**Assessment brief : DSD605\_2**

**Create a copy of the bike website you created in DSD602.**

**We are going to modify it to include authorization and authentication.**

1. Set the following Password properties to make it easier for debugging

• RequireDigit = false;

• RequireLowercase = false;

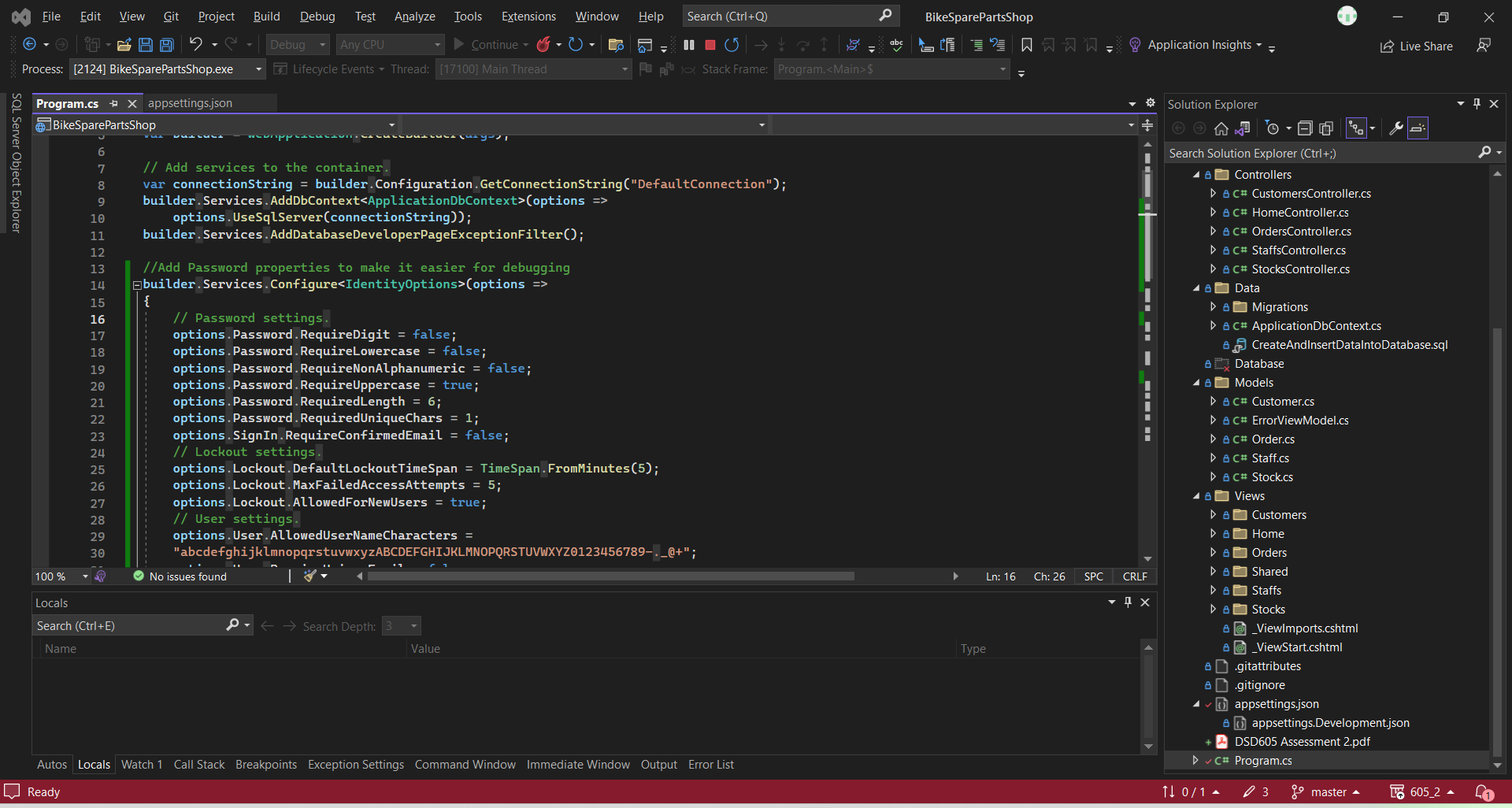
• RequireNonAlphanumeric = false;

