[Back to Overview](http://people.senecacollege.ca/wayne.bryan/ols655-20162/project/index.html).

**Stage 2**

Project details with this background colour apply solely to groups that have two members only. For two member groups, your application will not be able to add destinations to a cruise or remove destinations from a cruise, so this will have to be performed manually using SQL Developer.

Project details with this background colour apply solely to groups that have three members.

This stage of the project will be divided into two or three sections. Select the section for which you will be responsible (Part A, Part B, or Part C) and e-mail (one per group) each members part to your professor before class on Wednesday July 6, 2016.

For your Stage 3 presentation you will need to connect to a common database (that all teams will use), so to aid in the consistency of the development, the [Create Table](http://people.senecacollege.ca/wayne.bryan/ols655-20162/project/Project_CreateTable.sql) statements have been published. Use these to create your tables so that all groups are using the same table design.

Make sure to include any views and stored procedures (stored functions) in a **.sql** file within your .NET project. This will ensure that these files are versioned under the subversion software control and that these files are submitted when you submit your project.

Make sure that all database changes are committed!

All pages should have appropriate page titles.

You must submit your work to the digital drop box by **Tuesday August 2, 2016**. There should only be one submission per group. Please note that this is my recommended submission date. Your professor, however, has the ultimate last say and may decide to have you submit on a different day. Please respect his authority in this regard.

This stage is worth **8%** of the 15% project mark.

**Part A**

**Pages**

**Base Page**

The person doing this part of the project will be responsible for the **BasePage**. This page provides code that overrides the **System.Web.UI.Page**'s **OnInit** method. In this method, the program will check for a **Traveller** object in the session-scope and, if not found, will redirect to the login page.

**Login Page**

The person doing this part of the project will be responsible for the **Login** page. This page shows a Logon box prompting the user for their traveller id (Oracle username) and password. Be sure that the password text box hides the characters (that is, use a password text box). When the user clicks **Logon**, the program uses the traveller id and password to log on to Oracle. If the logon is invalid (an exception occurs), a message stating that should appear on the page. Once the user has successfully logged on to Oracle, the program creates a **Traveller** object containing the traveller's traveller id, password, first name, and last name, email, and administrator status and then stores it into the session-scope. Then the login page should redirect to the application's home page.

This page should not inherit from the base page.

**Reservations Page**

The person doing this part of the project will be responsible for the **Reservations** page. When this page first opens, the traveller will have already logged in and the appropriate **Traveller** object stored into the session-scope. The application should show the traveller's name (first and last) on the page, and get a list of cruise reservations for the traveller.

Use a GridView for the list of cruise reservations. Include the ship id in a hidden column in the grid view. The list of cruise reservations should also include the ship name and the cabin number as well as a comma-separated list of destinations for the cruise. Beside each cruise reservation should be a **Cancel Reservation** button (hint: add a ButtonField column to the GridView) that can be used to cancel that traveller's cruise reservation. If the traveller currently has no cruise reservations, the page should contain a message stating that. When the **Cancel Reservation** button is clicked, the application must cancel the traveller's cruise reservation and then update the GridView list of current cruise reservations to reflect the change.

The page should contain a button or link to the Reserve Cruise page. This button/link is clicked to create a new cruise reservations.

Finally, the page should contain a **Home** button or link. However, if the traveller is not an administrator, then the button or link should be used to log out instead (note that you can dynamically change the button or link text during the page load event). When clicked, if the traveller is not an administrator, the application should abandon the current session. Then, whether the traveller is an administrator or not, the applicaiton should redirect to the main home page.

This page should inherit from the base page.

**Classes**

Develop a **Traveller** class that represents a cruise traveller and includes the first and last name of the traveller, and the traveller's email address. The traveller id and password represent the traveller's Oracle login information (username and password).

**Constructors**

<constructor>

Parameters: **string** Id, **string** Password, **string** FirstName, **string** LastName, **string** EMail, **bool** IsAdmin  
Stores the id (Oracle username), (Oracle) password, first name, last name, email, and administrator status into private class fields.

This is the only constructor method that should be publicly supported in the class.

**Properties**

Id

A **string** property that returns the id (Oracle username) of the traveller.

Password

A **string** property that returns the (Oracle) password of the traveller.

FirstName

A **string** property that returns the first name of the traveller.

LastName

A **string** property that returns the last name of the traveller.

EMail

A **string** property that returns the traveller's email address.

IsAdmin

A **bool** property that returns **true** if the traveller is an administrator, otherwise **false** is returned.

These are the only properties that should be publicly supported in the class. You and your partner(s) may agree to change names of the properties indicated above.

**Methods**

ToString

Override the **ToString** method to return the full name of the traveller as **firstname** **lastname** (with a space between).

