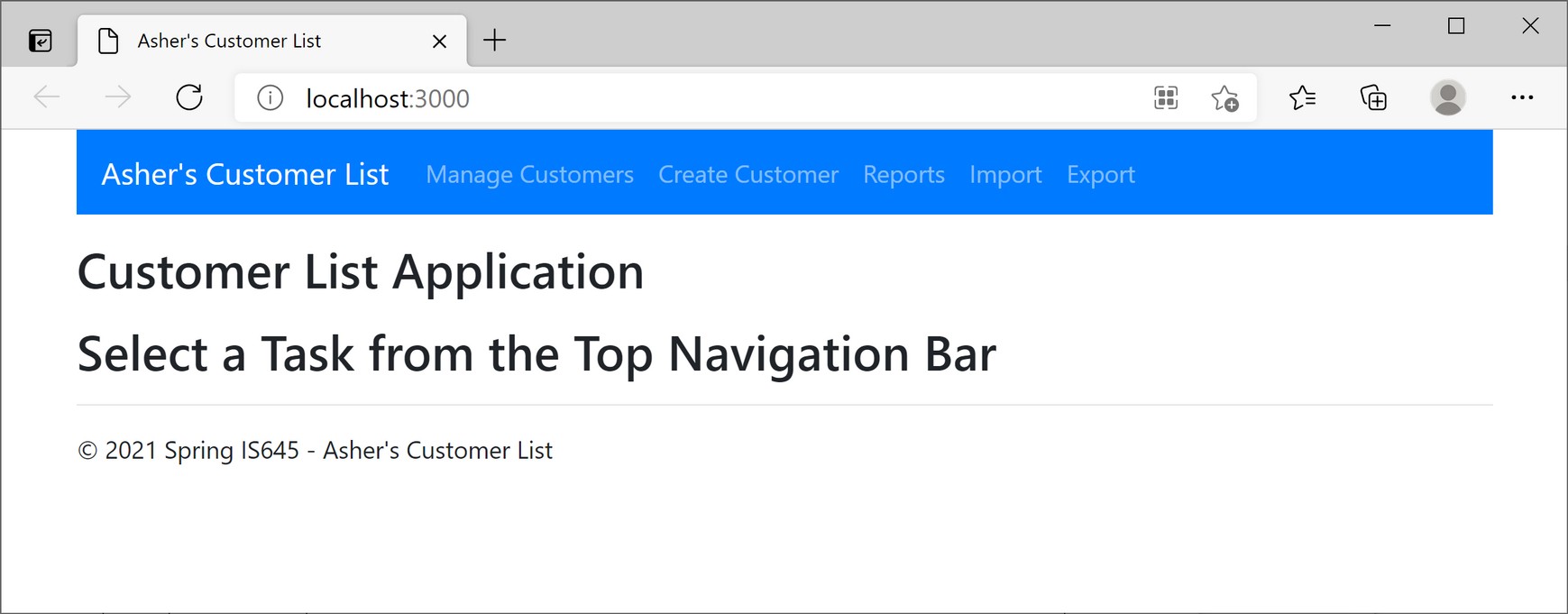
**JavaScript Project (CRUD application with database integration)**

Below are screen shots of the application. This will be a CRUD application. Using CSS, BootStrap, and CSS framework.

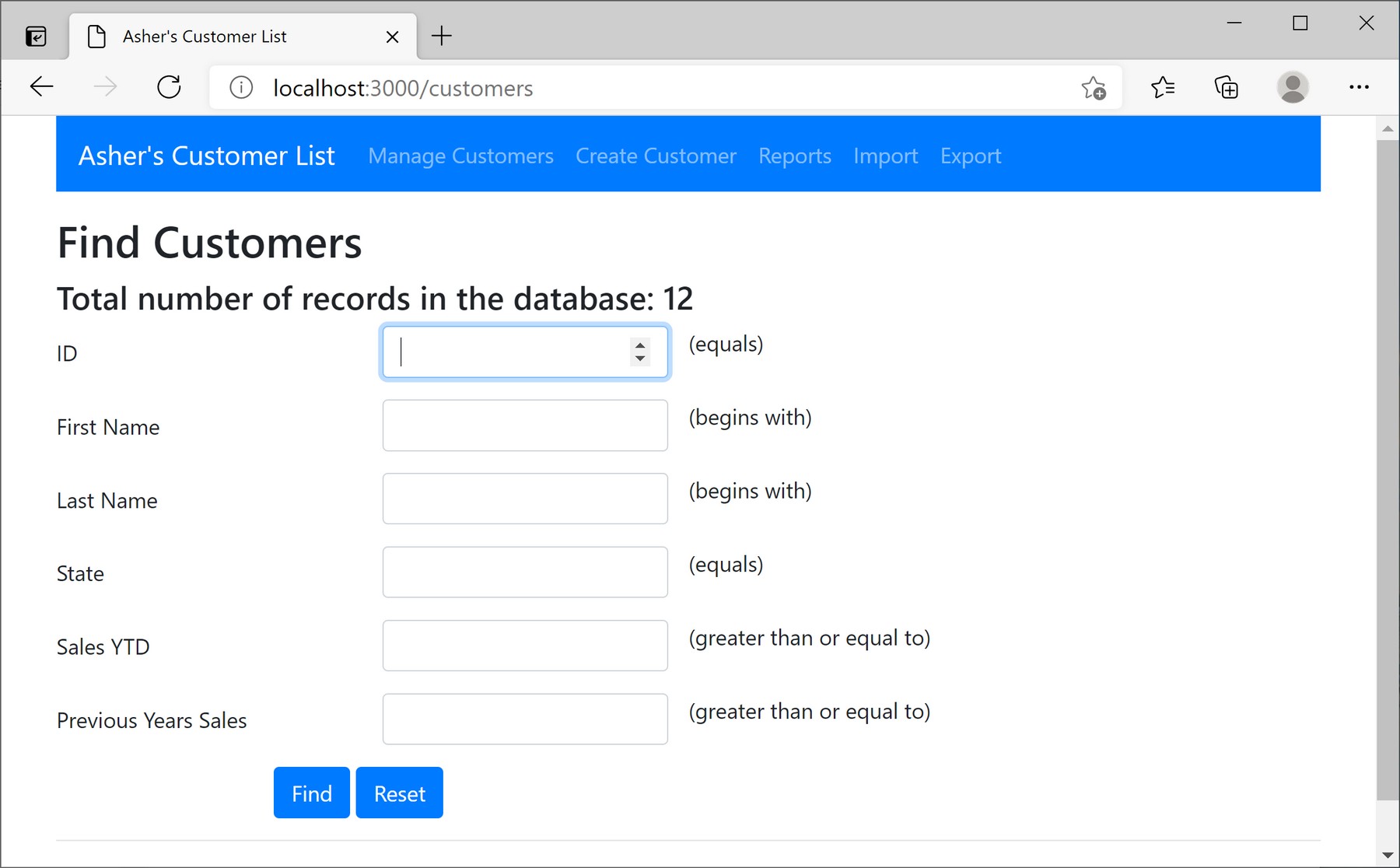
**PROGRAM VISION**

You envision that when you start the program you will have a menu similar to the one below:



Use your own last name for the Title, Footer, and Application Name.