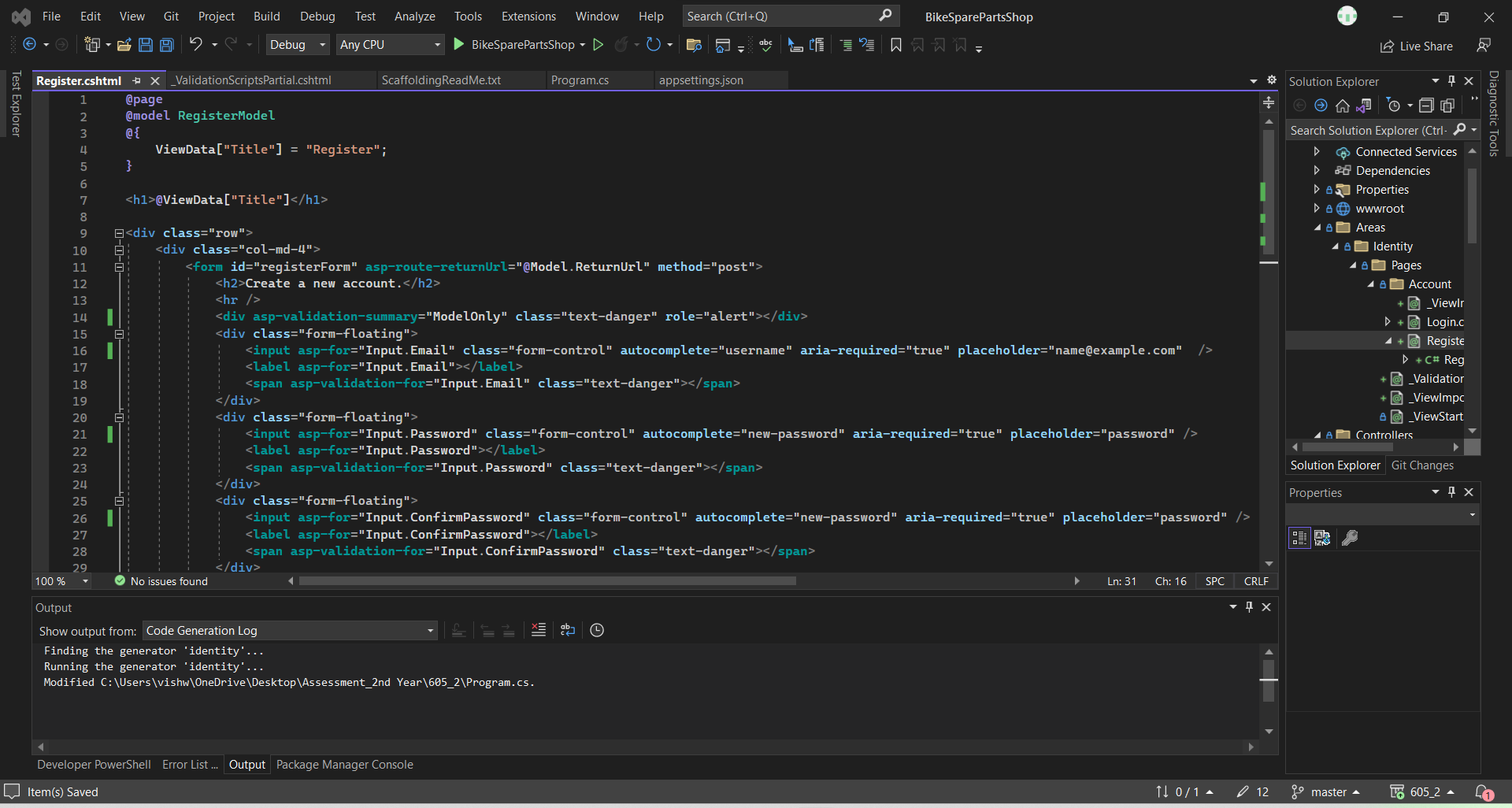
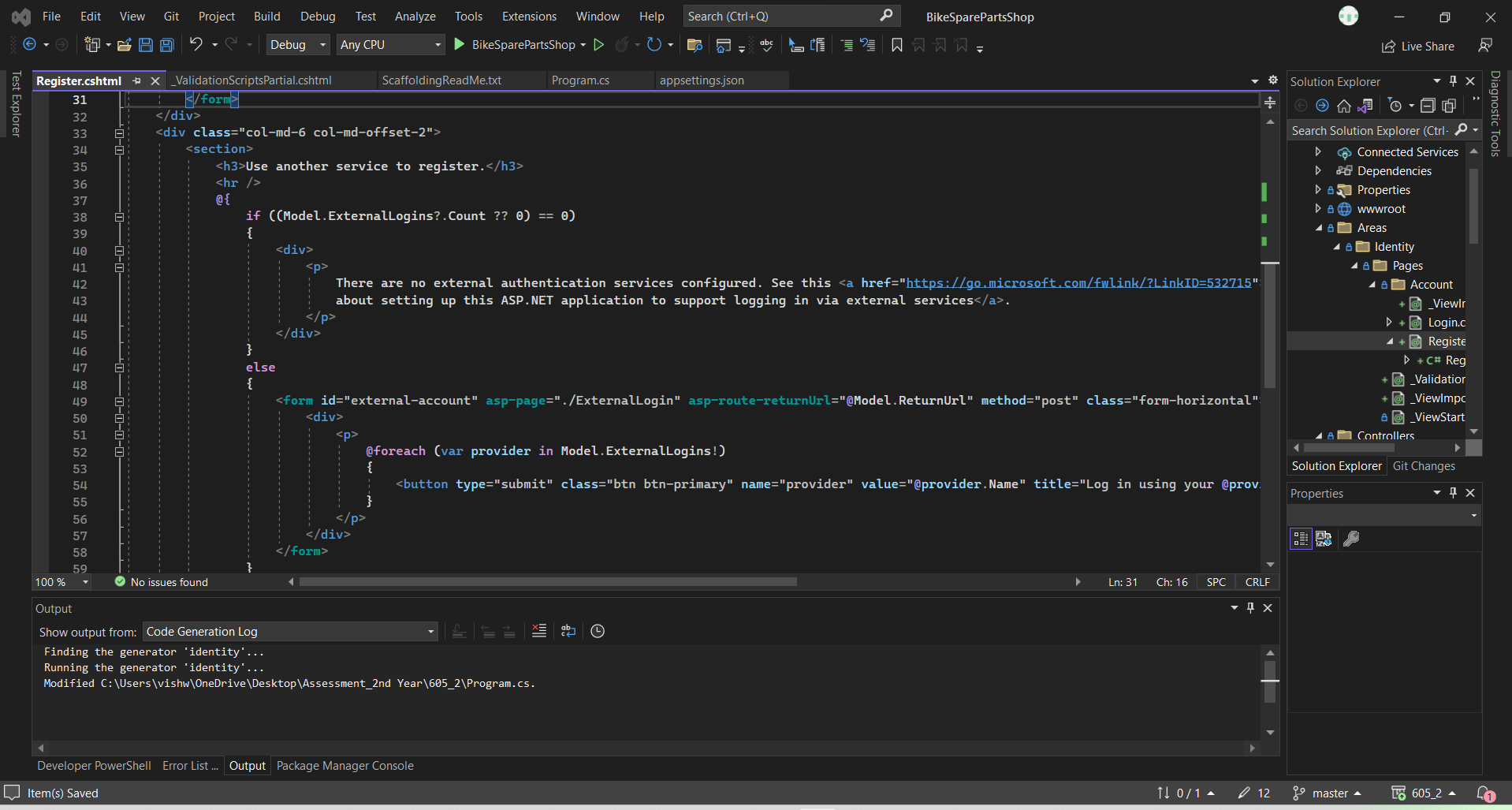
• RequireUppercase = true;

• RequiredLength = 6;

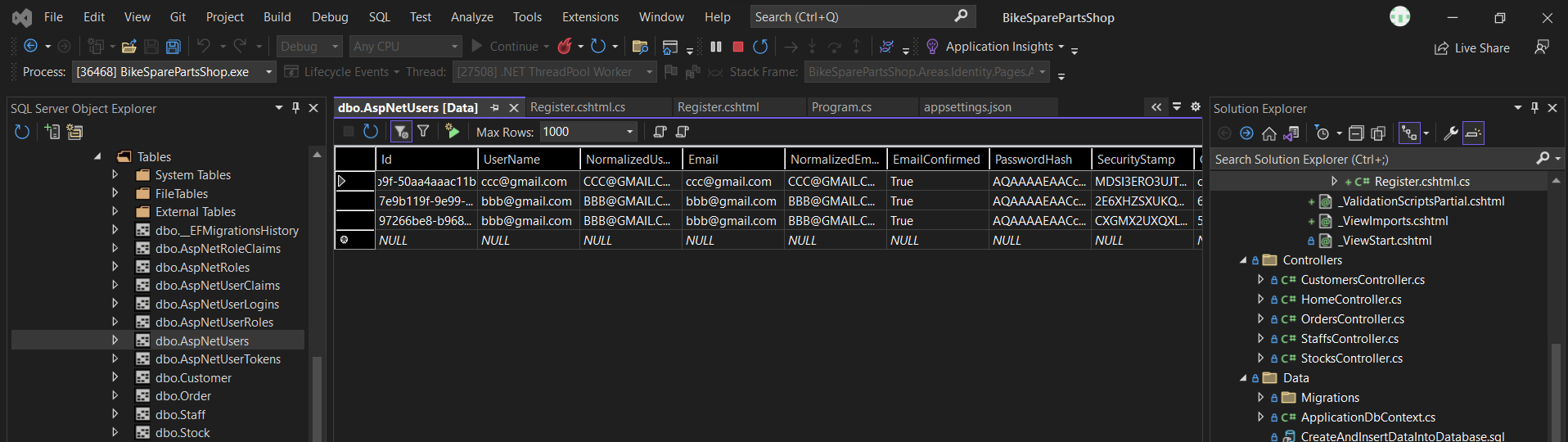
• RequiredUniqueChars = 1;

• RequireConfirmedEmail = false;

