigint
);

create table INVOICE_DETAILS
(       ENTRY_NUMBER int generated by default as identity(start with 1) not null primary key,
        ORDER_NUMBER int not null,
        ITEM_ID int not null,
        QUANTITY_ORDERED bigint not null,
        BACK_ORDERED smallint not null
);

create table INVOICES
(       ORDER_NUMBER int generated by default as identity(start with 1) not null primary key,
        CUSTOMER_ID int not null,
        BILLING_ID int not null,
        SHIPPING_ADDRESS_ID int not null,
        ORDER_PLACED date not null,
        ORDER_NOTES text,
        ORDER_SHIPPED datetime,
        TOTAL_COST decimal(9,2) not null,
        PAID smallint not null,
        BACK_ORDERED smallint not null
);

create table SUPPLIERS
(       SUPPLIER_ID int generated by default as identity(start with 1) not null primary key,
        COMPANY varchar(25) not null,
        CONTACT_TITLE character(10),
        CONTACT_NAME varchar(60),
        ADDRESS_LINE_1 varchar(40) not null,
        ADDRESS_LINE_2 varchar(40),
        CITY varchar(25) not null,
        REGION_OR_STATE varchar(20) not null,
        COUNTRY varchar(25) not null,
        POSTAL_CODE character(10) not null,
        EMAIL varchar(40),
        PHONE varchar(15) not null,
        WEBSITE varchar(40)
```

```
);

alter table BILLING_INFORMATION add constraint
        foreign key (BILLING_ADDRESS_ID)
        references CUSTOMER_ADDRESSES(ADDRESS_ID);

alter table BILLING_INFORMATION add constraint
        foreign key (CUSTOMER_ID)
        references CUSTOMERS(CUSTOMER_ID);

alter table CUSTOMER_ADDRESSES add constraint
        foreign key (CUSTOMER_ID)
        references CUSTOMERS(CUSTOMER_ID);

alter table INVENTORY add constraint
        foreign key (ARTIST_ID)
        references ARTISTS(ARTIST_ID);

alter table INVENTORY add constraint
        foreign key (SUPPLIER_ID)
        references SUPPLIERS(SUPPLIER_ID);

alter table INVOICE_DETAILS add constraint
        foreign key (ITEM_ID)
        references INVENTORY(ITEM_ID);

alter table INVOICE_DETAILS add constraint
        foreign key (ORDER_NUMBER)
        references INVOICES(ORDER_NUMBER);

alter table INVOICES add constraint
        foreign key (BILLING_ID)
        references BILLING_INFORMATION(BILLING_ID);

alter table INVOICES add constraint
        foreign key (CUSTOMER_ID)
        references CUSTOMERS(CUSTOMER_ID);

alter table INVOICES add constraint
        foreign key (SHIPPING_ADDRESS_ID)
        references CUSTOMER_ADDRESSES(ADDRESS_ID);
```

```
create view V_ARTIST_SUMMARY
 as
 select A.ARTIST_ID,
          A.FIRST_NAME || ' '  || A.LAST_NAME as NAME,
          count(I.ITEM_ID) as INVENTORY_COUNT
 from ARTISTS A
    full outer join
      INVENTORY I
 on A.ARTIST_ID = I.ARTIST_ID
 group by A.ARTIST_ID


create view V_CUSTOMER_SUMMARY
 as
select C.CUSTOMER_ID,
C.FIRST_NAME || ' ' || C.MIDDLE_INITIAL || ' ' || C.LAST_NAME as NAME,
COMPANY,
(select count(*) from CUSTOMER_ADDRESSES
          where CUSTOMER_ID = C.CUSTOMER_ID) as ADDRESS,
(select count(*) from BILLING_INFORMATION
          where CUSTOMER_ID = C.CUSTOMER_ID)as BILLINGS
from CUSTOMERS C


create view V_SUPPLIER_SUMMARY
as
select S.SUPPLIER_ID,
       S.COMPANY,
       S.CONTACT_NAME,
       count(I.ITEM_ID) as INVENTORY_COUNT
from SUPPLIERS S
  full outer join
     INVENTORY I
  on S.SUPPLIER_ID = I.SUPPLIER_ID
  group by S.SUPPLIER_ID
```

```
create procedure SHIP_ORDER(in ORDERNUMBER integer)
          modifies sql data
begin atomic
  declare SHIPDATE integer;
  set(SHIPDATE)=(select count(*)from INVOICES
         where ORDER_NUMBER=ORDERNUMBER and ORDER_SHIPPED IS NULL);
  if SHIPDATE=1 then
        update INVOICES
           set ORDER_SHIPPED=current_timestamp,
                BACK_ORDERED=0
                  where ORDER_NUMBER=ORDERNUMBER;
         update INVOICE_DETAILS
           set BACK_ORDERED=0
                  where ORDER_NUMBER=ORDERNUMBER;
  end if;
  if SHIPDATE<1 then
    signal SQLSTATE '45000' set message_text = 'Invoice already shipped';
  end if;
end

create procedure RECEIVE_INVENTORY(in ITEMID integer, in CNT integer)
          modifies sql data
          dynamic result sets 1
 begin atomic
   declare CHECKER integer;
   declare CURR integer;
   declare CUR cursor for select ORDER_NUMBER from INVOICES
      where ORDER_NUMBER in
        (select ORDER_NUMBER from INVOICE_DETAILS where ITEM_ID =  ITEMID
         and BACK_ORDERED = '1');

   set CHECKER = select count(*) from INVENTORY where ITEM_ID = ITEMID;

   if  CHECKER < 1 then
      signal SQLSTATE '45000' set message_text = 'Inventory item not found';
   end if;

   if  CNT < 1 then
         signal SQLSTATE '45001' set message_text = 'Count is less than 1';
   end if;

   set CURR = select NUMBER_IN_STOCK from INVENTORY where ITEM_ID = ITEMID;
   set CURR = CURR + CNT;

   update INVENTORY set NUMBER_IN_STOCK = CURR where ITEM_ID = ITEMID;

   open CUR;
 end
```