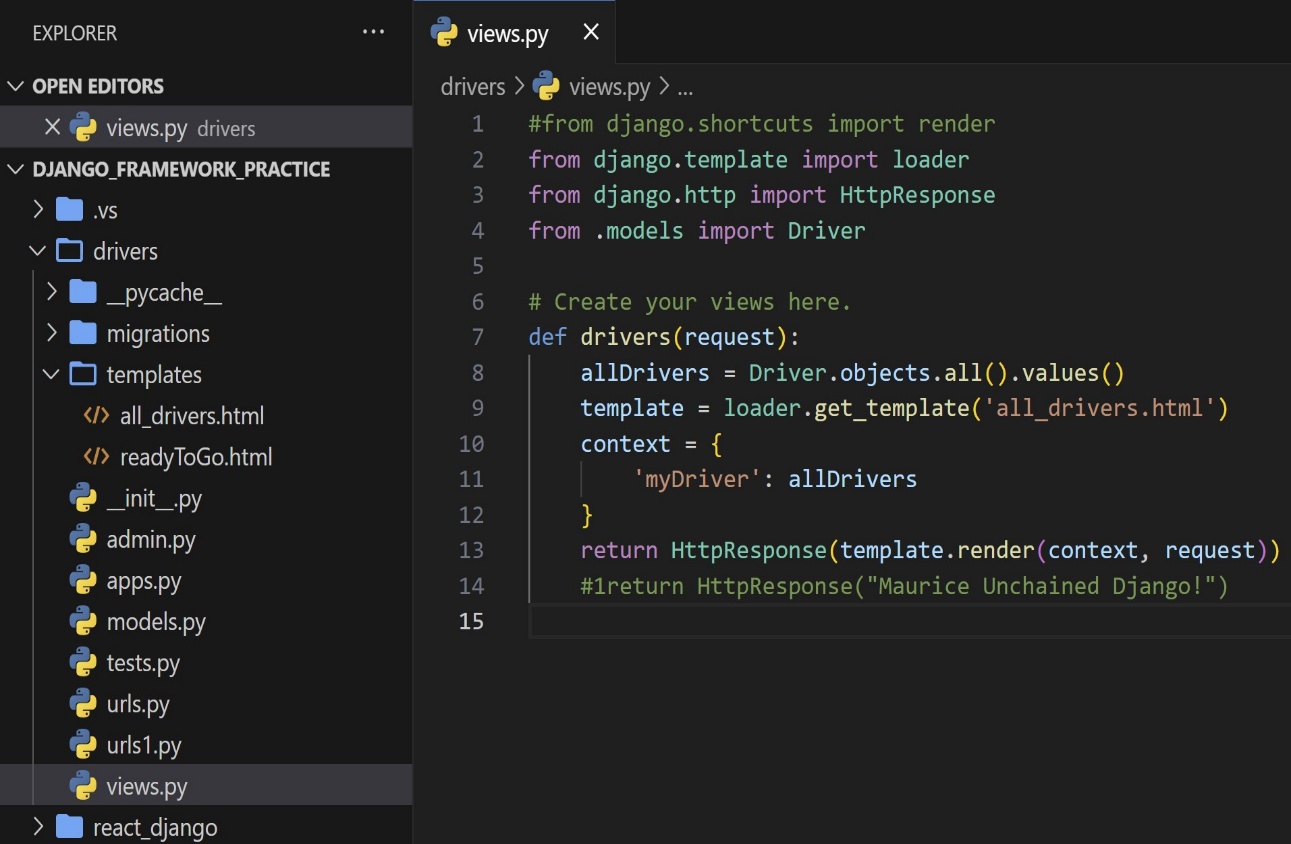
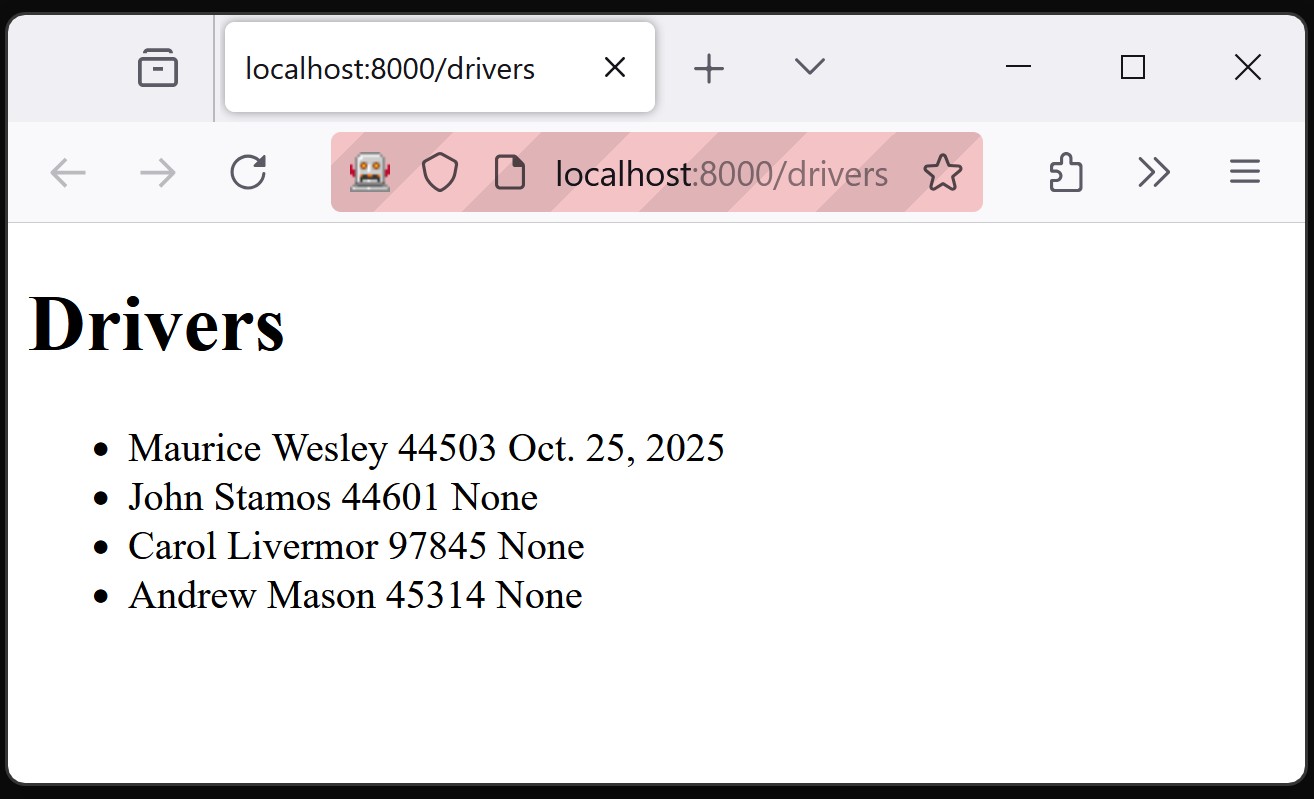
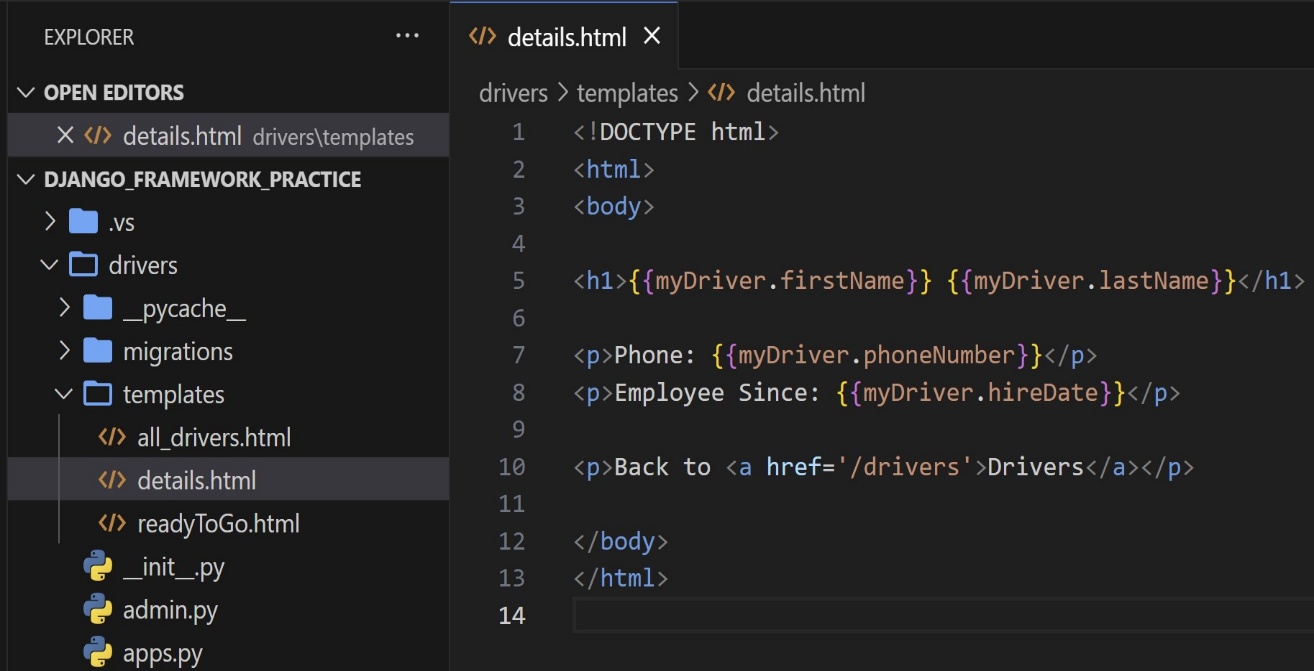