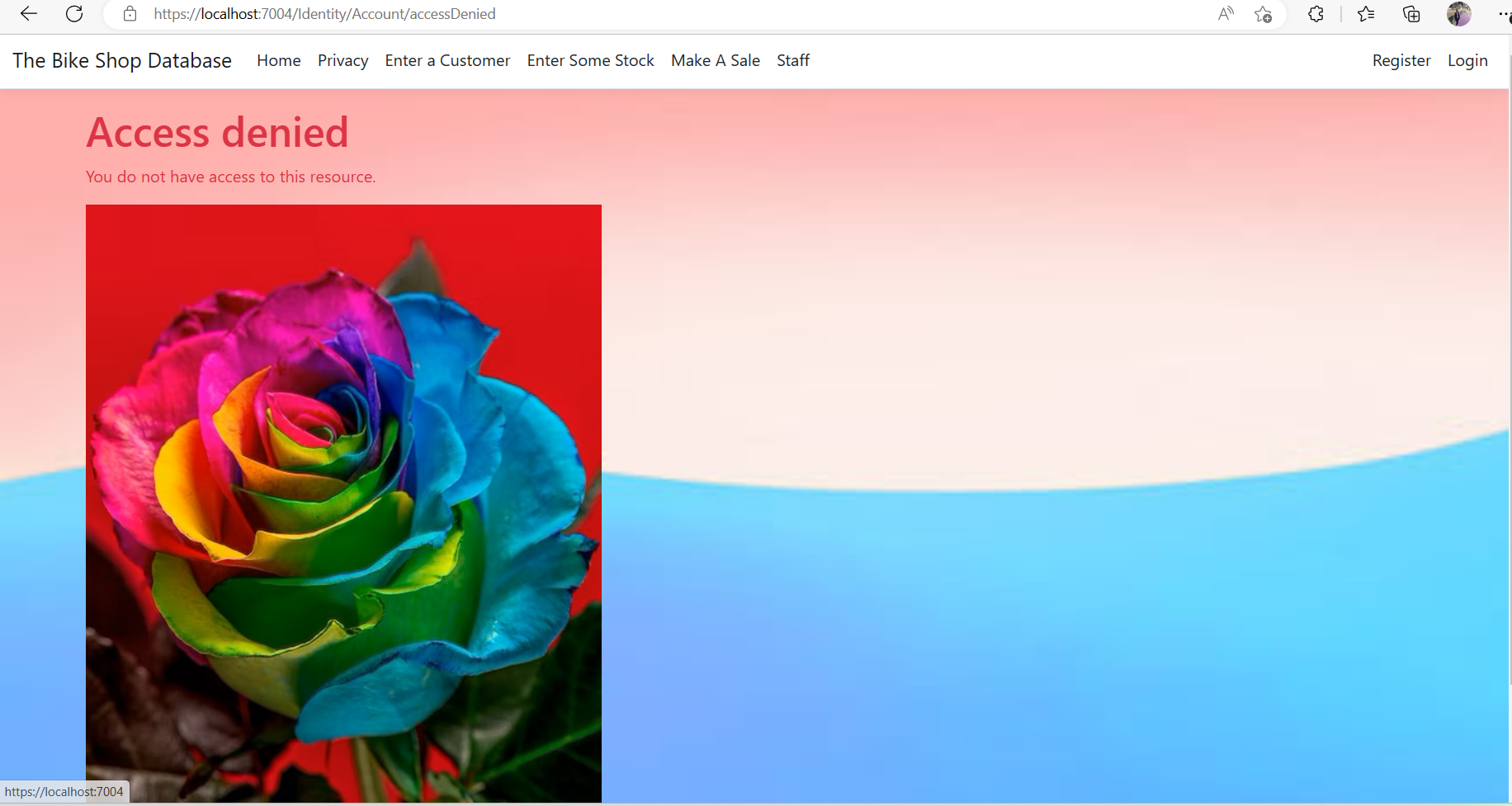




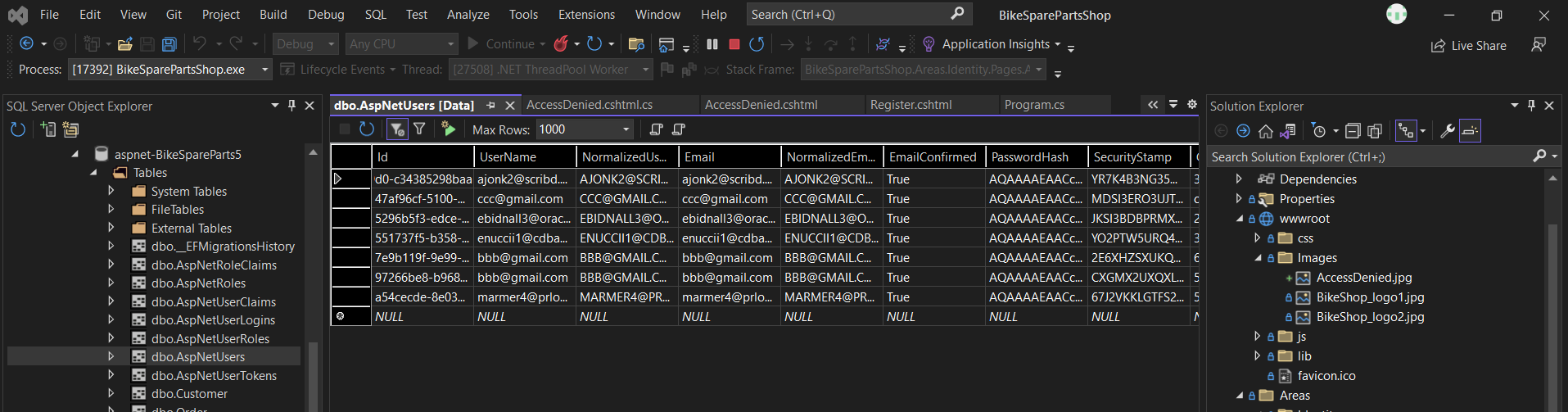
2. Modify the Register Page to automatically set confirm email to true



3. Customise the Access Denied page to add an image of your own choice, something unique



4. Add the following staff into your ASPNetUsers table via the Identity Register system(These guys are in your staff table already)