**A. Manage Customers**



Requirement  
1. Show total number of records in the database

Find Records - Requirements

1. All fields are optional
2. If no criterion is entered, all records are returned (see below screen shot)
3. If no records are found, a message is sent back to the user (see below screen shot)
4. When results are shown, **the data entered in the search criteria is kept on the form**
5. Validation (client side is sufficient)

a. ID – Only takes numeric values (can't type text)

b. State

1. Can only type two characters max.
2. Use pattern to validate alpha characters entry (validate on submit is

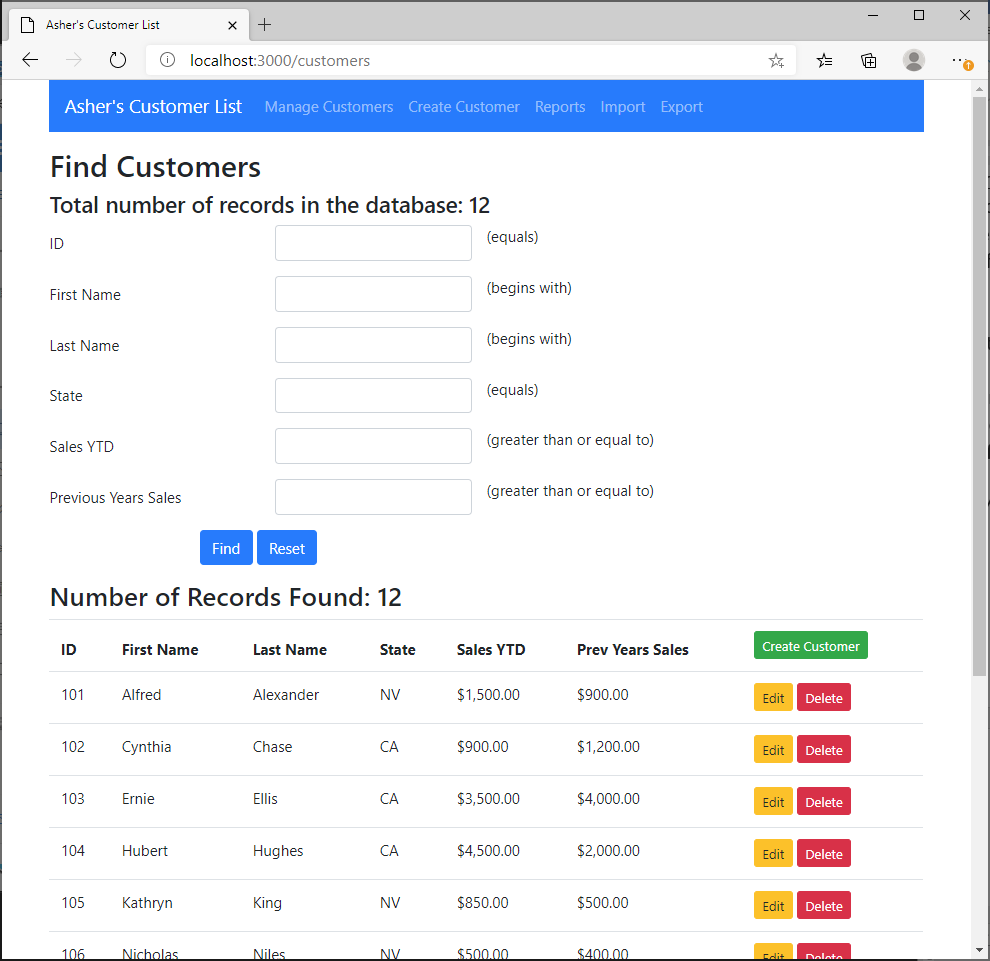
sufficient, not per each key entry)

c. Sales YTD – Only takes numeric values

d. Previous Years Sales – Only takes numeric values

6. Searches are **NOT case sensitive**

Screen Shot – All Records Found

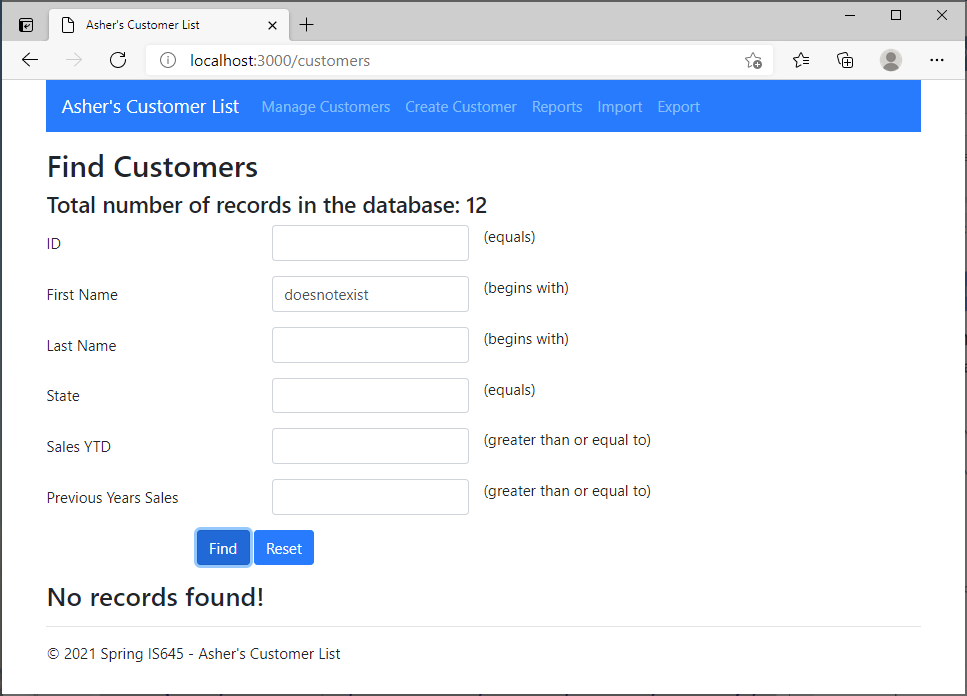


Requirements

1. Each record is displayed with hyperlinks to "Edit" or "Delete" the records
2. A "Create Customer" button is displayed in the title