This is the only method that should be publicly supported in the class.

Develop a **Destination** class that represents a cruise ship destination and includes the destination.

**Constructors**

<constructor>

Parameters: **string** DestinationName  
Stores the name of the destination into a private class field.

This is the only constructor method that should be publicly supported in the class.

**Properties**

DestinationName

A **string** property that returns the name of the destination.

This is the only property that should be publicly supported in the class. You and your partners may agree to change name of the property indicated above.

**Methods**

ToString

Override the **ToString** method to return the name of the destination.

This is the only method that should be publicly supported in the class.

**Data Access Objects**

Develop a **TravellerDAO** class that represents the traveller data access object. The class interacts with the database to perform the traveller authentication operation as well as providing lists of traveller reservations and cruise destinations.

**Constructors**

<constructor>

Parameters: Username As String, Password As String  
Stores the Oracle username and password login information into a private class fields.

This is the only constructor method that should be publicly supported in the class.

**Methods**

FindById

Parameters: <None>  
Returns: **Traveller**  
This method returns the Traveller corresponding to the Oracle username (passed to the constructor) or **null** (**Nothing** for Visual Basic) if a traveller with the specified username is not found in the database.

GetReservations

Parameters: <None>  
Returns: cs:**List<Reservation>** or vb:**List(Of Reservation)**  
This method returns a List<Reservation> representing the reservations that the specified traveller corresponding to the Oracle username (passed to the constructor) currently has.  
  
The reservations should be ordered alphabetically by ship name.  
  
Note that each reservation must also include a comma-separated list of destinations for the reserved cruise (e.g. Jamaica, Saint Lucia, Mexico) as a **string**. Note that a List<Destination> can be easily converted into a comma-separated list using the **string.Join** method.

GetDestinations

Parameters: **int** ShipId  
Returns: cs:**List<Destination>** or vb:**List(Of Destination)**  
This method returns a List<Destination> representing the destinations for the cruise ship with the specified ship id.  
  
The destinations must be returned in alphabetical order.

GetShips

Parameters: <None>  
Returns: cs:**List<CruiseShip>** or vb:**List(Of CruiseShip)**  
This method returns a List<CruiseShip> representing only the cruise ships in the database that currently have one or more destinations. If there are no cruise ships in the database with one or more destinations, then an empty list should be returned.  
  
The cruise ships should be returned in alphabetical order by the ship name.

These are the only instance methods that should be publicly supported in the class. You and your partner(s) may agree to change the names of the methods indicated above.

**Stored Procedure**

ADD\_DESTINATION

Parameters: pship\_id NUMBER, pdestination VARCHAR2, psuccess OUT NUMBER  
Returns: None  
This stored procedure adds the specified destination to the specified cruise. If the cruise already contains the specified destination, then the procedure does nothing and returns 0 through the **psuccess** OUT parameter to indicate the operation was not successful. Otherwise the destination is added to the cruise and the procedure returns 1 through the **psuccess** OUT parameter to indicate that the operation was successful.

You and your partner(s) may agree to change the names of the stored procedure and/or the parameters indicated above. Note: to specify the OUT parameter in ASP.NET, use:  
  
cmd.Parameters.Add("psuccess", OracleType.Number).Direction = ParameterDirection.Output

**Part B**

**Pages**

**Home Page**

The person doing this part of the project will be responsible for the main (home) page. When this page first opens, the traveller will have already logged in and the appropriate **Traveller** object stored into the session-scope. If the traveller is not an administrator, then the page must immediately redirect to the Reservations page. Otherwise, the Home page should show the administrator's name (first and last).

The page should contain a button or link to the Reservations page. This button/link is clicked to create a new cruise reservations or to cancel existing cruise reservations.

The page should contain a button or link to the Cruise Destination Management page. This button/link is clicked to add new destinations to a cruise or remove existing destinations from a cruise.

Finally, the page should contain a **Logout** button or link. When clicked the application should abandon the current session and redirect back to the main home page.

This page should inherit from the base page.

**Reserve Cruise Page**

The Reserve Cruise page should contain a drop down list to select the desired cruise ship and a text box for the cabin number. The cabin number is required. Use a **RequiredFieldValidator** to ensure that the cabin number is entered. The page should also include a **Reserve Cruise** button. A label should be used to show a comma-separated list of destinations for the currently selected cruise.

When the cruise ship is selected in the drop down list, the program must immediately update the label to show the destinations of the newly selected cruise ship in a comma-separated list (e.g. This ship travels to: Jamaica, Saint Lucia, Mexico).