5. Create the roles of Admin, Manager, Staff, using the RoleManager interface.

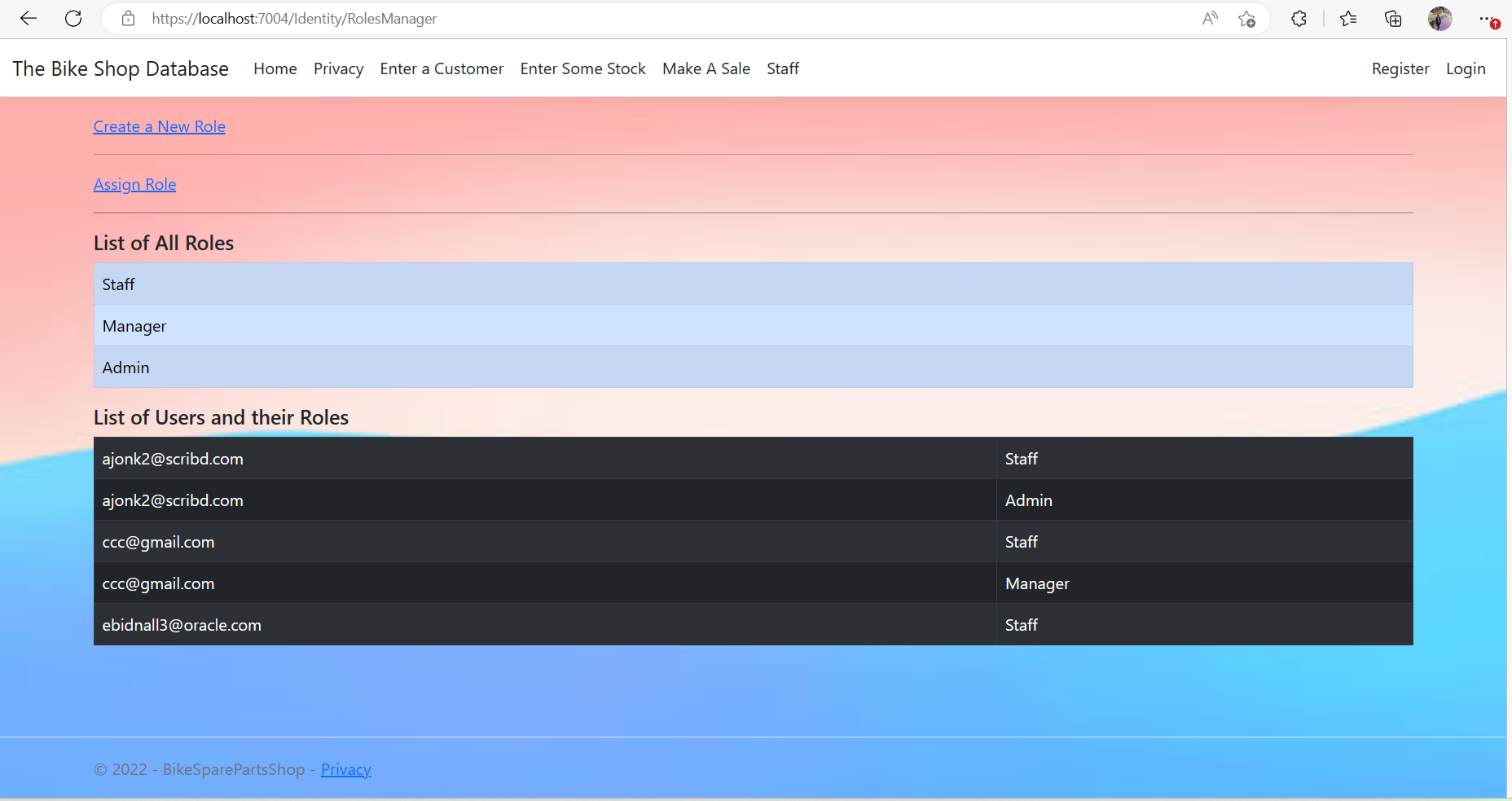
• Eddie: Manager

• Alasdair: Manager

• Phillipa: Admin

• Elsworth: Staff

• Maye: Staff

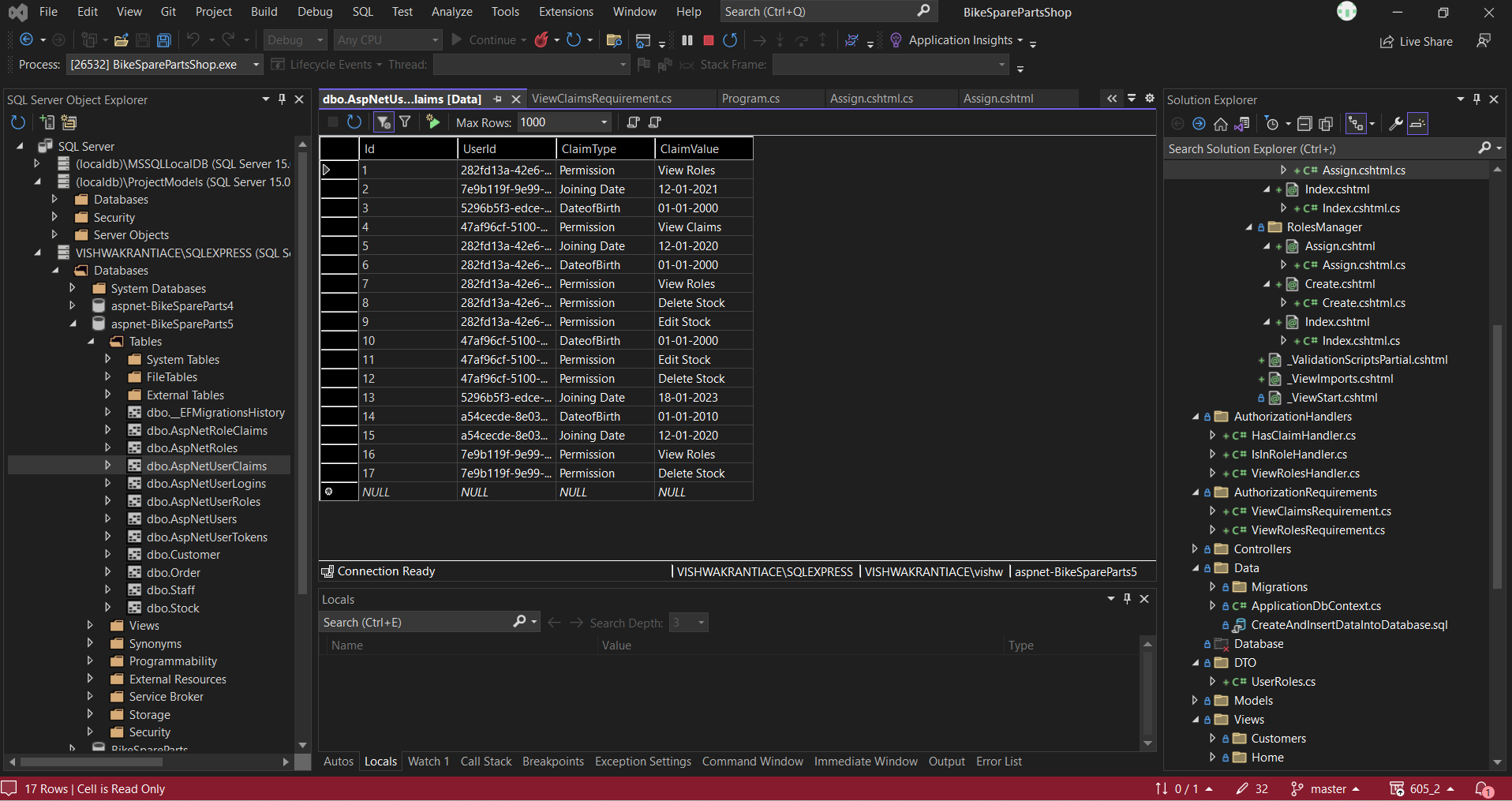


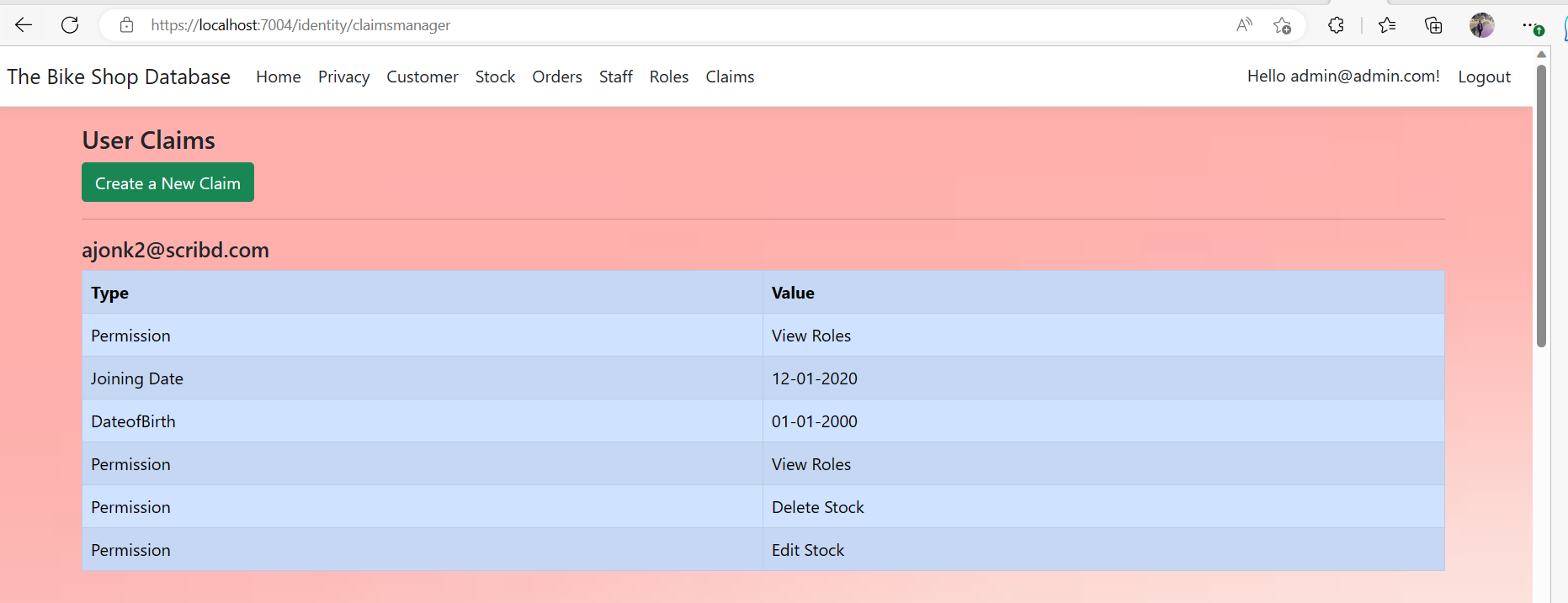
Create the claims below, using the ClaimsManager interface