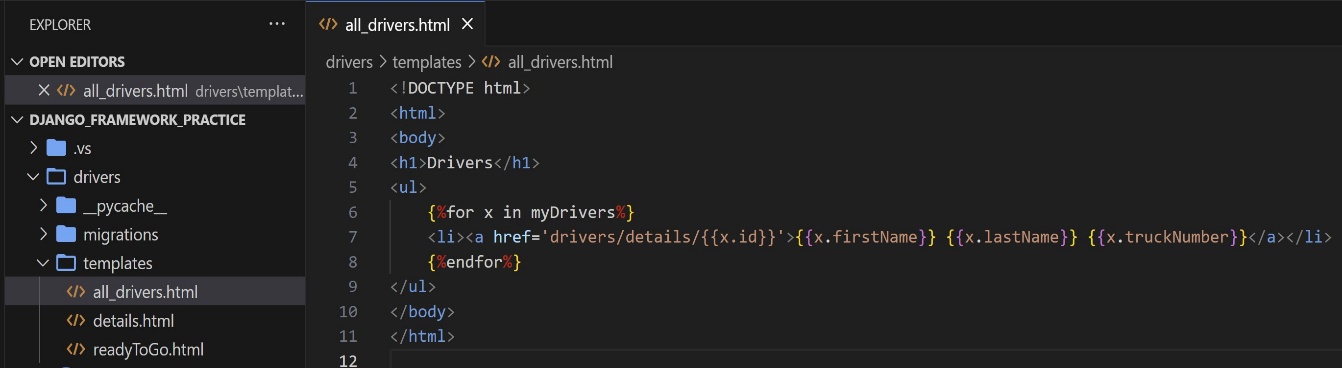
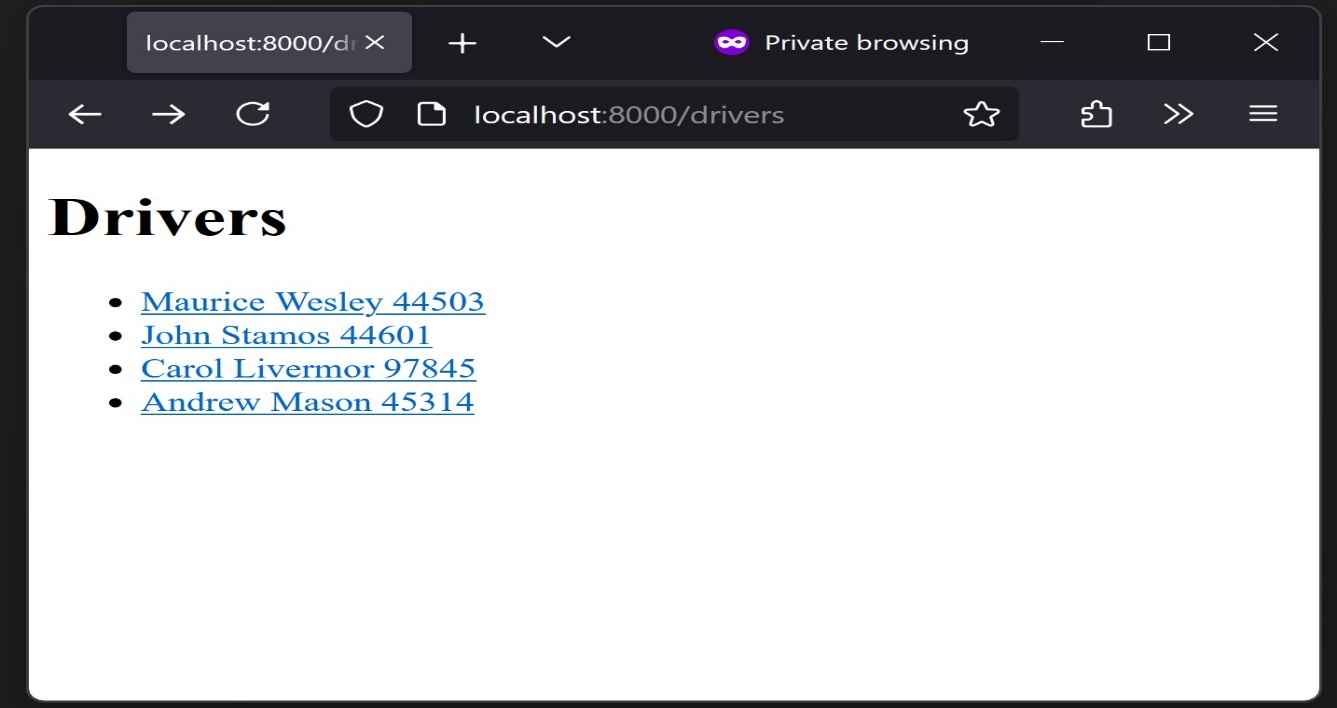
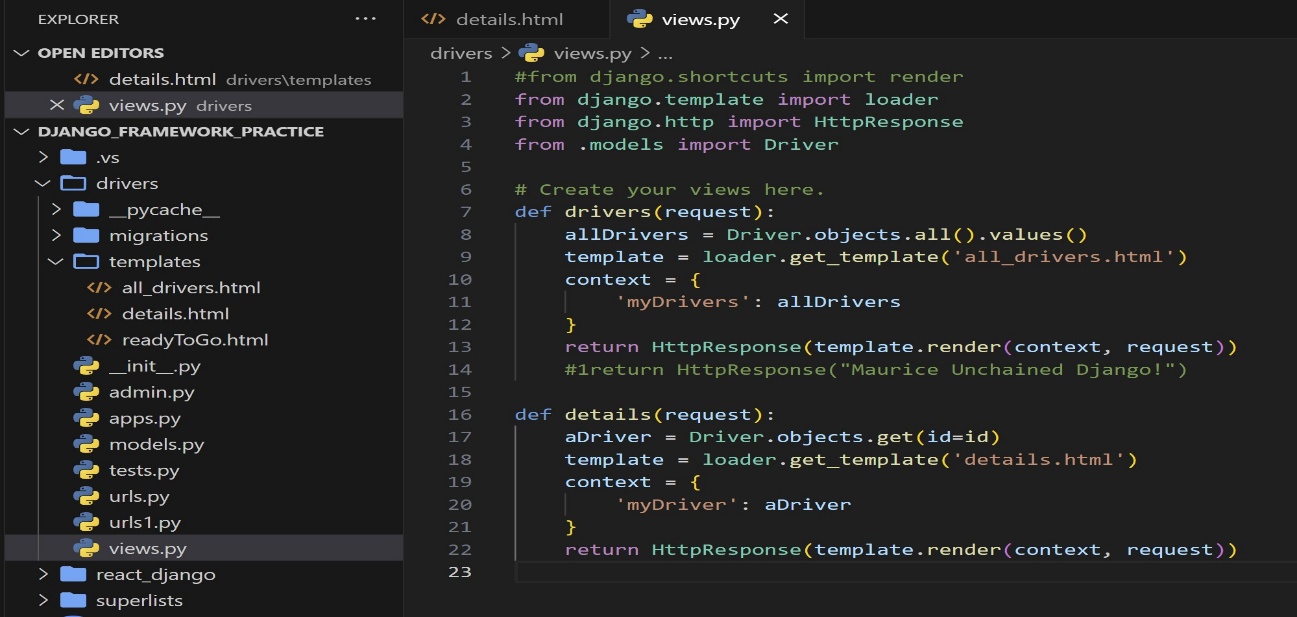