When this page first opens, the application should populate the drop down list with the *available* cruise ships — that is, the cruise ships that already have one or more destinations. If there are no cruise ships with destinations available, then the page must display a suitable message and disable the **Reserve Cruise** button. If there are available cruise ships, then the first one will already be selected in the drop down list (by default). So a comma-separated list of destinations for that cruise ship should also be displayed.

When the **Reserve Cruise** button is clicked, the application must create the traveller's reservation for the specified cruise ship and cabin number. If the specified cabin is not available, the program must display an appropriate message. An error message should be displayed if any other error occurs. If the reservation was successfully made, then the application should redirect to the Reservations page.

Finally, the page should contain a **Home** button or link. If a button is used, make sure the button does not cause a validation. When clicked the application should redirect to the home page.

This page should inherit from the base page.

**Classes**

Develop a **Reservation** class that represents a cruise reservation and includes the ship id, ship name, cabin number, and a comma-separated list of destinations.

**Constructors**

<constructor>

Parameters: **int** ShipId, **string** ShipName, **int** CabinNo, **string** Destinations  
Stores the ship id, ship name, cabin number, and comma-separated list of destinations into private class fields.

This is the only constructor method that should be publicly supported in the class.

**Properties**

ShipId

An **int** property that returns the ship id for the cruise reservation.

ShipName

A **string** property that returns the name of the ship for the cruise reservation.

CabinNo

An **int** property that returns the cabin number for the cruise reservation.

Destinations

A **string** property that returns the comma-separated list of destinations for the cruise being reserved.

These are the only properties that should be publicly supported in the class. You and your partner(s) may agree to change names of the properties indicated above.

Develop a **CruiseShip** class that represents a cruise ship and includes the ship id and the ship name.

**Constructors**

<constructor>

Parameters: **int** ShipId, **string** ShipName  
Stores the ship id and ship name into private class fields.

This is the only constructor method that should be publicly supported in the class.

**Properties**

ShipId

An **int** property that returns the id of the cruise ship.

ShipName

A **string** property that returns the name of the cruise ship.

These are the only properties that should be publicly supported in the class. You and your partners may agree to change names of the properties indicated above.

**Methods**

ToString

Override the **ToString** method to return the ship name.

This is the only method that should be publicly supported in the class.

**Data Access Objects**

Develop a **ReservationDAO** class that represents the reservation data access object. The class interacts with the database to perform the reservation and reservation cancellation operations required as well as providing lists of available destinations for a specific cruise ship.

**Constructors**

<constructor>

Parameters: Username As String, Password As String  
Stores the Oracle username and password login information into a private class fields.

This is the only constructor method that should be publicly supported in the class.

**Methods**

GetAvailableDestinations

Parameters: **int** ShipId  
Returns: cs:**List<Destination>** or vb:**List(Of Destination)**  
This method returns a List<Destination> representing the destinations that are available to be added to cruise ship indicated. A destination is available if the cruise ship is not already travelling to that destination.  
  
The destinations should be returned in alphabetical order.

CreateReservation

Parameters: **int** ShipId, **int** CabinNo  
Returns: **int**  
This method calls the **CREATE\_RESERVATION** stored procedure to create the cruise reservation and then returns **false** if the success status is 0, otherwise it returns **true**.  
  
Note that the value 0 or 1 can be directly converted to **false** or **true** using the **Convert.ToBoolean** method.

CancelReservation

Parameters: **int** ShipId, **int** CabinNo  
Returns: <None>  
This method deletes the cruise reservation with the sepcified ship id and cabin number from the database.

These are the only instance methods that should be publicly supported in the class. You and your partner(s) may agree to change the names of the methods indicated above.

**Stored Procedure**

CREATE\_RESERVATION

Parameters: ptraveller\_id VARCHAR2, pship\_id NUMBER, pcabin\_no NUMBER, psuccess OUT NUMBER  
Returns: None  
This stored procedure checks if there already exists a reservation for the specified cruise ship and cabin number and if so, returns 0 through the **psuccess** OUT parameter to indicate that the reservation cannot be made (cabin is already reserved). Otherwise the procedure adds a reservation for the specified traveller to the database for the specified cruise ship and cabin number and returns 1 through the **psuccess** OUT parameter to indicate reservation was successfully made.

You and your partner(s) may agree to change the names of the stored procedure and/or the parameters indicated above. Note: to specify the OUT parameter in ASP.NET, use:  
  
cmd.Parameters.Add("psuccess", OracleType.Number).Direction = ParameterDirection.Output

**Part C**

**Pages**

**Cruise Destination Management Page**

The Cruise Destination Management page should contain a drop down list to select the desired cruise ship, a list of current cruise destinations and a list of available cruise destinations.

