**CS649  
Assignment 5**

In this assignment, you will again build on your previous assignment. In this assignment you will focus on routing and adding update and delete APIs to your application.

**Routing**

Demonstrate your knowledge of route parameters by adding a new page for the image display. When the user clicks the View link in your grid, they should redirect to the new page and the product appears by programmatically setting an <img> tag to the URL that you're storing in the database.

**Edit / Update**

Create an edit / update page. This will require you to perform a few tasks:

* An edit link will need to be added to your main table.
* You will demonstrate your knowledge of routing here as well. When you click the Edit link, the user should be routed to the edit page which you will build next.
* Create a new .jsx file for this new edit functionality. Within this file, add the necessary form fields to handle editing of the content for the row that you selected.   
  *Note: In chapter 10, you learned about paging from one record to the next. You won't be adding that functionality here. Just focus on clicking the Edit link, routing to this page, and displaying the data within the appropriate form fields.*
* Create at least two specialized input components. Ideally one for number inputs (NumInput.jsx) and one for text inputs (TextInput.jsx). This will need to be correctly integrated into your edit component.

**Create an Update API**

* Modify the schema.
* Add a new mutation entry point.
* Connect the API to its resolver in api\_handler.js (assuming you named your file as such).
* Create an update() function and implement the actual resolver in your main .js file.
* Don't forget to export the update function.
* Add the changes for saving updates to the database to your edit component.

**Create a Delete API**

* Modify the schema by adding a new mutation entry point. This should just take in the ID of the field to be deleted and return a Boolean value to indicate successful deletion.
* Connect the API to its resolver in api\_handler.js (assuming you named your file as such).
* Create a remove() function (not delete() because delete is a reserved keyword) and implement the actual resolver in your main .js file.
* Don't forget to export the delete function as remove.
* Add a Delete button within every row in your grid. When the user clicks the Delete button it should delete that product.