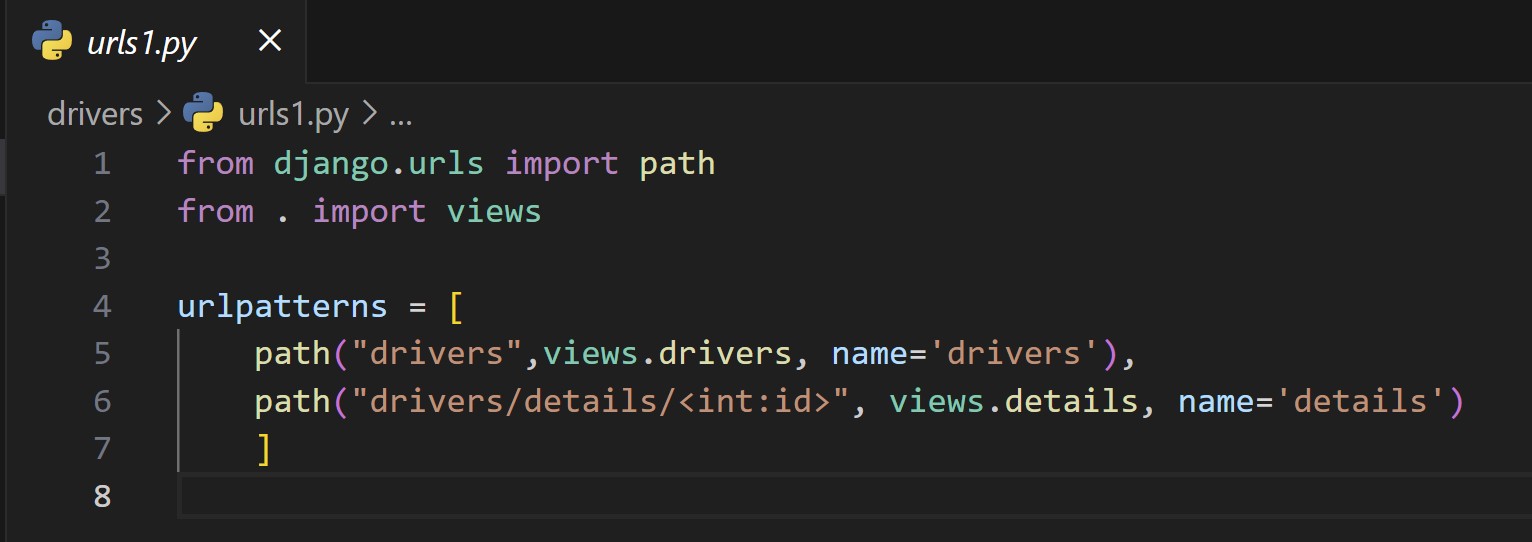
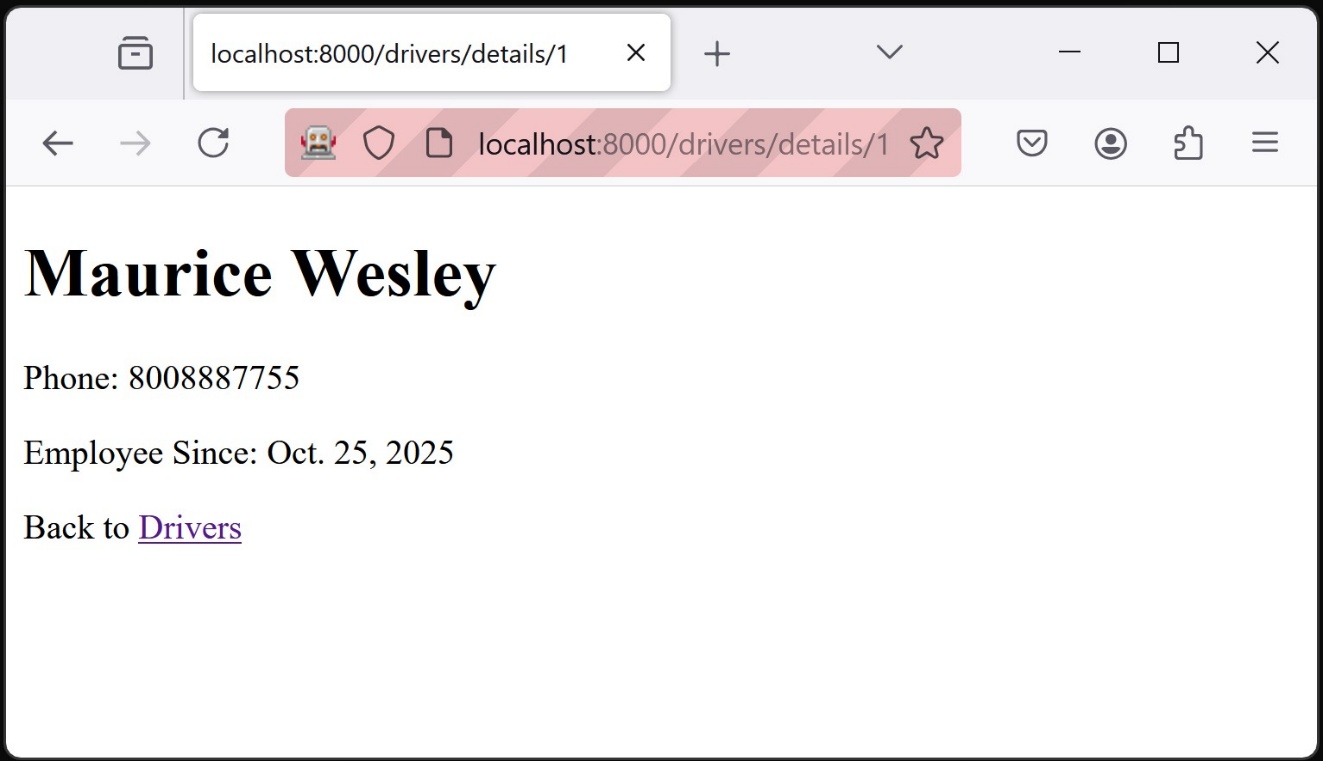
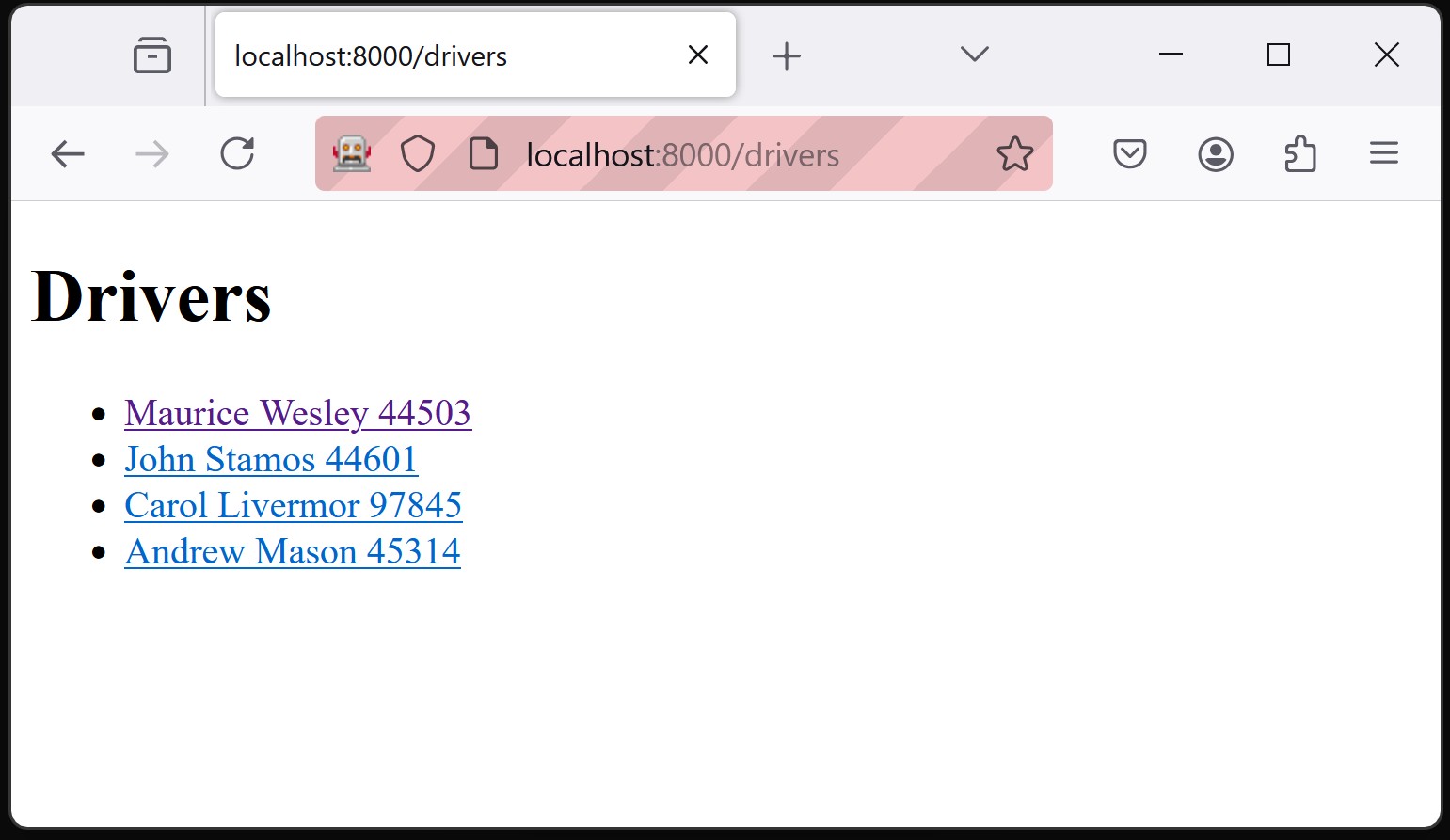