6. Create the following staff User Claims



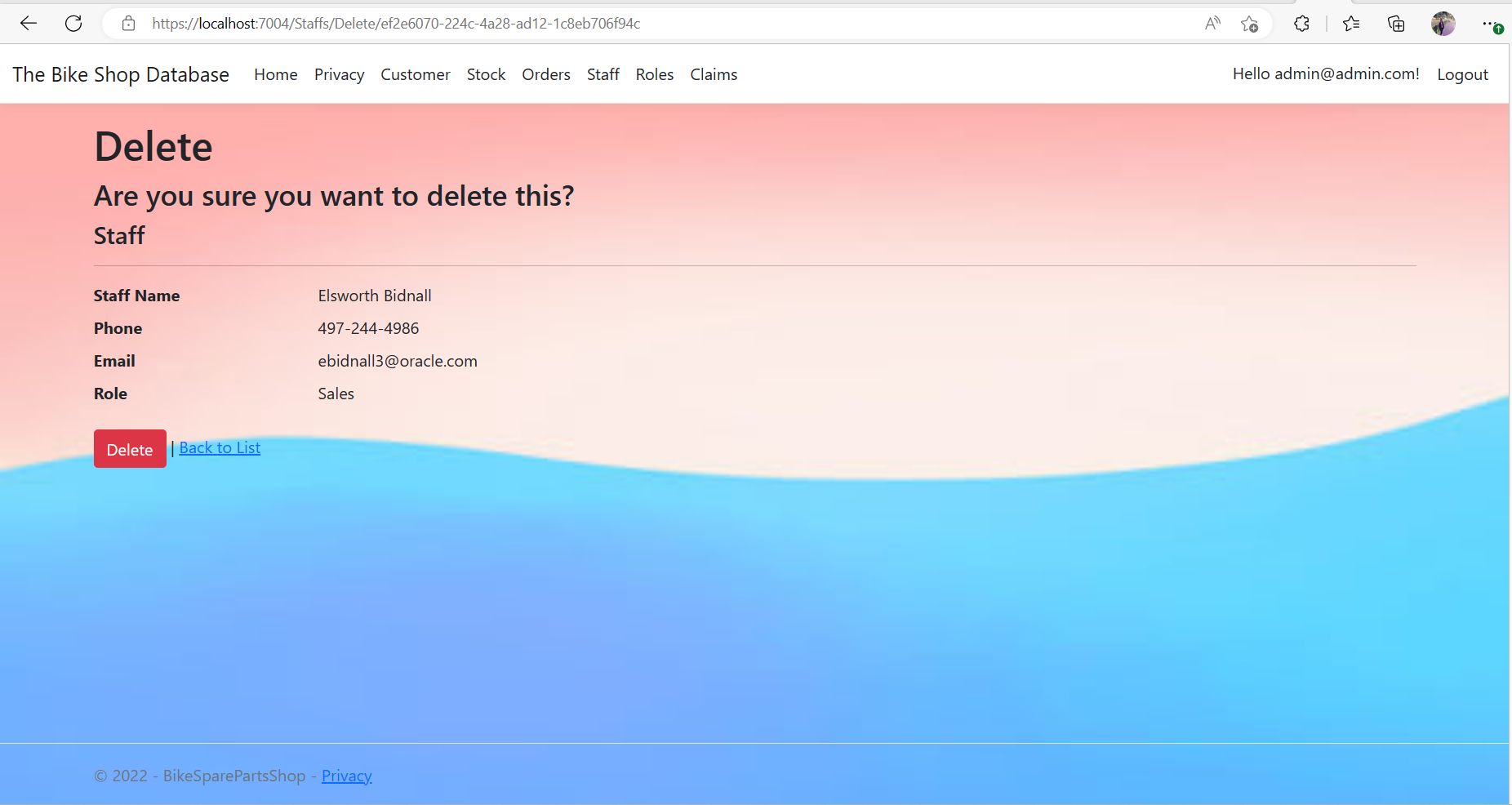


****

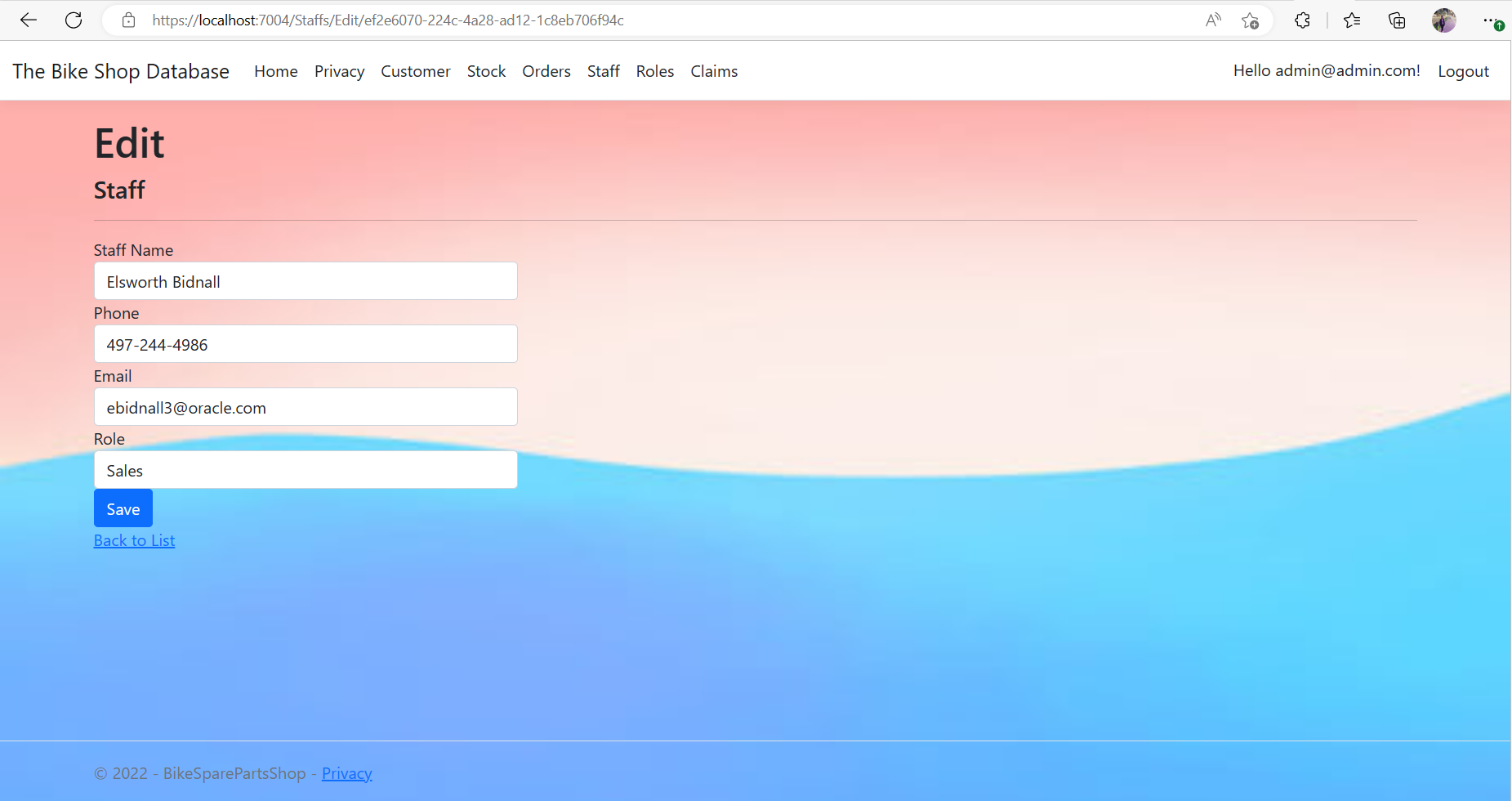
* 1. 7. Create the following policies and apply them.
  2. i) Staff with over 6 months of service and permission “View Roles” can view RolesManager folder
  3. 
  4. ii) Staff with over 6 months of service and permission “View Claims” can view ClaimsManager Folder



