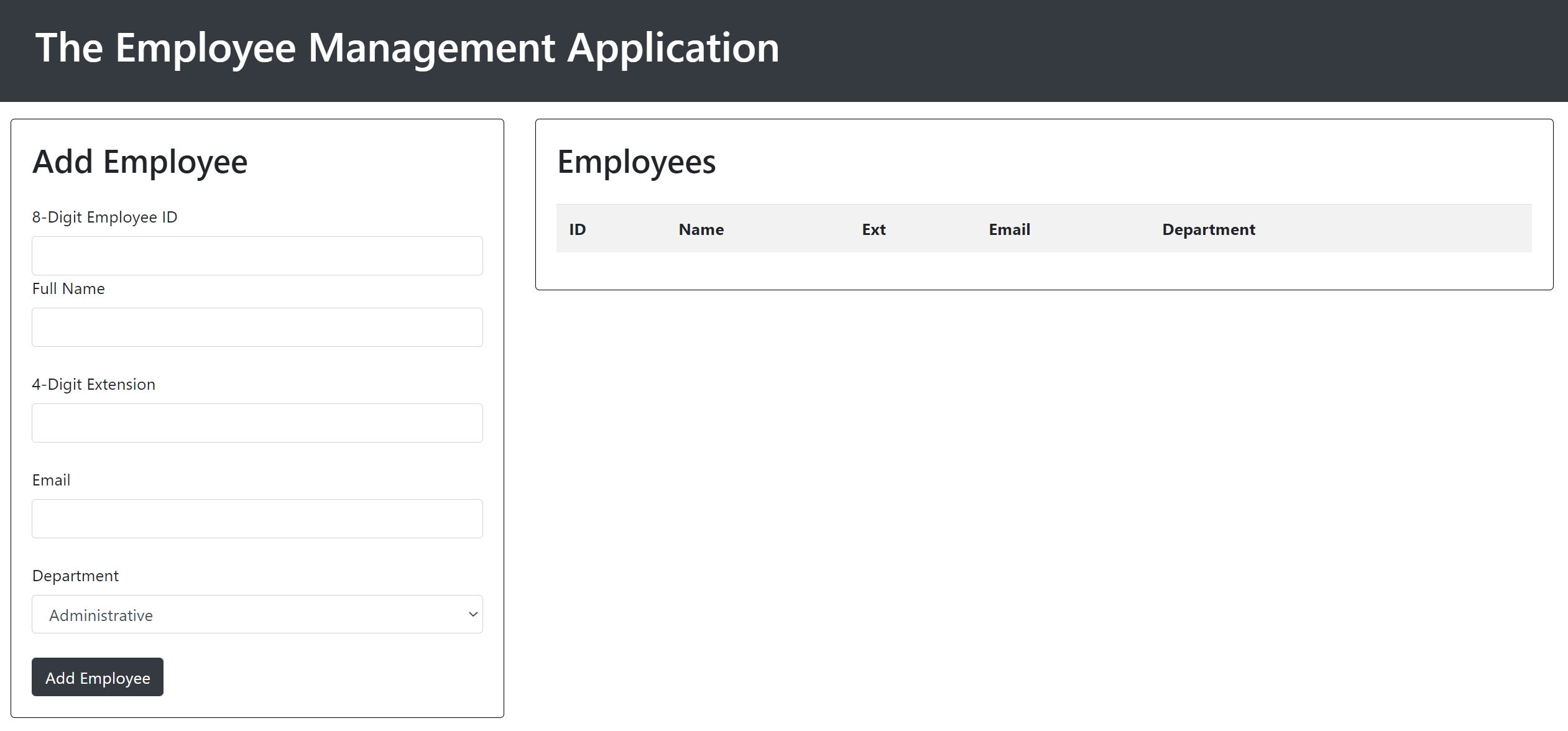
**Assignment 7**

**The Employee Management System (Add, View, and Delete Employees)**

In this assignment you will combine elements of Assignment 6 with what you learned in Lab 10. The idea is to have the ability for a user to add an employee (like Assignment 6), view an employee via a tabular interface (Lab 10), and then delete an employee from the table (Lab 10).

**The Interface**

The UI has been created for you using Bootstrap. You will not need to touch the HTML for this assignment. The example below outlines what the UI will look like.