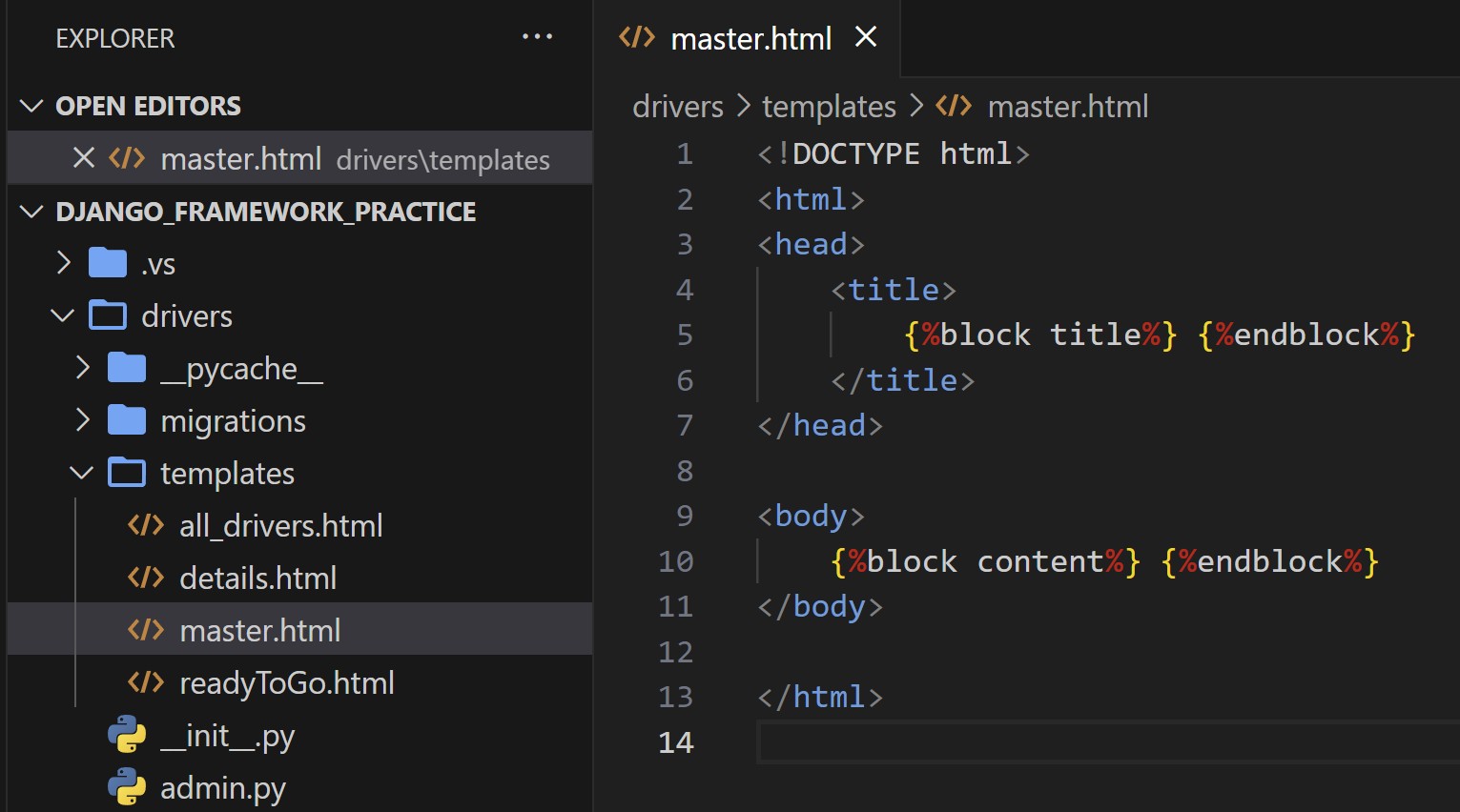
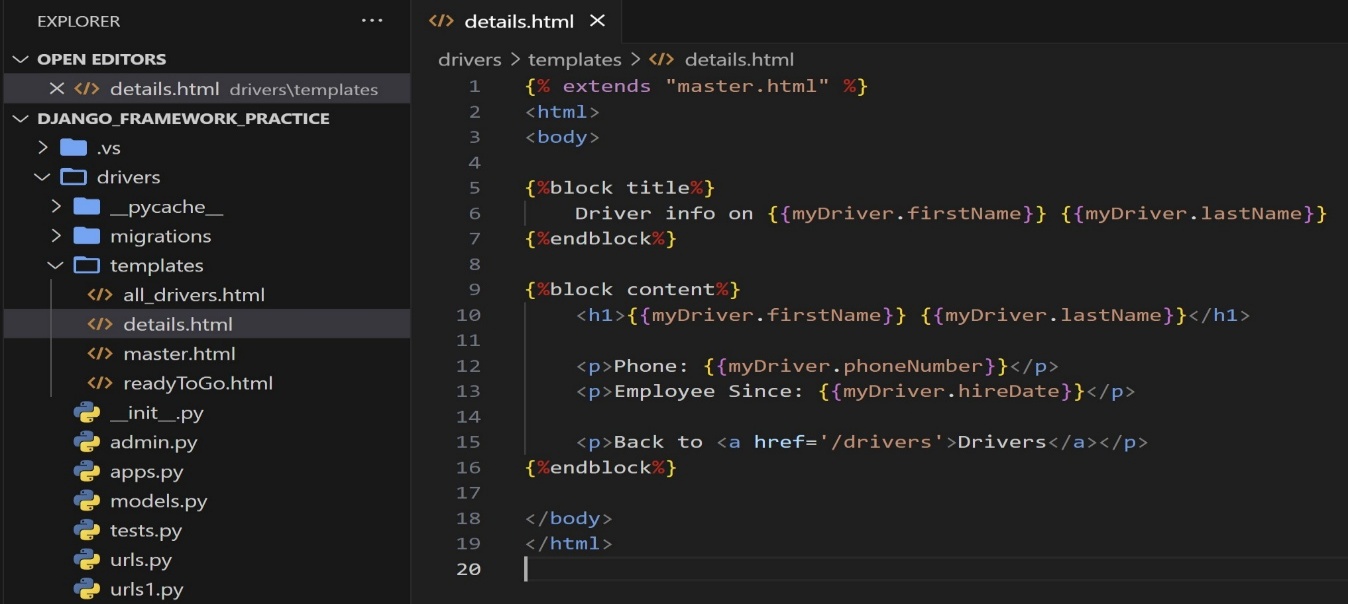
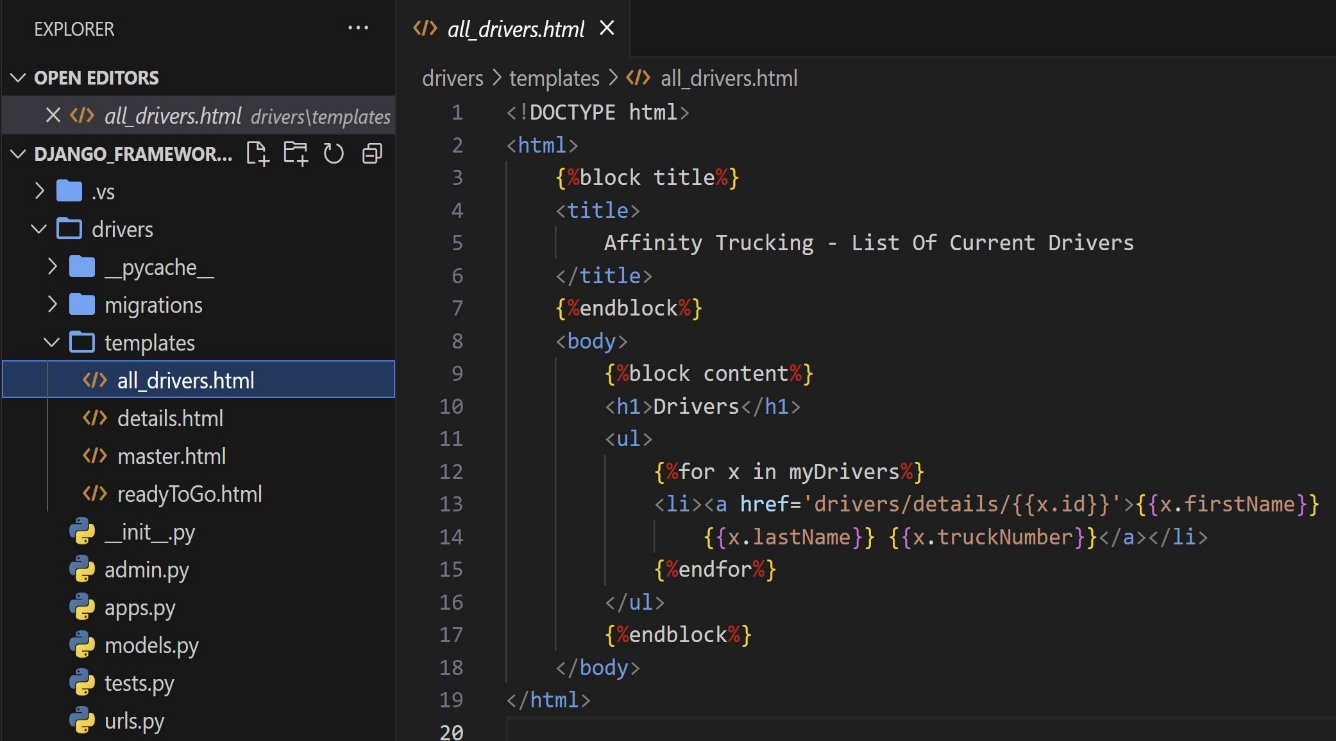