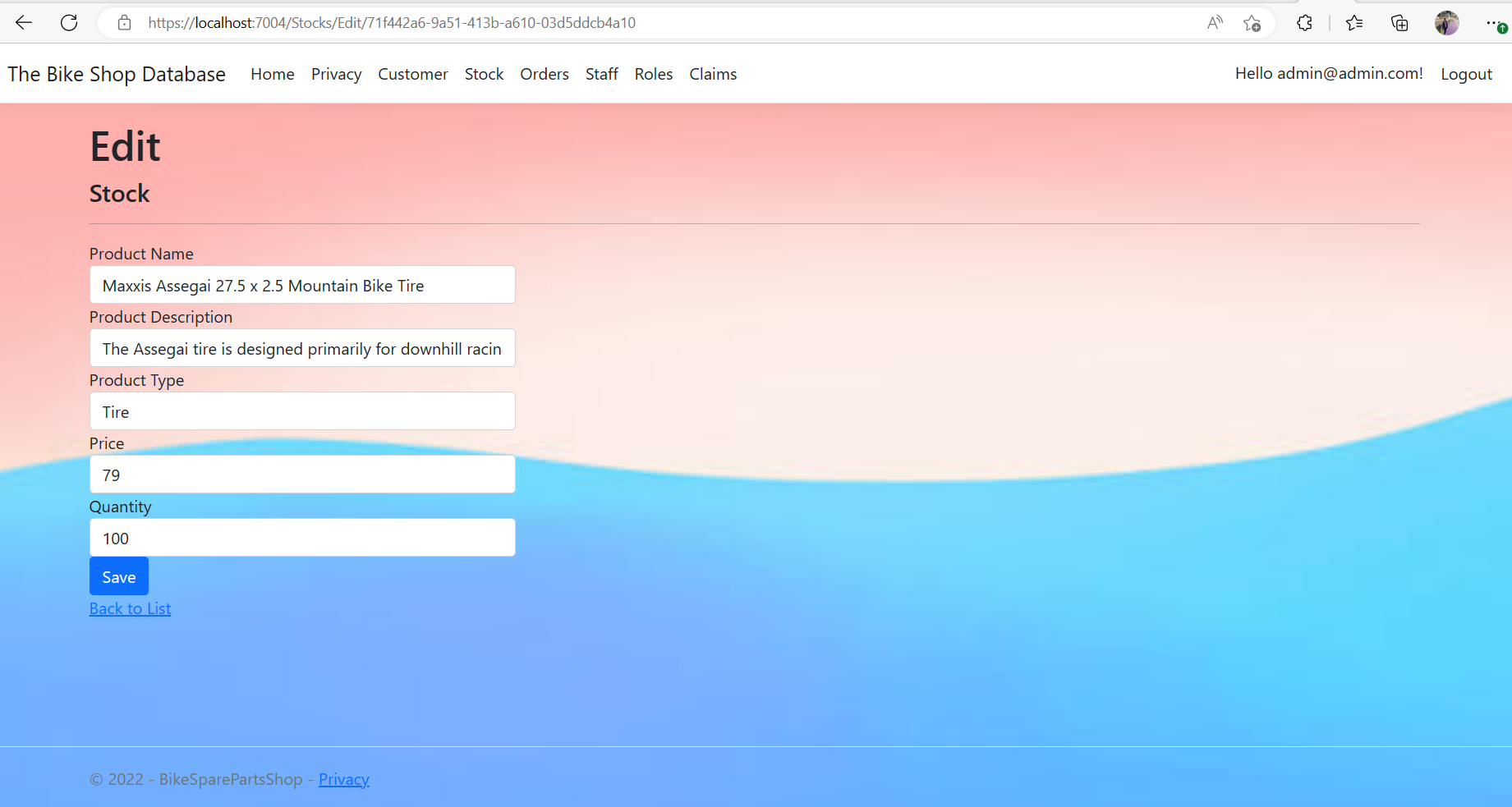
* 1. iii) Staff with permission Delete Stock can Delete Stock



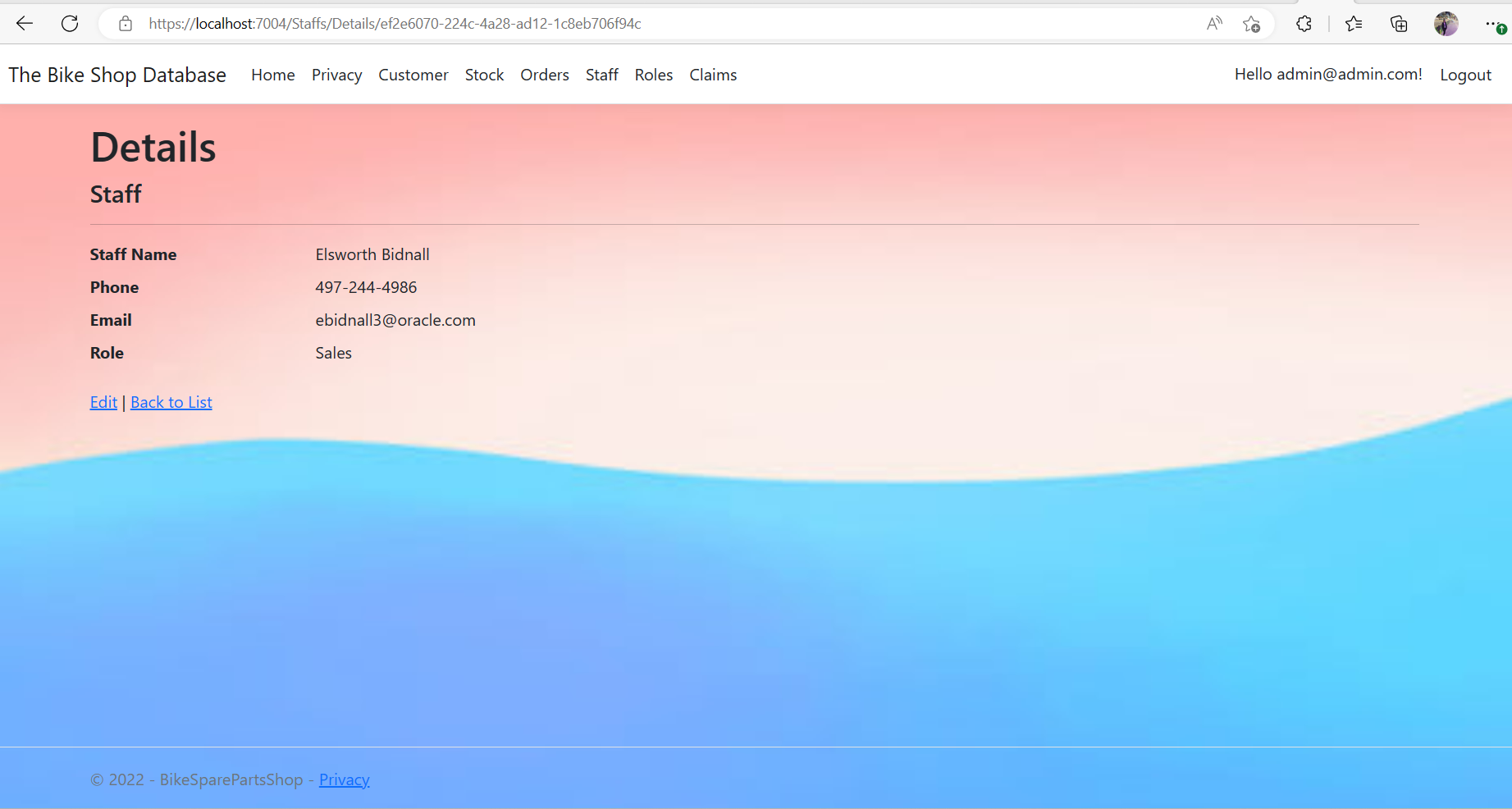
* 1. iv) Staff with permission Edit Stock can Edit Stock