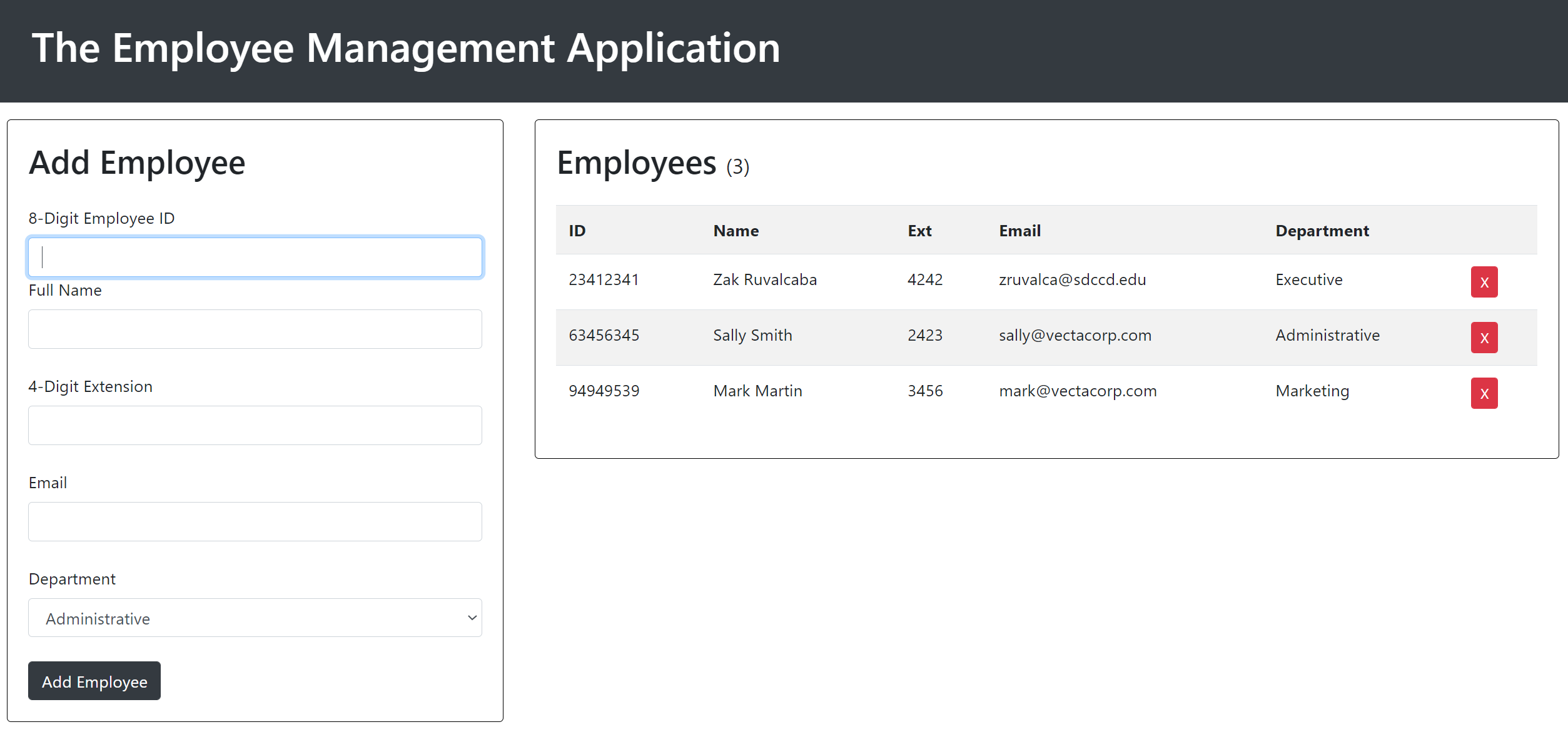


**Adding an Employee**

Similar to Assignment 6, you will want to capture the values of the Add Employee form. These values will then be displayed within the table to the right, in the “Employees” pane. The example below shows what this will look like.

General Considerations

* Upon submission, the form should completely clear itself of the entered values.
* Upon submission, the user’s cursor should immediately return to the Employee ID field.

****

**Viewing Employees**

As you can see from the example above, adding an employee will result in the values entered being shown in a tabular view. Each form field’s value has its own cell in the table that matches up with its respective table header.

Steps for dynamically creating the tabular data

* In Lab 10 you got a list form the DOM. In this assignment you will get the table from the DOM.
* Rather than creating a new LI element as you did in Lab 10, in this assignment you’ll need to create a new row for the selected table. This can be accomplished easily by calling the **.insertRow()** method for the selected table like this:  
    
  let row = selectedtable.insertRow();
* Next, you’ll have to create new cells within the new row. One cell for each piece of data captured from the Add Employee form. This can be done using the **.insertCell()** method for the row as follows (shown for ID cell only):  
    
  let cellID = row.insertCell();
* Then you’ll have to add the text value as a text node within the newly created cell. This can be down using the .createTextNode() method and passing it in to the .appendChild() method for each cell in the row.
* Don’t forget to designate a cell for the delete button. This will also have to be coded and added to a cell in the row. You don’t need the .delete class here as you had in Lab 10.

**Deleting Employees**

Similar to Lab 10, the entry should be removed when the user clicks the delete button.

General Considerations

* You don’t need to check to see if the .delete class exists here like you did in Lab 10.
* Confirm the deletion to the user. If the user clicks OK, the row should be deleted.
* Use the **.deleteRow()** method of the table to delete the row. This method will accept the rowIndex for the row you want to select. You’ll need to select the <tr> tag that the delete button exists within. Think about how you will accomplish this. The code to delete the selected row from the table will look something like this (replace path-to-tr-tag with DOM properties for selecting the <tr> tag):  
    
  table.deleteRow(e.target.path-to-tr-tag.rowIndex);

**Extra Credit (10 Points)**

**Maintaining an Employee Count**

When an employee is added to or deleted from the table, the count of employees in the table should be maintained in the <output> tag next to the heading. You’ll need to create a count variable in code, increment it when a new employee is added, decrement it when an employee is deleted, and display the result in the output tag when an employee is added or deleted.