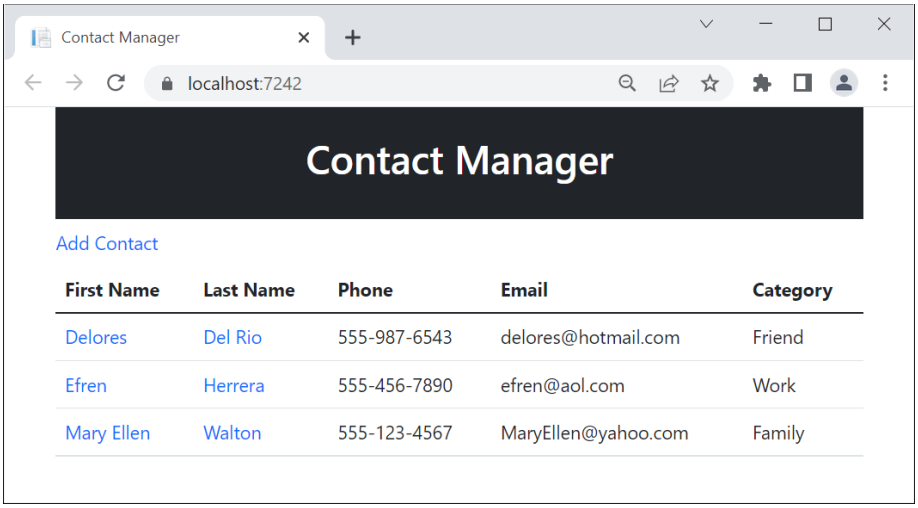
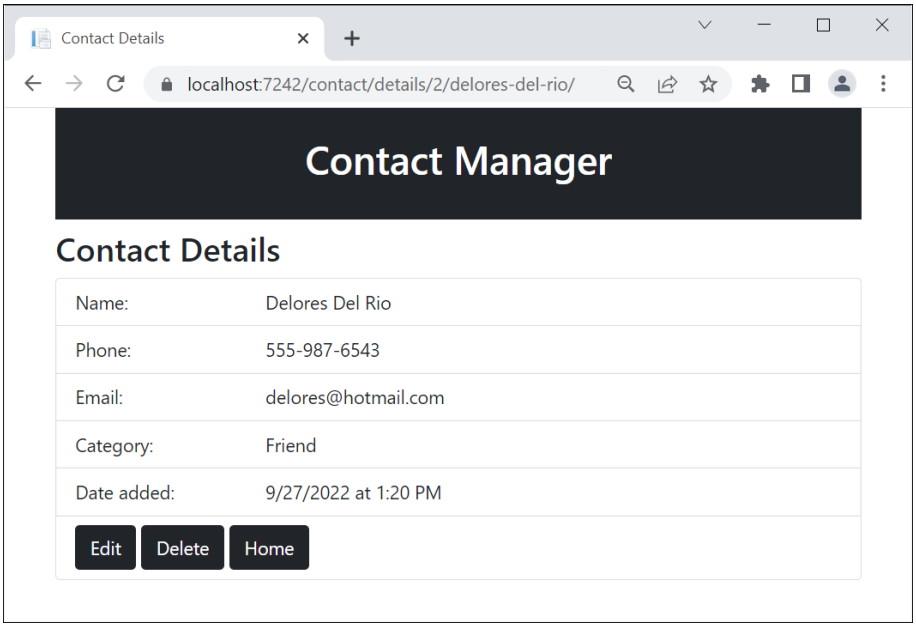
Project 4-1 Build the Contact Manager app

For this project, you will build a multi-page, data driven app like the one that's shown below.