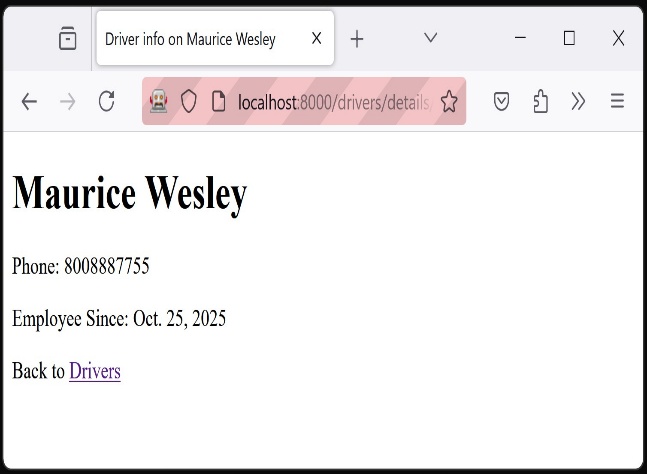
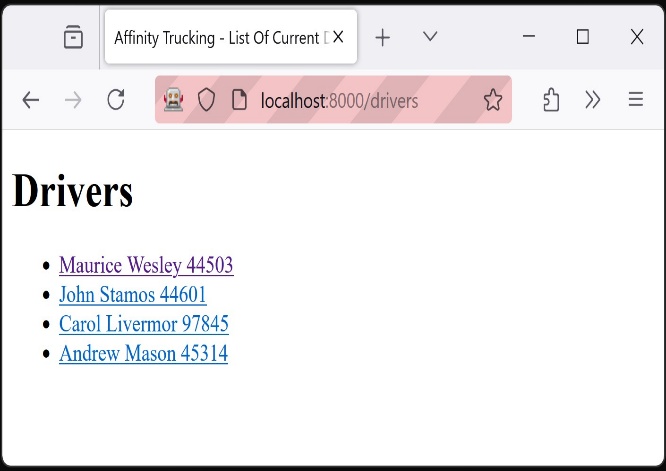
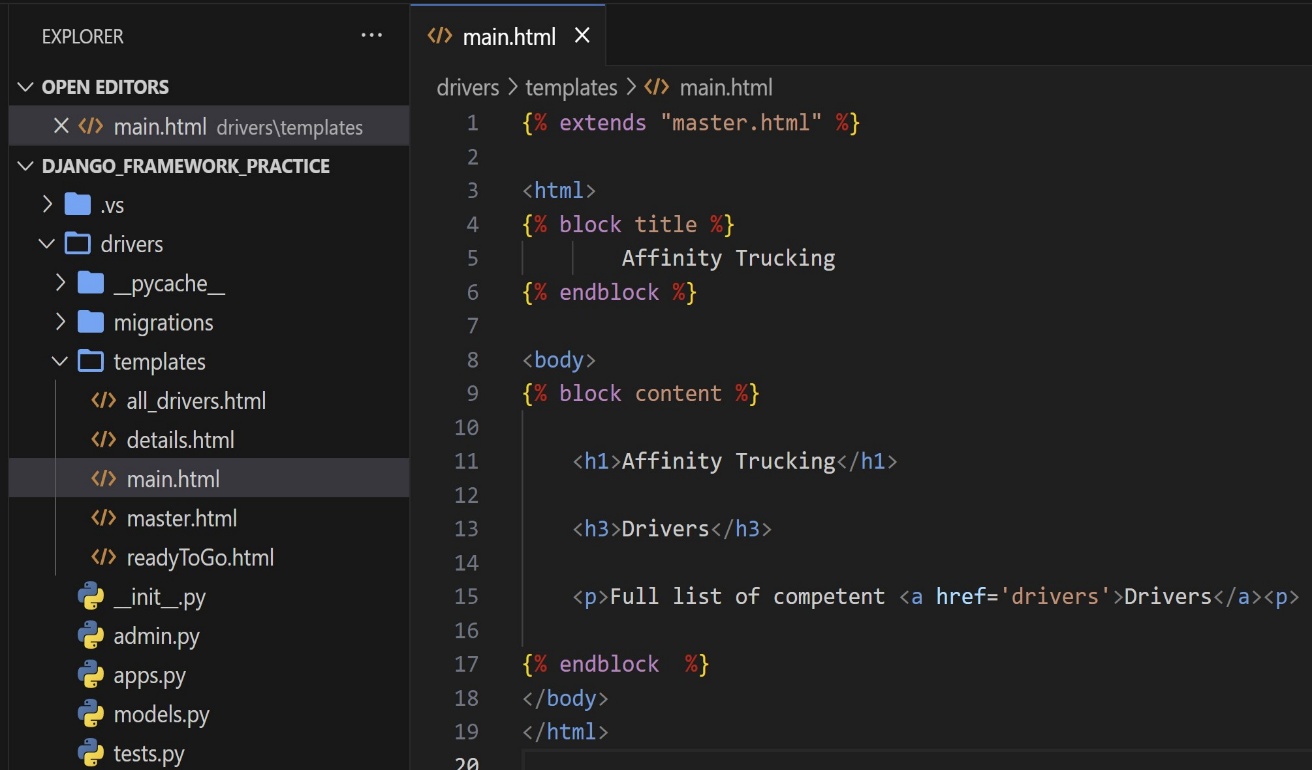