Use a GridView for the the list of current cruise destinations and another GridView for the list of available cruise destinations showing the destination. Beside the destination in the current destinations GridView should be a Remove Destination button (hint: add a ButtonField column to the GridView) that can be used to remove that destination from the cruise currently selected in the drop down list. If the cruise ship currently has no destinations, the page should contain a message stating that. Beside the destination in the available destinations GridView should be an Add Destination button that can be used to add that destination to the cruise currently selected in the drop down list. If there are no available destinations, the page should contain a message stating that.

When the Remove Destination button is clicked, the application must remove the destination from the cruise ship currently selected in the drop down list and then update the both GridView lists to reflect the change.

When the Add Destination button is clicked, the application must add the destination to the cruise ship currently selected in the drop down list and then update the both GridView lists to reflect the change. If the destination cannot be added to the cruise ship, a suitable error message should be displayed.

When a cruise ship is selected in the drop down list, the two GridViews must be immediately updated to reflect the current and available destinations for the newly selected cruise ship.

When this page first opens, the traveller (administrator) will have already logged in and the appropriate **Traveller** object stored into the session-scope. The page should show the traveller's name (first and last). The page should also populate the drop down list with *all* of the cruise ships. You may assume that there will always be at least one cruise ship. Once the drop down list has been populated, the first cruise ship will already be selected (by default). So, the current and available destinations for that cruise ship should be displayed in the GridViews.

Finally, the page should contain a **Home** button or link. When clicked the application should redirect to the home page.

This page should inherit from the base page.

**Classes**

Develop a **CruiseShip** class that represents a cruise ship and includes the ship id and the ship name.

**Constructors**

<constructor>

Parameters: **int** ShipId, **string** ShipName  
Stores the ship id and ship name into private class fields.

This is the only constructor method that should be publicly supported in the class.

**Properties**

ShipId

An **int** property that returns the id of the cruise ship.

ShipName

A **string** property that returns the name of the cruise ship.

These are the only properties that should be publicly supported in the class. You and your partners may agree to change names of the properties indicated above.

**Methods**

ToString

Override the **ToString** method to return the ship name.

This is the only method that should be publicly supported in the class.

Develop a **Destination** class that represents a cruise ship destination and includes the destination.

**Constructors**

<constructor>

Parameters: **string** DestinationName  
Stores the name of the destination into a private class field.

This is the only constructor method that should be publicly supported in the class.

**Properties**

DestinationName

A **string** property that returns the name of the destination.

This is the only property that should be publicly supported in the class. You and your partners may agree to change names of the property indicated above.

**Methods**

ToString

Override the **ToString** method to return the name of the destination.

This is the only method that should be publicly supported in the class.

**Data Access Objects**

Develop a **CruiseShipDAO** class that represents the cruise ship data access object. The class interacts with the database to perform the cruise destination addition and removal operations required as well as providing lists of cruise ships.

**Constructors**

<constructor>

Parameters: Username As String, Password As String  
Stores the Oracle username and password login information into a private class fields.

This is the only constructor method that should be publicly supported in the class.

**Methods**

GetShips

Parameters: **bool** OnlyWithDestiantions  
Returns: cs:**List<CruiseShip>** or vb:**List(Of CruiseShip)**  
If the parameter **OnlyWithDestinations** is **false**, then this method returns a List<CruiseShip> representing *all* of the cruise ships in the database. If the parameter **OnlyWIthDestinations** is **true**, then this method returns a List<CruiseShip> representing only the cruise ships in the database that currently have one or more destinations. If there are no cruise ships in the database with one or more destinations, then an empty list should be returned.  
  
The cruise ships should be returned in alphabetical order by the ship name.

AddDestination

Parameters: **int** ShipId, **string** Destination  
Returns: **bool**  
This method calls the **ADD\_DESTINATION** stored procedure to add the destination to the specified cruise ship and then returns **false** if the success status is 0, otherwise it returns **true**.  
  
Note that the value 0 or 1 can be directly converted to **false** or **true** using the **Convert.ToBoolean** method.

RemoveDestination

Parameters: **int** ShipId, **string** Destination  
Returns: <None>  
This method calls the **REMOVE\_DESTINATION** stored procedure to remove the destination from the specified cruise ship.

These are the only instance methods that should be publicly supported in the class. You and your partners may agree to change the names of the methods indicated above.

**Stored Procedure**

REMOVE\_DESTINATION

Parameters: pship\_id NUMBER, pdestination VARCHAR2  
Returns: None  
The procedure deletes the cruise destination with the specified ship id from the database. There cannot be any reservations for a cruise that has no destinations, so if there are no more destinations for this cruise ship, then all reservations for the cruise ship must be cancelled — that is, all reservations with the specified ship id should be deleted from the database.

You and your partners may agree to change the names of the stored procedure and/or the parameters indicated above.