Note:  
Results are shown below search form (data in the form is kept)

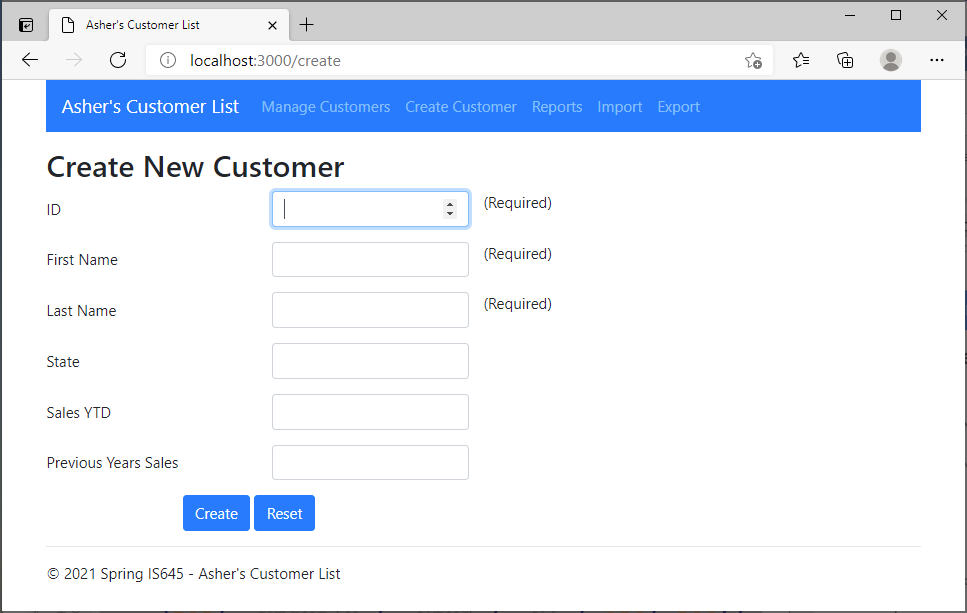
Screen Shot – No Records Found

page5image4547200

Requirements

1. Message is displayed indicating that no records were found
2. Results are shown below the search form (data in the form is kept)

**-Add New Customer Contact**When "Create Customer" is clicked from the navigation bar or from a search result, a form is displayed for the user to enter customer information.



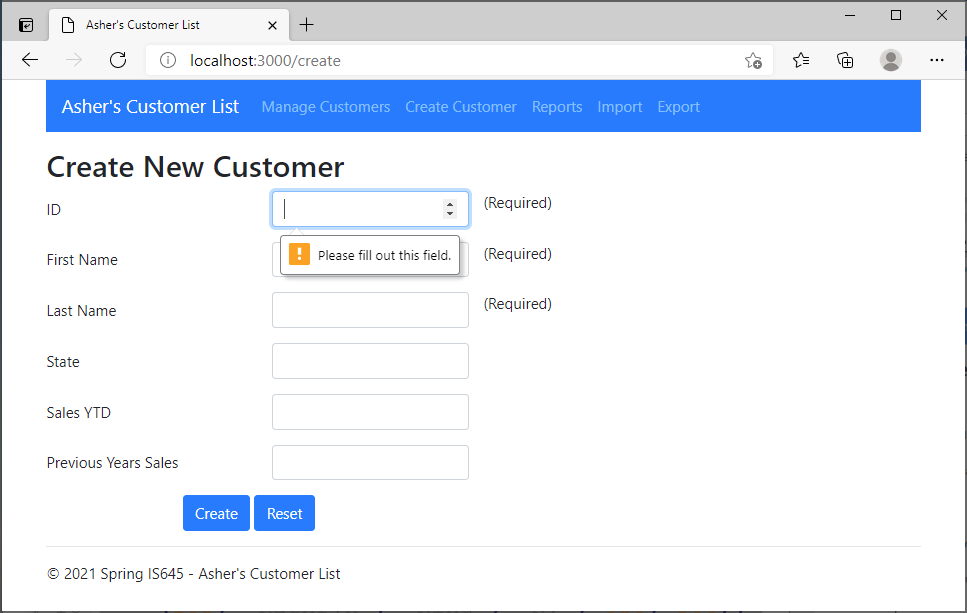
Requirements (see screen shots below)

1. Validation – same as above
2. ID, First Name, and Last Name are **required**.
   1. use client-side validation

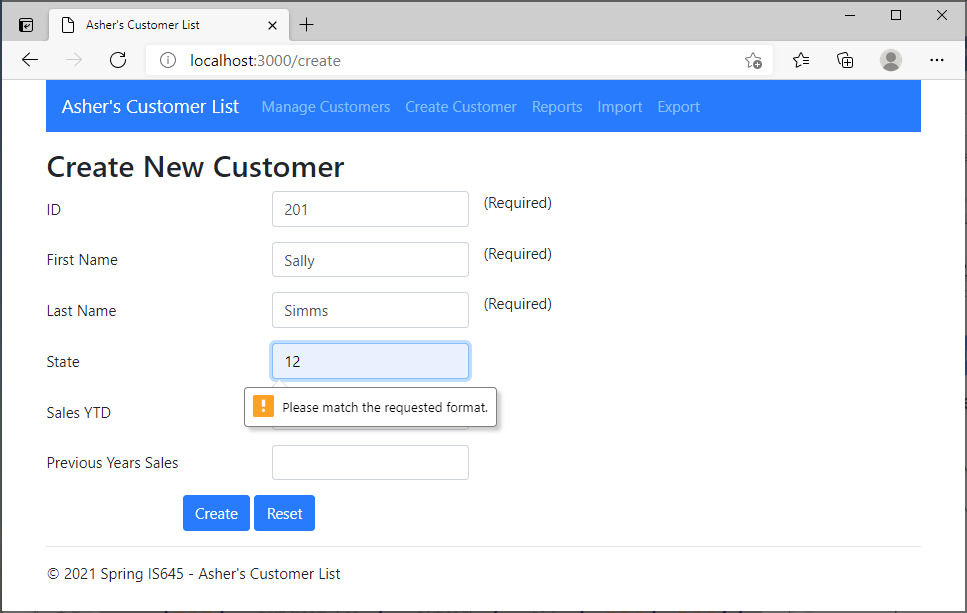
Note:

• For this case, entering the ID is fine. Can auto-populate this if you want. However, make sure that collisions do not occur.