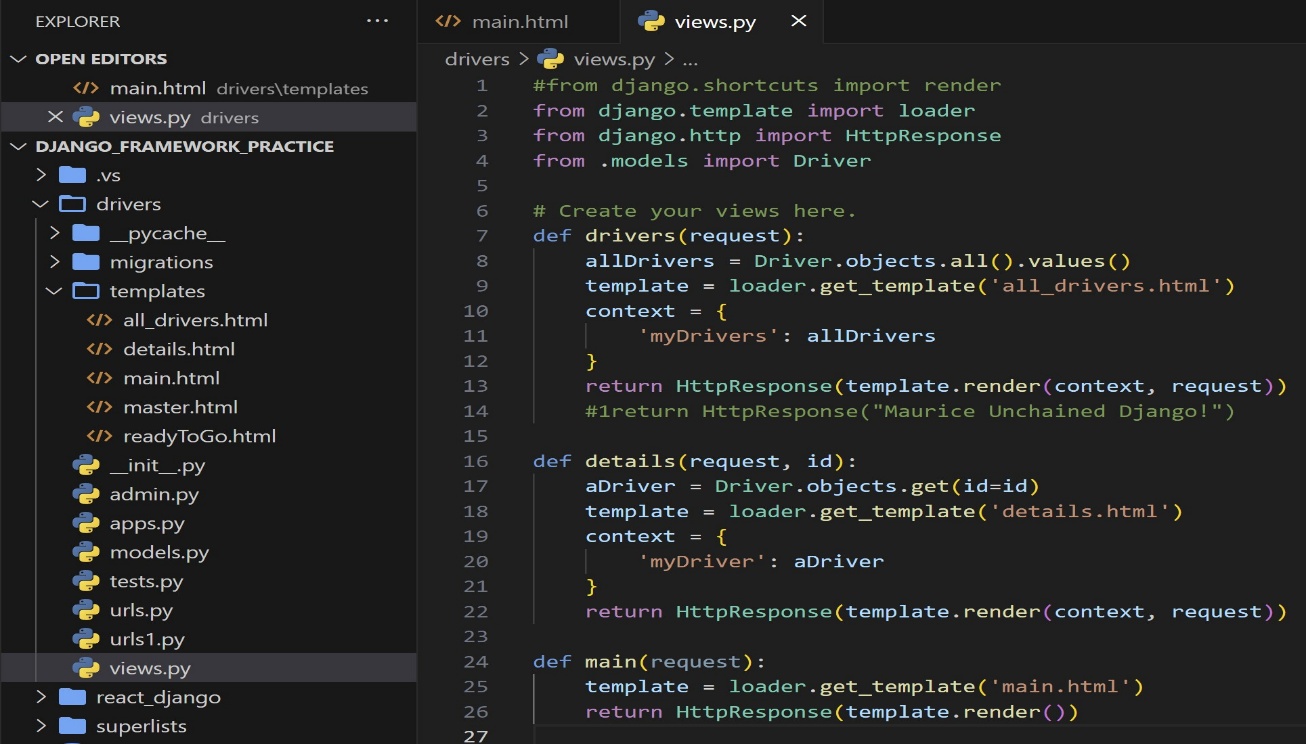
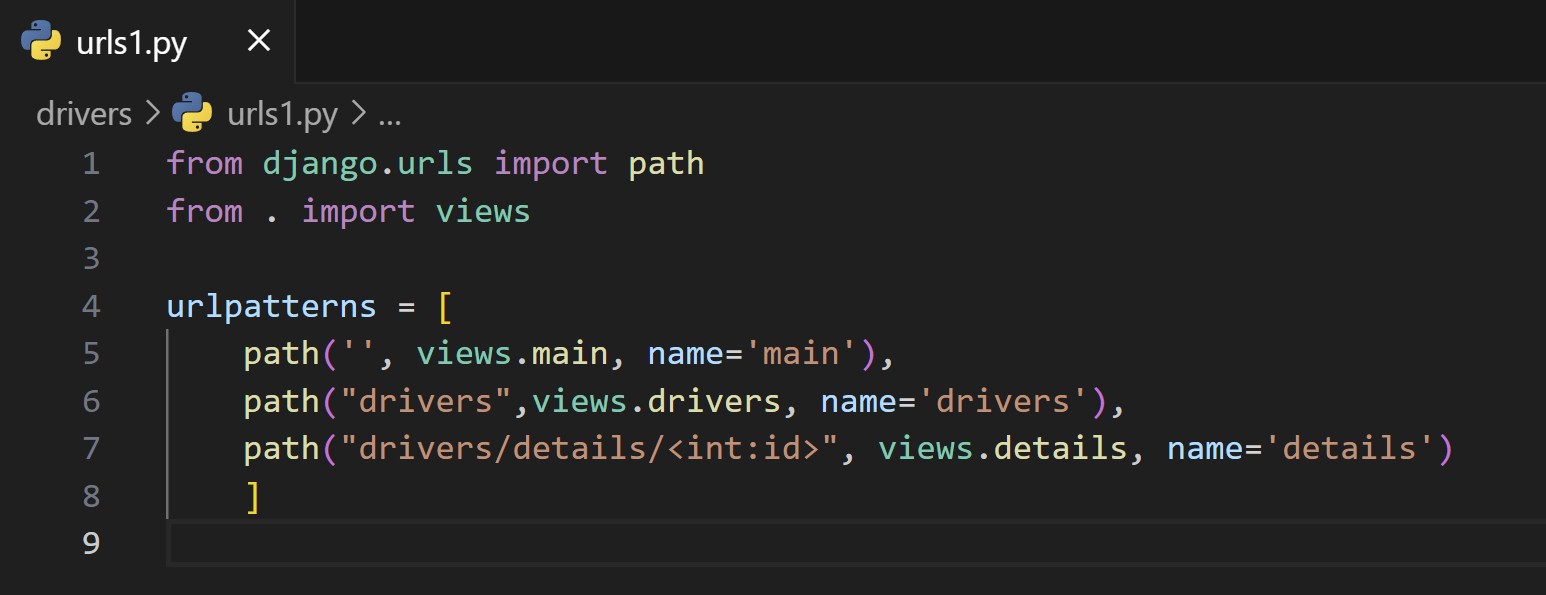
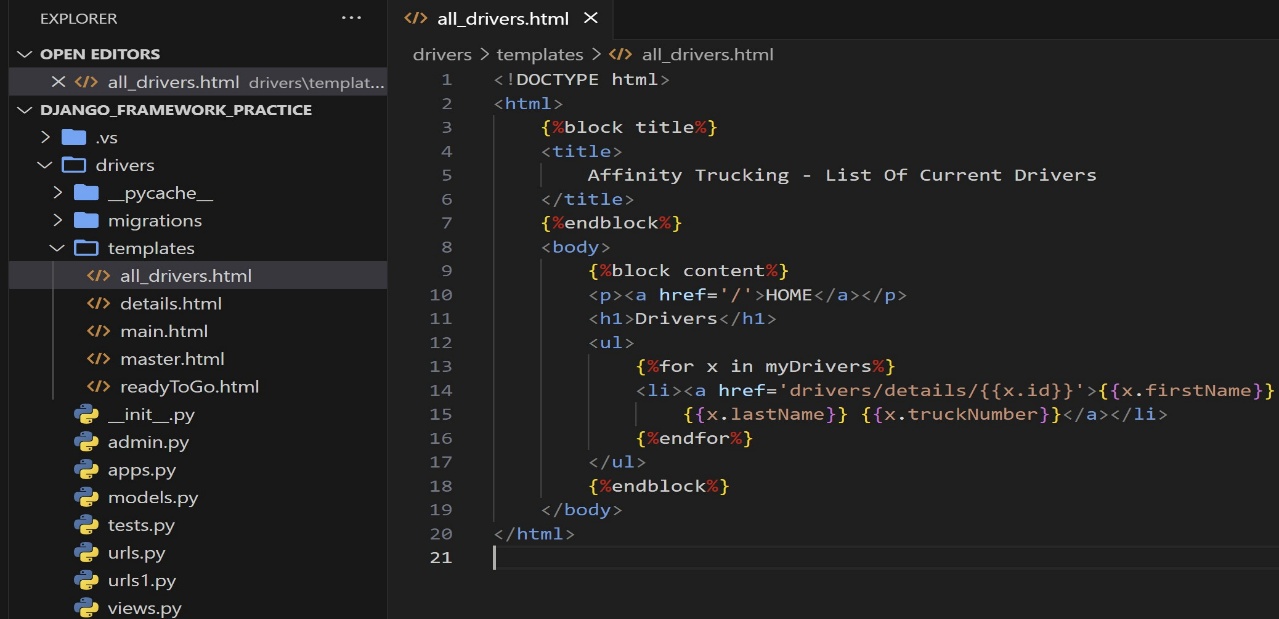
