For sql injection, we used the django ORM api to counter it.

For example, when we trying to get a customer’s information, instead of using sql language directly, we use the django’s API to do it.

test = Customer.objects.filter(emailID=request.user).first()

Django then converts the Python query to SQL query and communicates with the database.

And for the csrf attack, we use the form to send POST request and we also add the {%csrf\_token%} in the html.

For example, in the registration page, the part of individual registration form looks like this:

<form action="/registerIndi/" class="form-inline" style="margin-left : 430px" method="post">{% csrf\_token %}  
 <div class="form-group">  
 <label class="text-primary" for="inputEmail">Email</label>  
{# <div class="controls" >#}  
 <input id="inputEmail" aria-describedby="emailHelp" name = "inputEmail" type="text" placeholder="Enter email"/>  
  
 </div>

By using the form to send the post request and add the csrf token, we can prevent the csrf attack.