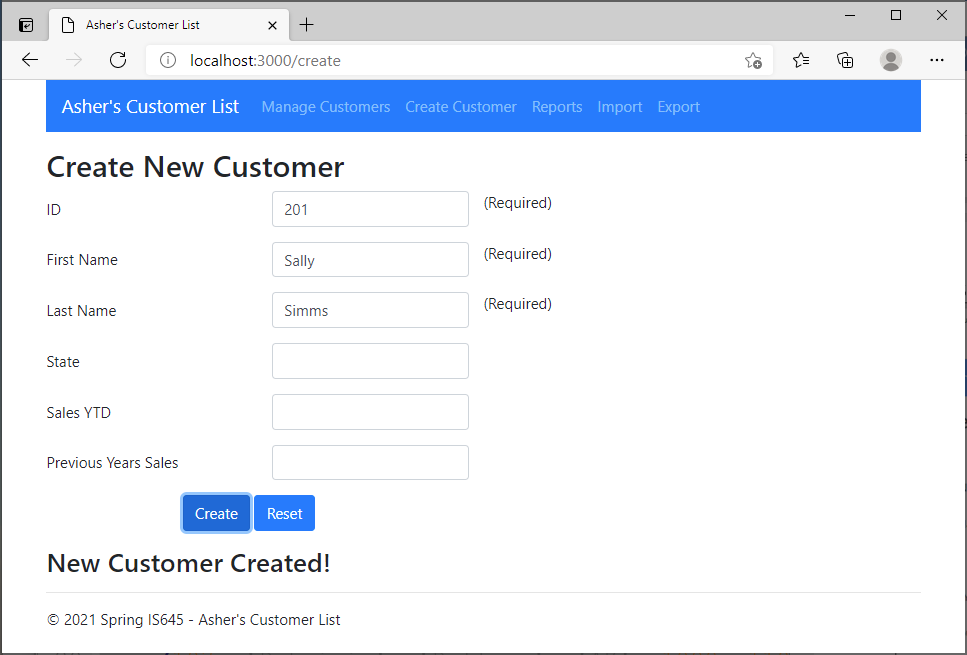
Screen Shot (client-side validation of ID)



Screen Shot (client-side validation of State Pattern)



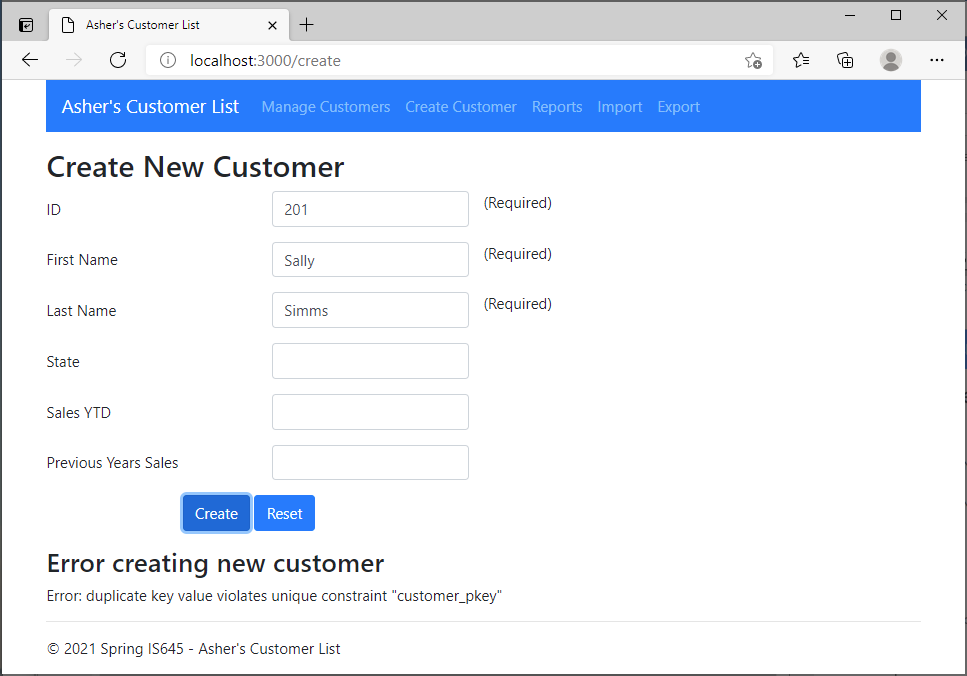
Screen Shot (Customer Successfully Created)



Requirements

1. Notification that customer has been created
2. Data entered in the form is maintained

Screen Shot (Customer Creation Error)