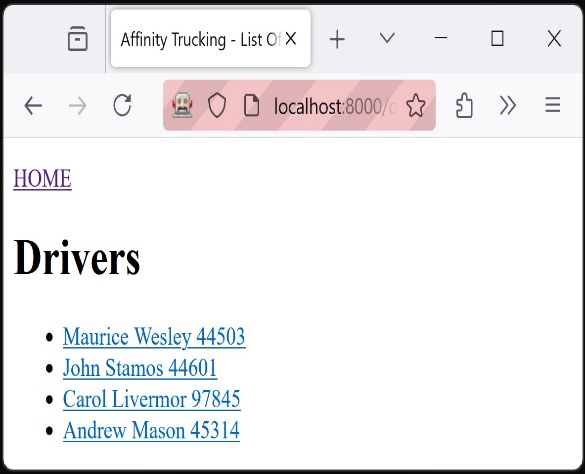
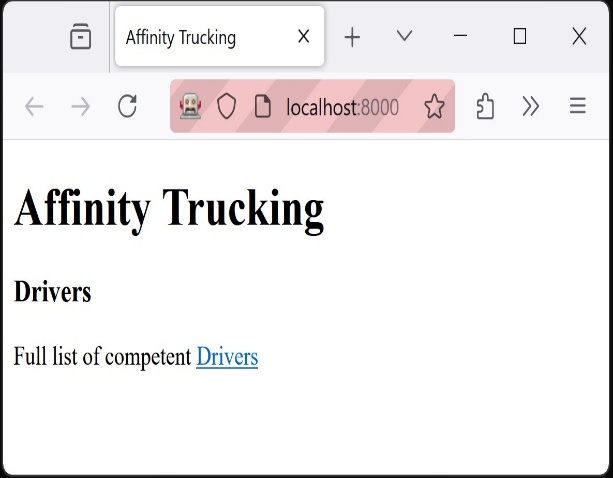
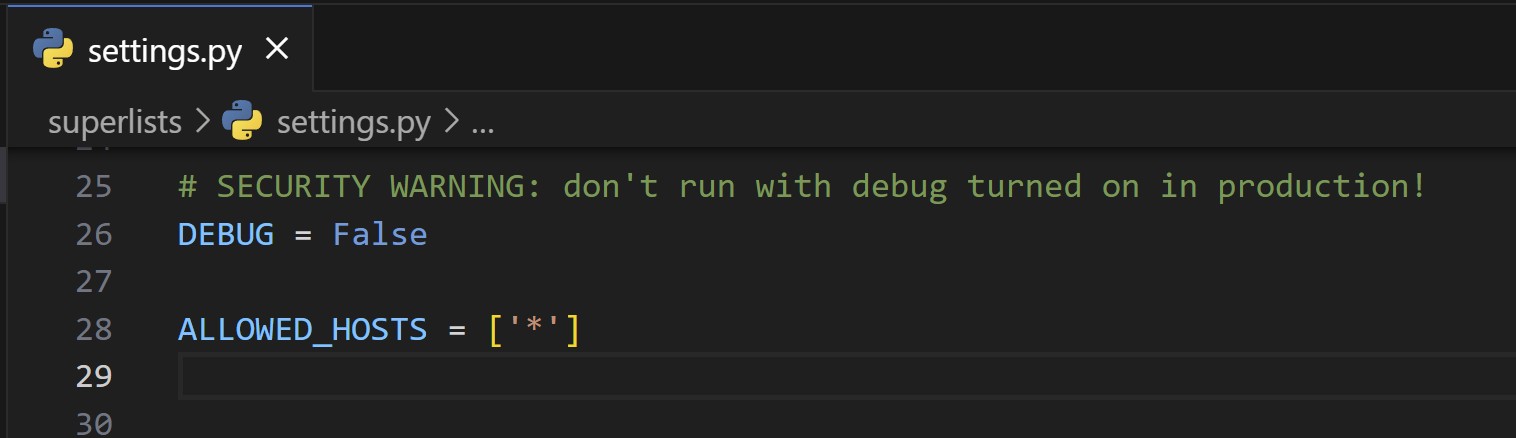