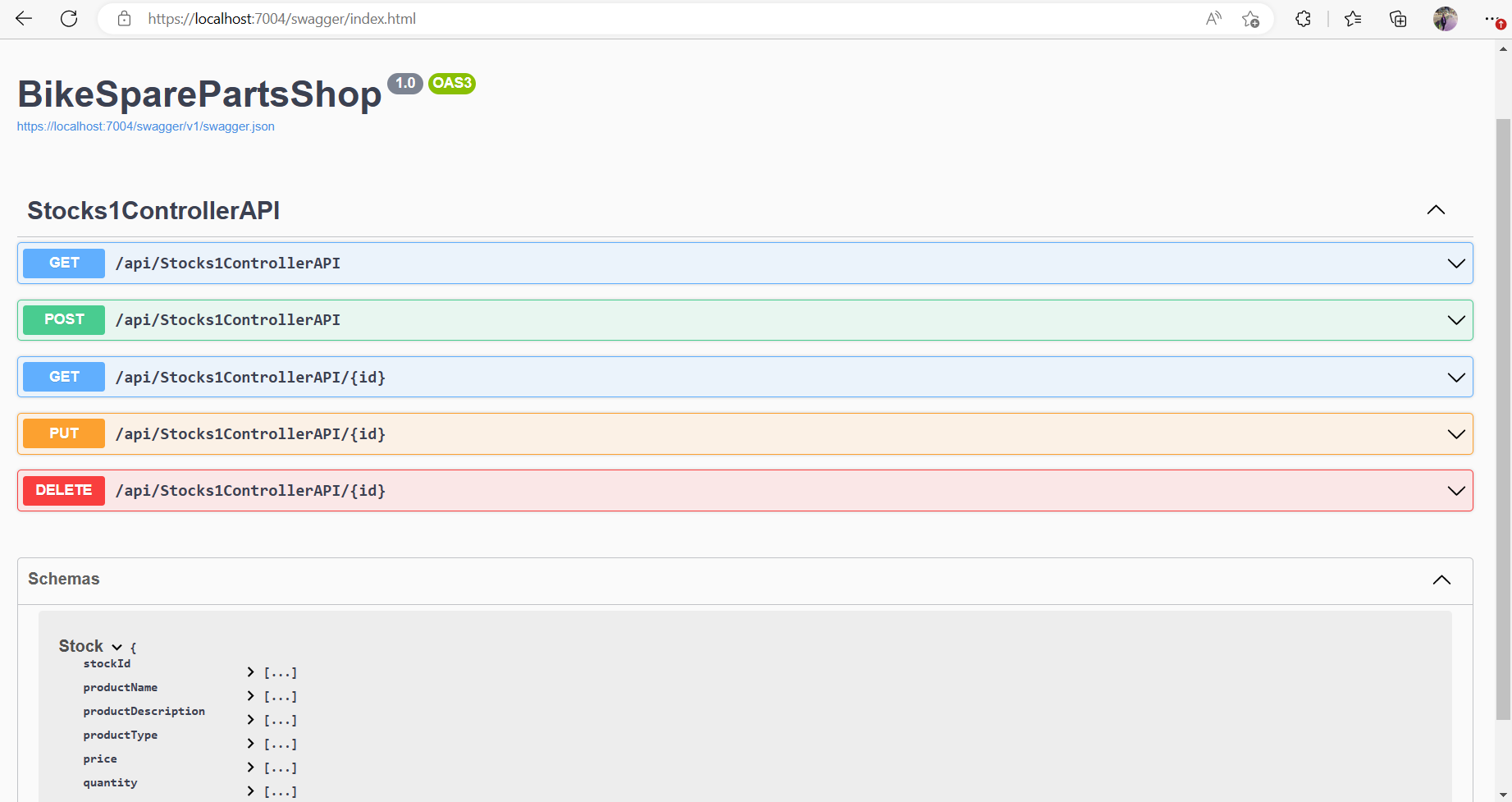
* 1. v) Staff over 18 can add stock



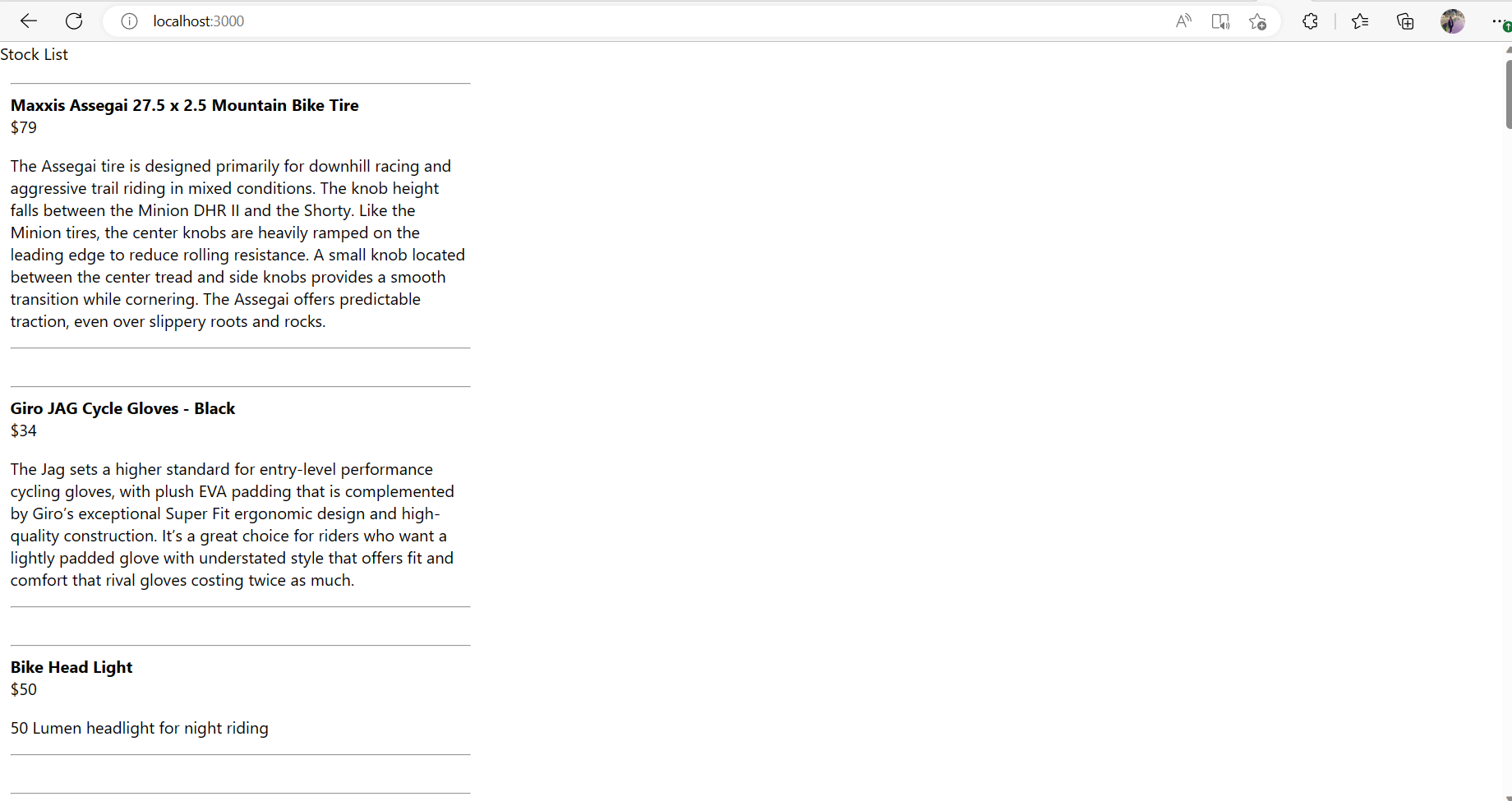
* 1. vi) Staff in Admin Role can view the Staff details by controlling the StaffsController.



8. Generate a WebAPI Framework, add a StockAPIController and install Swagger / Swatchbuckle



9. Create a ReactJs front end that consumes the Web API.



10. Outline the Purpose of CORS

CORS is a mechanism that allows restricted resources on a web page to be requested from another domain outside the domain from which the first resource was served. A web page may freely embed cross-origin images, stylesheets, scripts, iframes, and videos.

Cross-origin resource sharing (CORS) is a mechanism that allows restricted resources on a web page to be requested from another domain outside the domain from which the first resource was served. A web page may freely embed cross-origin images, stylesheets, scripts, iframes, and videos.

11. Outline how CORS operates and the types of restrictions it offers.

“CORS” stands for ***C***ross*-****O***rigin***R***esource***S***haring. It allows you to make requests from one website to another website in the browser, which is normally prohibited by another browser policy called the Same-Origin Policy (SOP).

 CORS and SOP are both browser policies that have developed in response to issues of browser security and vulnerabilities. The specific browser vulnerability that Same Origin Policy is meant to address is called “cross-site request forgery” (CSRF, or alternatively XSRF, don’t you love all these acronyms?).

Before browsers implemented SOP, malicious websites were able to exploit cookies stored by your browser to make unauthorized requests to other domains. Some of these unauthorized requests could do things like make purchases, delete user information, fetch sensitive data, etc.

**Types of CORS requests**

There are two types of cross-origin requests: simple requests and pre-flighted requests.

**Simple requests**

A simple request is one that uses methods such as GET, HEAD, or POST. These methods are considered safe because they are not capable of causing a change in state on the server.

**Pre-flighted requests**

A pre-flighted request is one that uses a method such as PUT or DELETE. These methods can cause a change in state on the server, so the browser sends a request to the server to check if the request is allowed. The server then responds with the appropriate headers, and if the response is successful, the browser sends the actual request.