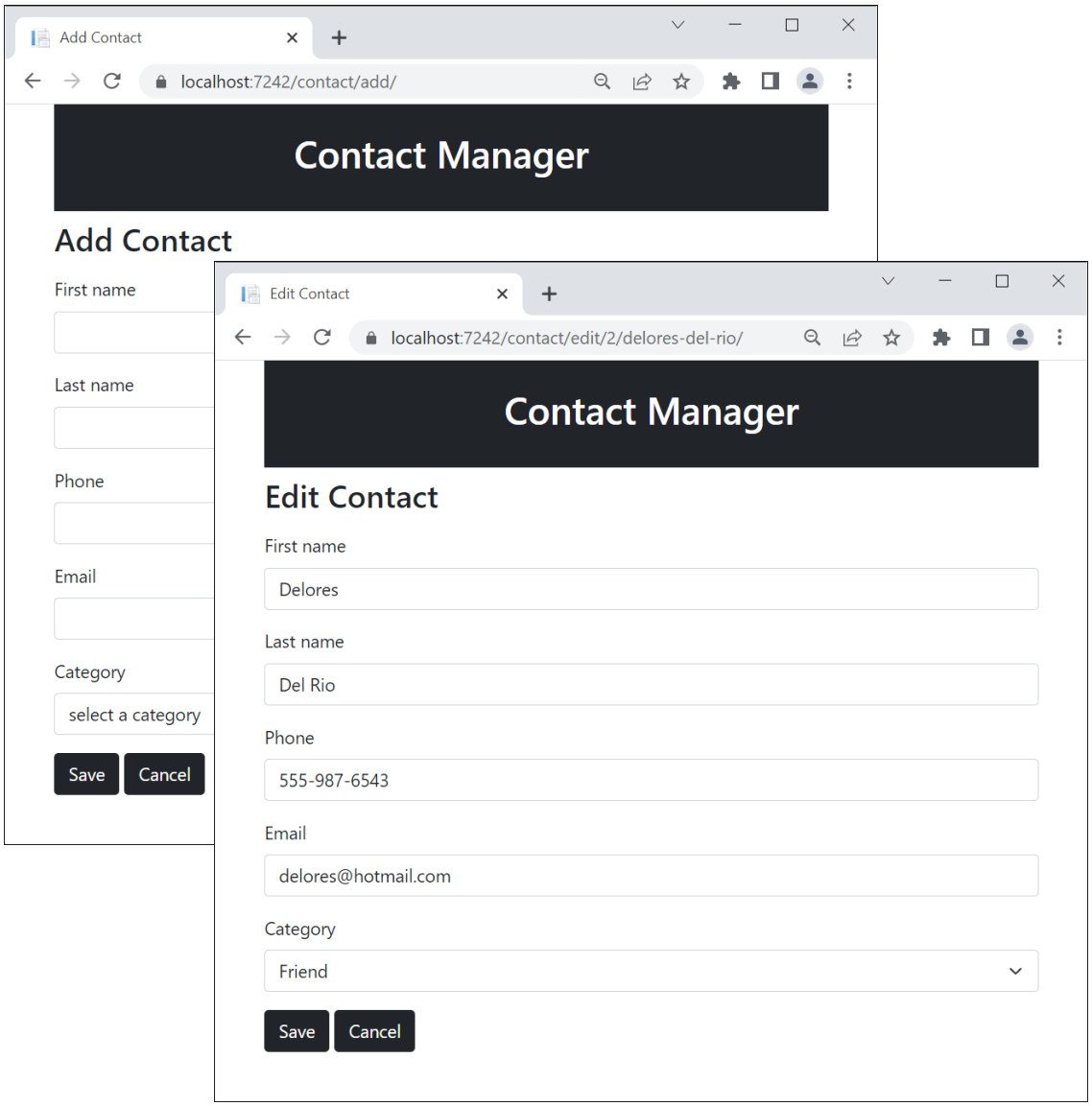
The Home page



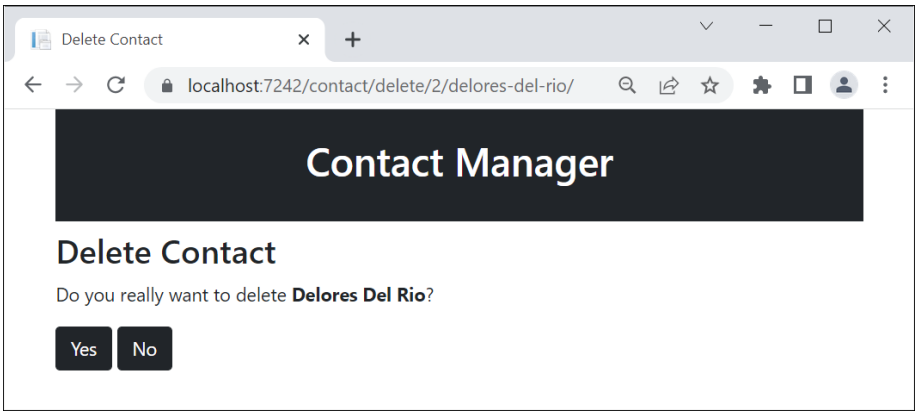
The Details page



The Add and Edit pages (both use the same view)



The Delete page



Specifications

- When the app starts, it should display a list of contacts and a link to add a contact.
- If the user clicks the first or last name of a contact, the app should display the Details page for that contact.
- The Details page should include buttons that allow the user to edit or delete the contact. Before deleting a contact, the app should display the Delete page to confirm the deletion.
- To reduce code duplication, the Add and Edit pages should both use the same view. This view should include a drop-down list for Category values.
- The Add and Edit pages should *not* include the Date Added field that's displayed by the Details page. That field should be set by code only when the user first adds a contact.
- If the user enters invalid data on the Add or Edit page, the app should display a summary of validation errors above the form.
- Here are the requirements for valid data:
    - The Firstname, Lastname, Phone, Email, and CategoryId fields are required.
    - The CategoryId field is an int (see domain model specifications below). You can't use the Required validation attribute with an int, but you can use the Range attribute to make sure the value of CategoryId is greater than zero.
- If the user clicks the Cancel button on the Add page, the app should display the Home page.
- If the user clicks the Cancel button on the Edit page, the app should display the Details page for that contact.
- The domain model classes for contacts and categories should use primary keys that are generated by the database.
- The Contact class should have a foreign key field that relates it to the Category class. It should also have a read-only property that creates a slug of the contact's first and last name that can be added to URLs to make them user friendly.
- If necessary, use the ValidateNever attribute in the Contact class to suppress unwanted validation of the Category navigation property.
- Use EF Code First to create a database based on your domain model classes. Include seed data for the categories and one or more contacts.
- Use a Razor layout to store the <html>, <head>, and <body> elements.
- Use Bootstrap to style the views. If necessary, use a custom CSS style sheet to override Bootstrap classes.
- Use the default route with an additional route segment that allows an optional slug at the end of a URL.
- Make the app URLs lowercase with trailing slashes.