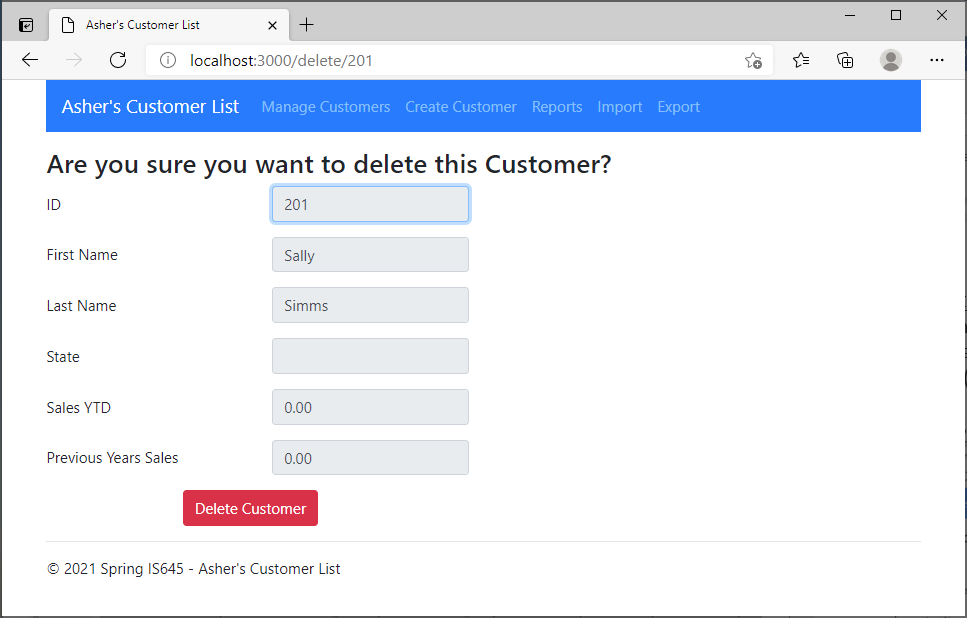
Requirements

1. Notification of error
2. Data entered in the form is maintained

**-Delete Customer**

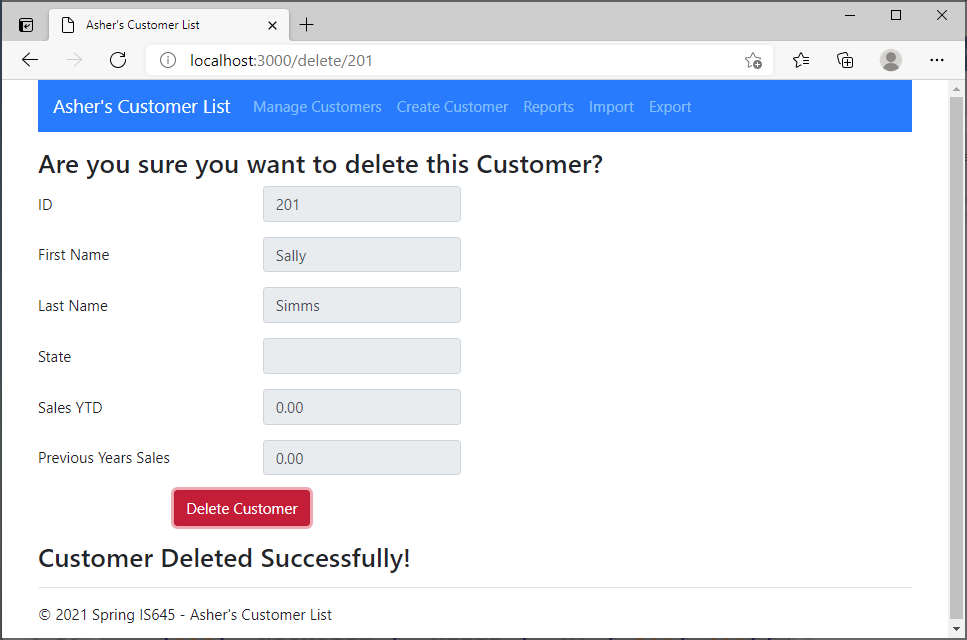
When the delete button is clicked, the user is shown the record and asked for confirmation. Upon confirmation, the record is deleted, and the user is notified of success.

Screen Shot – Delete Clicked



Requirements  
1. All fields are read only

Screen Shot – Delete Clicked on confirmation page

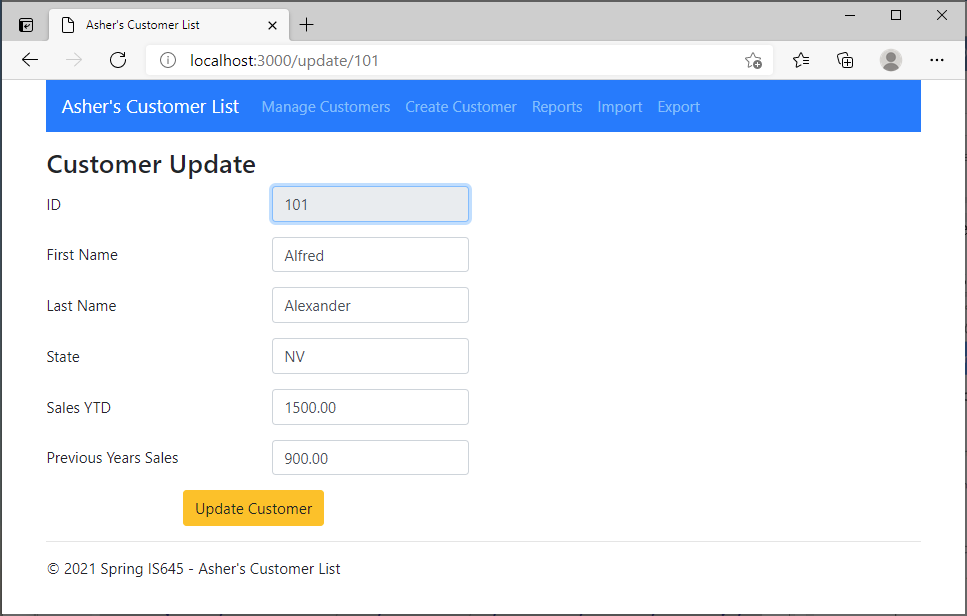


Requirements

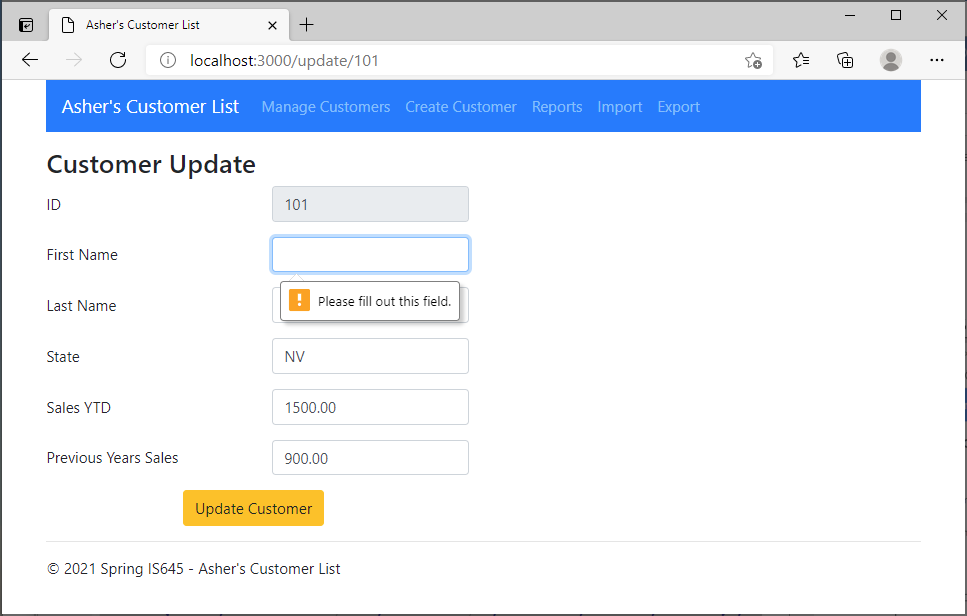
1. All fields are read only
2. Notification of deletion
3. Data entered in the form is maintained

Update Record  
When the Edit button is clicked, the user is shown the record and given the opportunity to make changes. The ID is READ ONLY. If any data is invalid, the user is shown an error message. Upon a successful update, a confirmation is displayed.