## CORS mechanism

The CORS mechanism works by adding HTTP headers to cross-origin HTTP requests and responses. These headers indicate whether the request or response is allowed to access the resources.

## CORS work:

When a browser sends a request to a server, it includes an Origin header. This header contains the origin of the request, which is the domain, protocol, and port of the page making the request.

The server can then decide whether to allow or deny the request. If the request is allowed, the server includes the Access-Control-Allow-Origin header in the response. This header specifies the origin that is allowed to access the resources.

If the request is denied, the server includes the Access-Control-Allow-Origin header with a value of "\*", which indicates that no origin is allowed to access the resources.

## CORS importance:

CORS is important because it allows browsers to enforce the same-origin policy. The same-origin policy is a security measure that prevents a malicious script from accessing resources that it should not have access to.

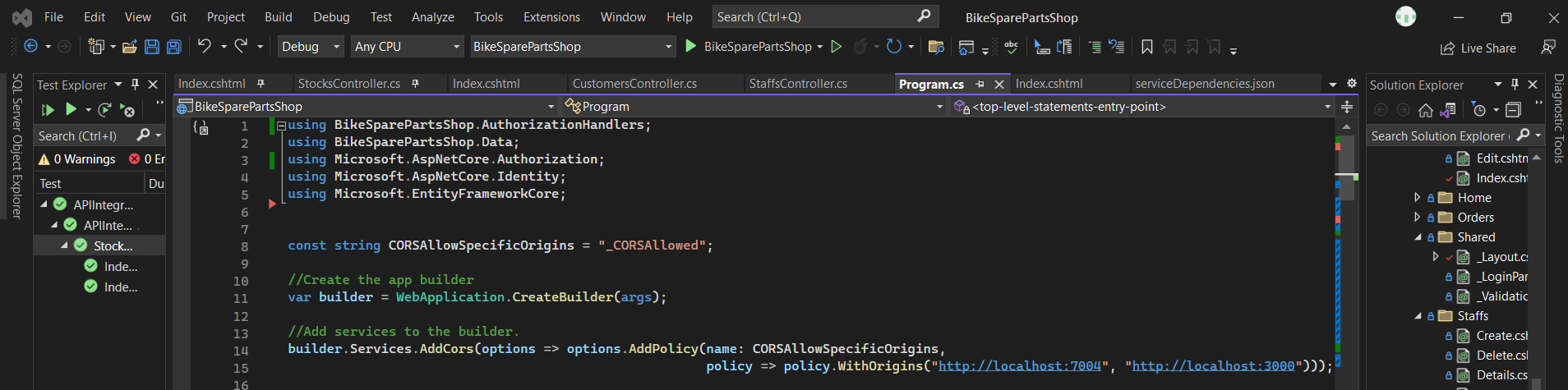
Without CORS, a malicious script could make a request to a server in another domain and access the resources that the user of the page is not intended to have access to.

## Benefits of CORS:

CORS provides a number of benefits:

* It allows browsers to enforce the same-origin policy, which is a security measure that prevents a malicious script from accessing resources that it should not have access.
* It allows restricted resources on a web page to be requested from another domain. This can be useful when you want to embed a resource from another domain, such as an image or a video.
* It allows the browser to send a pre-flighted request to the server to check if the request is allowed. This can be useful when you want to make a request that could potentially cause a change in state on the server.

12. Implement the CORS policy between them so that the front end consumes the data from the back end and displays it on your app.



13. Create a new Integration Project, name it APIIntergrationTest

Create and integration test for the StockAPIContoller that uses 1 mock entry and checks that it returns stock.