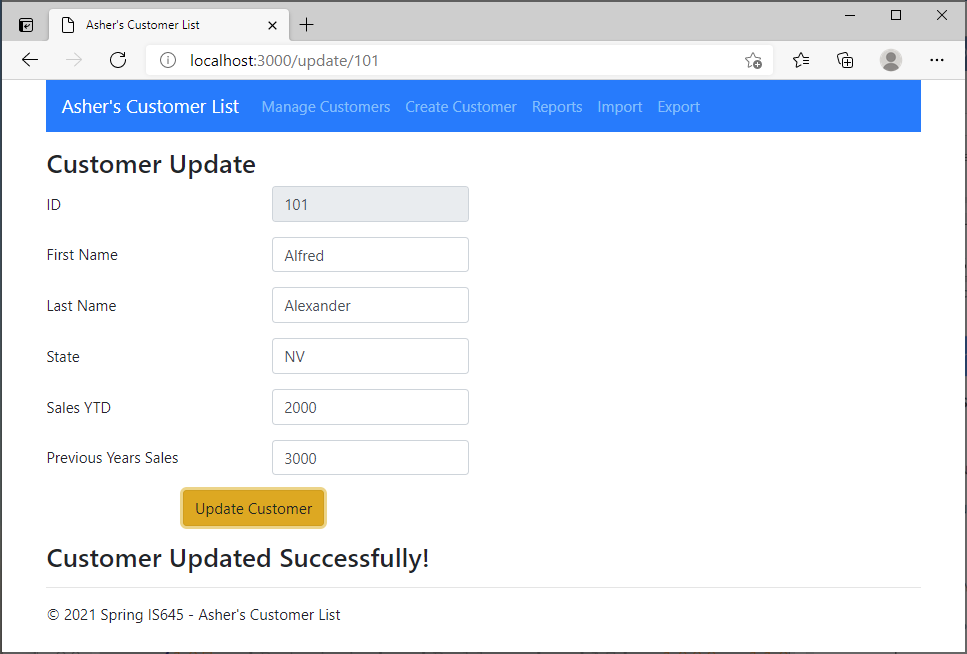
Screen Shot – Initial Edit View



Screen Shot - Invalid Data (First Name is Empty)



Screen Shot – Successful Update



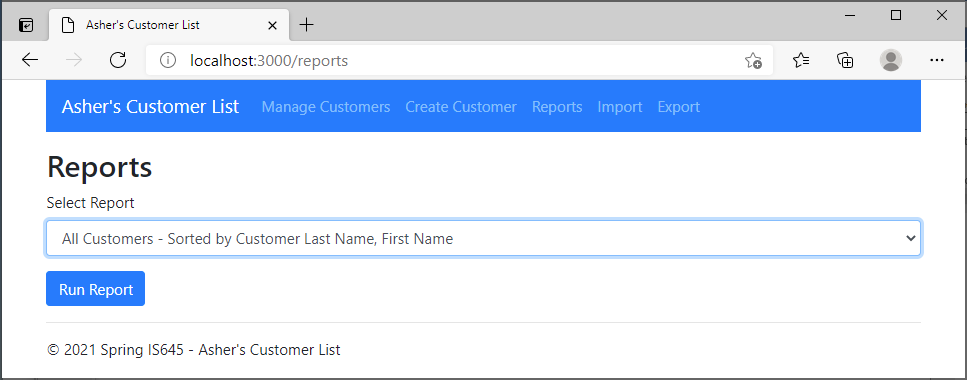
Requirements

1. Notification of update
2. Data entered in the form is maintained

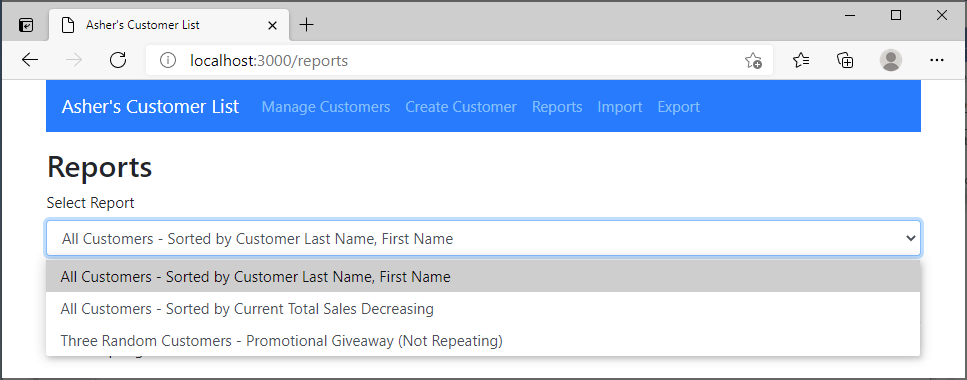
**B. Reports**

You have a need for numerous reports. You want to be able to easily choose which report to run. Therefore, you design the following.

Screen Shot – Reports Initial View



Screen Shot - Drop down list showing the reports available.

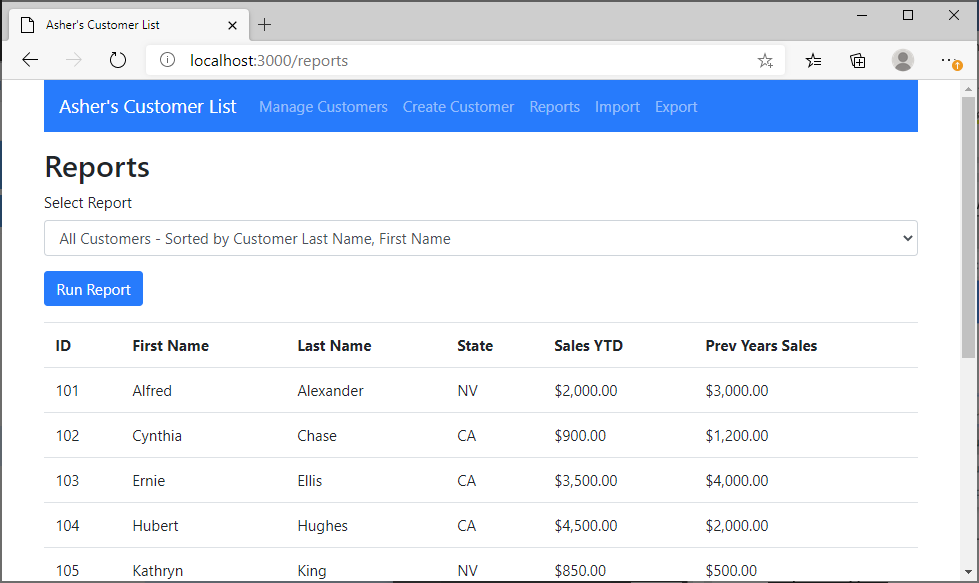


Report Types  
1. All Customers - Sorted by Customer Last Name, First Name  
2. All Customers - Sorted by Current Total Sales Decreasing  
3. Three Random Customers - Promotional Giveaway (Not Repeating) –

From the list of customers, select and display three random customers.

* + Do not display the same customer twice
  + Validate that at least three customers exist

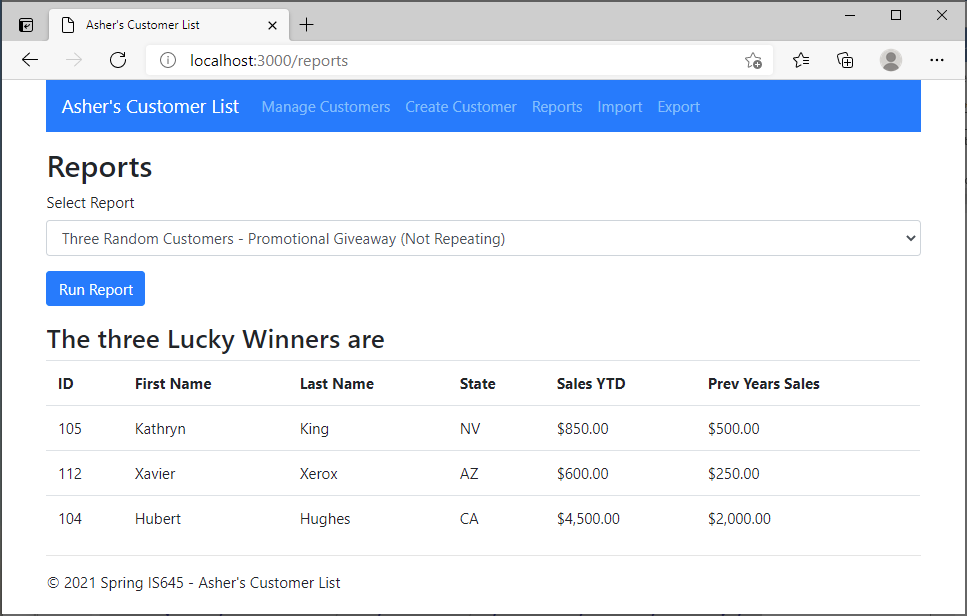
Screen Shot – All Customers sorted by Last Name, First Name



Requirements  
1. Output is displayed under the user selected report

2. Maintain user selection in the drop-down list

Screen Shot – Three Random Customers

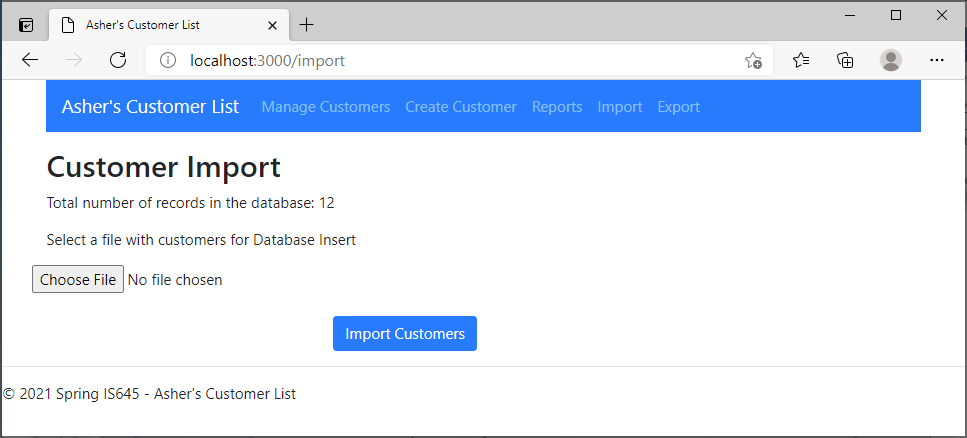


Requirements  
1. Output is displayed under the user selected report  
2. Maintain user selection in the drop-down list  
3. Do not repeat selected customers  
4. Validate those three customers exist otherwise display an error

**C. Import Files**

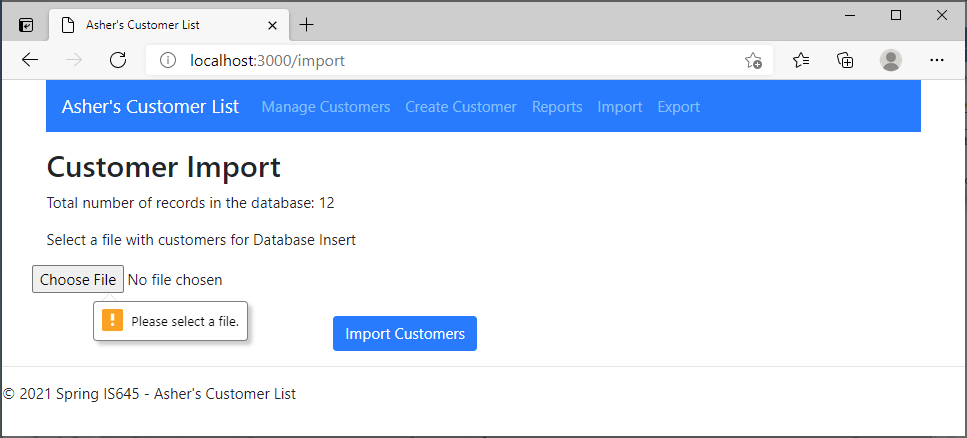
Note/Comment:  
Importing records to the database and providing user feedback is not best suited via a web-based interface per the requirements in this section. You will need to wait for each of the asynchronous inserts to complete in order to determine success/failure. As a result, the user will have to wait at the browser window. However, this is a good exercise to ensure you understand not only database inserts, but how to work with asynchronous processes.

You receive potential customer listings electronically. You want to be able to insert these customers to your program. You create the view that follows.

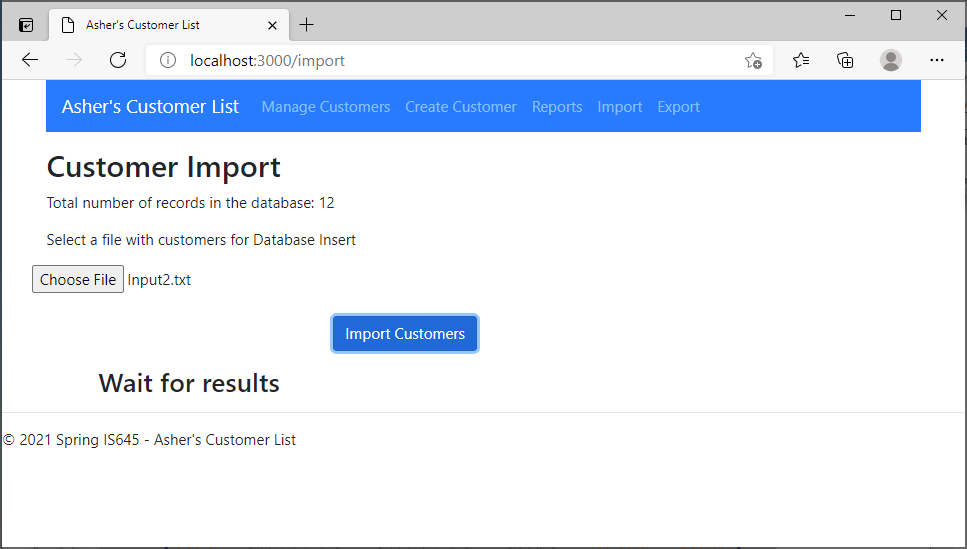


Requirements

1. Display the total number of records in the database
2. Validate that a file is chosen (see screen shot below)

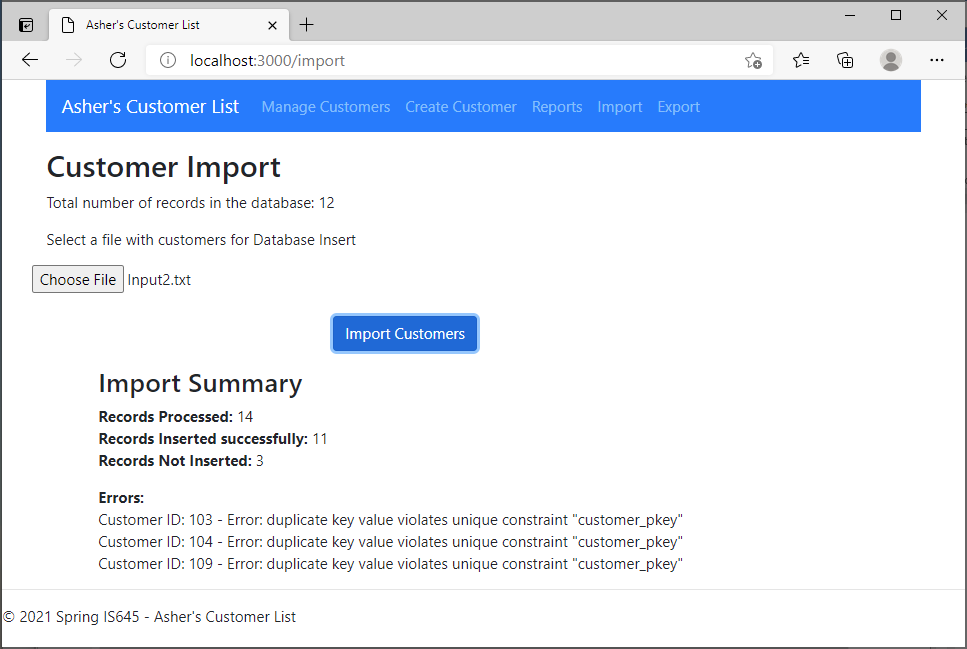
Screen Shot – No file chosen 

Screen Shot – Import Customer (while waiting, see below)



Requirements  
1. Add a notification to wait for results  
Note:  
You will use a loop to read the records and insert into the database.  
You will wait for a response on each record to be inserted. So, this might take a few seconds. To let the user, know that activity is occurring in the background, display a notification.

Screen Shot – Successful Import Summary



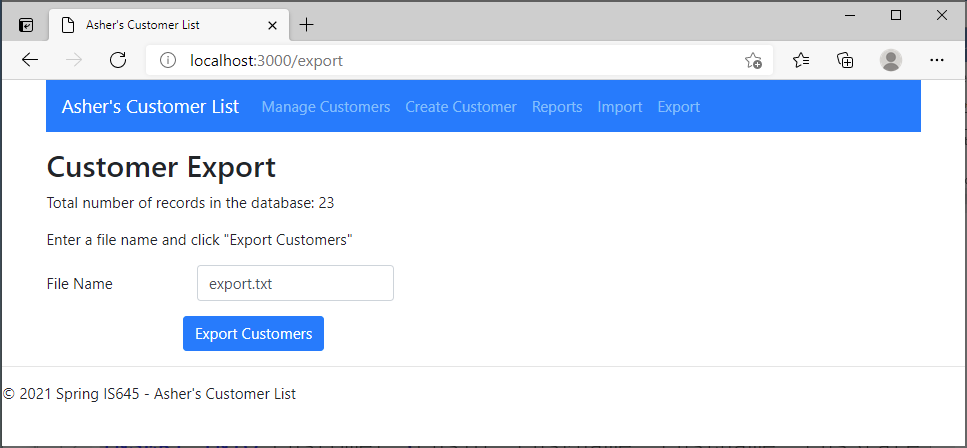
Requirements  
1. Display a summary of processed files

1. Total records processed
2. Records inserted successfully
3. Records not inserted
4. Errors – Details of why records were not inserted.

**D. Export Files**

You want the ability to back up and archive your data. You envision the following view.

Screen Shot - Initial View



Requirements

1. Display the total number of records in the database
2. Default the filename to "export.txt"
3. When "Export Customers" is clicked, a file is downloaded to the client
   1. File is to be a comma separated file, using the same format as the Input files provided. You should be able to delete all the database records and use this file to re-import

**Database**

**The script for database creation and adding sample entries is as follows:**

delete from customernew;

DROP TABLE customernew;

CREATE TABLE customernew (

cusId INTEGER PRIMARY KEY,

cusFname VARCHAR(20) NOT NULL,

cusLname VARCHAR(30) NOT NULL,

cusState CHAR(2),

cusSalesYTD MONEY,

cusSalesPrev MONEY

);

INSERT INTO customernew (cusId, cusFname, cusLname, cusState, cusSalesYTD, cusSalesPrev)

VALUES

(101, 'Alfred', 'Alexander', 'NV', 1500, 900),

(102, 'Cynthia', 'Chase', 'CA', 900, 1200),

(103, 'Ernie', 'Ellis', 'CA', 3500, 4000),

(104, 'Hubert', 'Hughes', 'CA', 4500, 2000),

(105, 'Kathryn', 'King', 'NV', 850, 500),

(106, 'Nicholas', 'Niles', 'NV', 500, 400),

(107, 'Patricia', 'Pullman', 'AZ', 1000, 1100),

(108, 'Sally', 'Smith', 'NV', 1000, 1100),

(109, 'Shelly', 'Smith', 'NV', 2500, 0),

(110, 'Terrance', 'Thomson', 'CA', 5000, 6000),

(111, 'Valarie', 'Vega', 'AZ', 0, 3000),

(112, 'Xavier', 'Xerox', 'AZ', 600, 250),

(113, 'Xavier', 'Xerox', 'AZ', 600, 250);

--select \* from customernew;