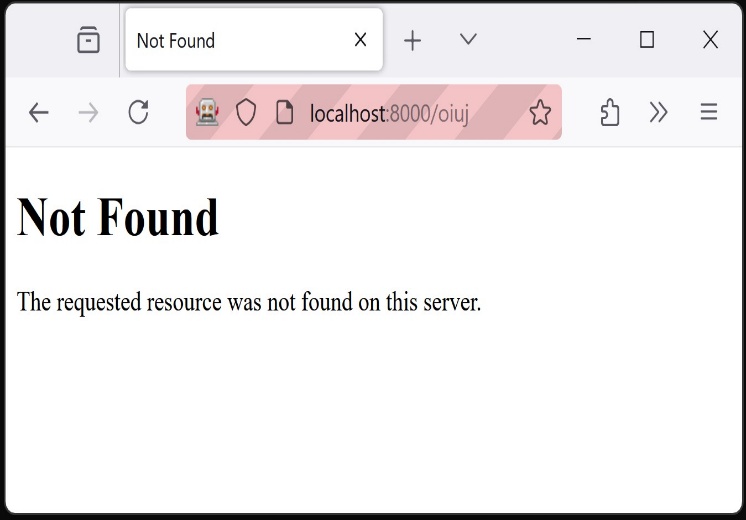
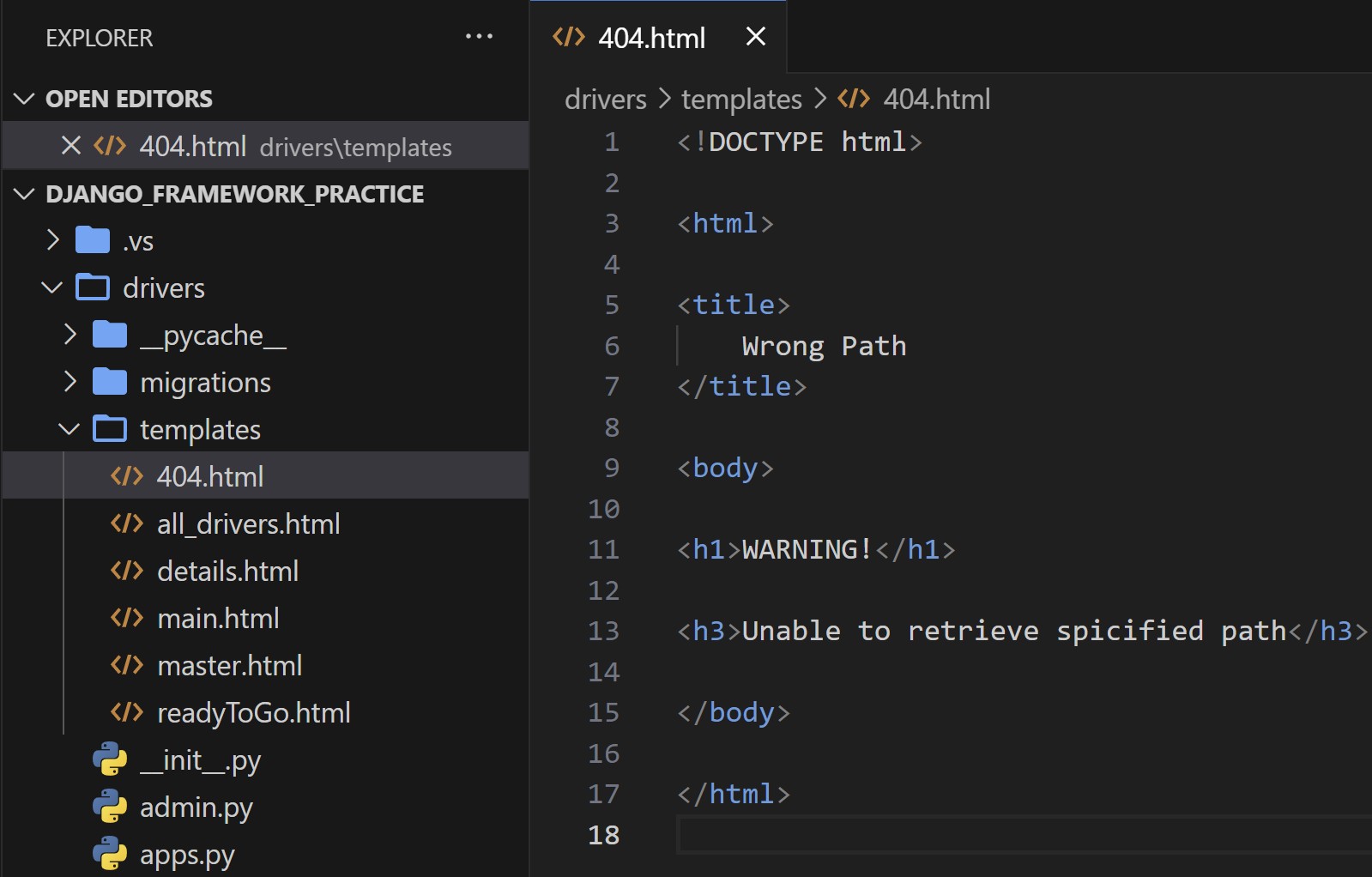
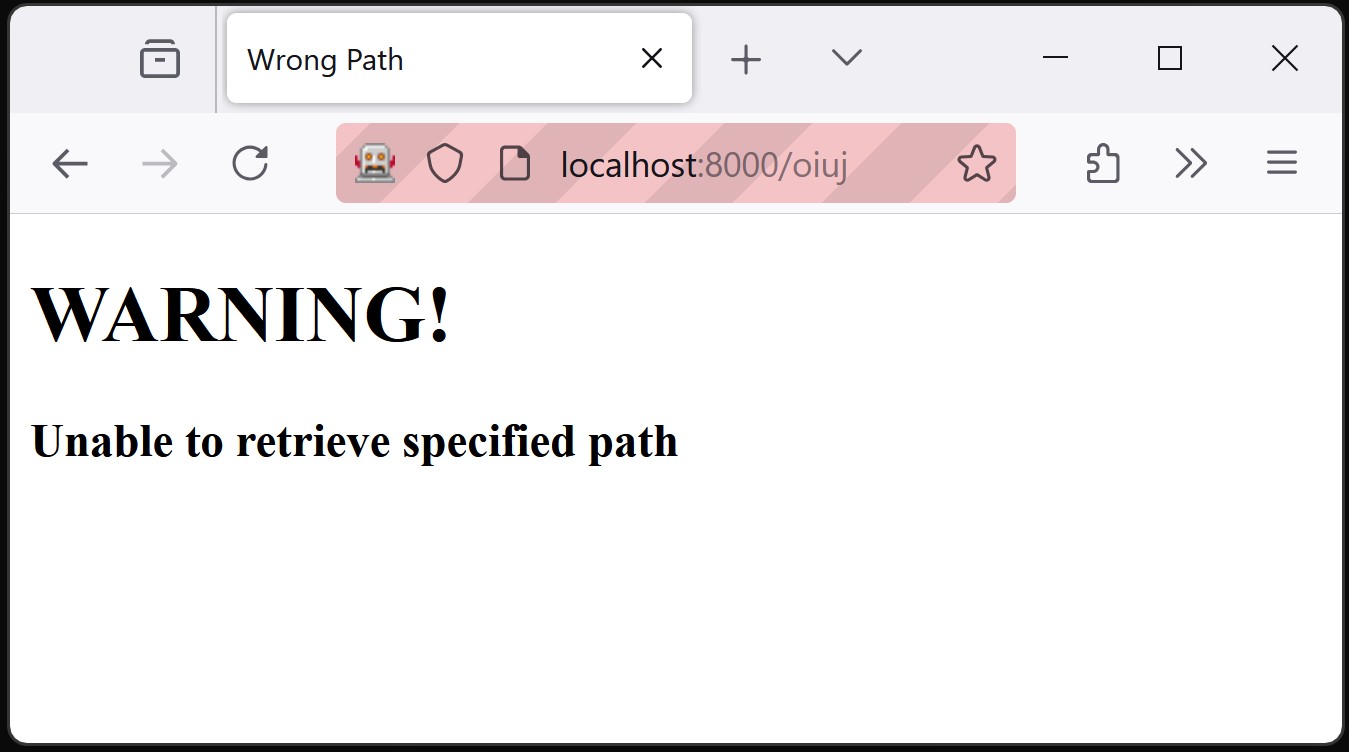
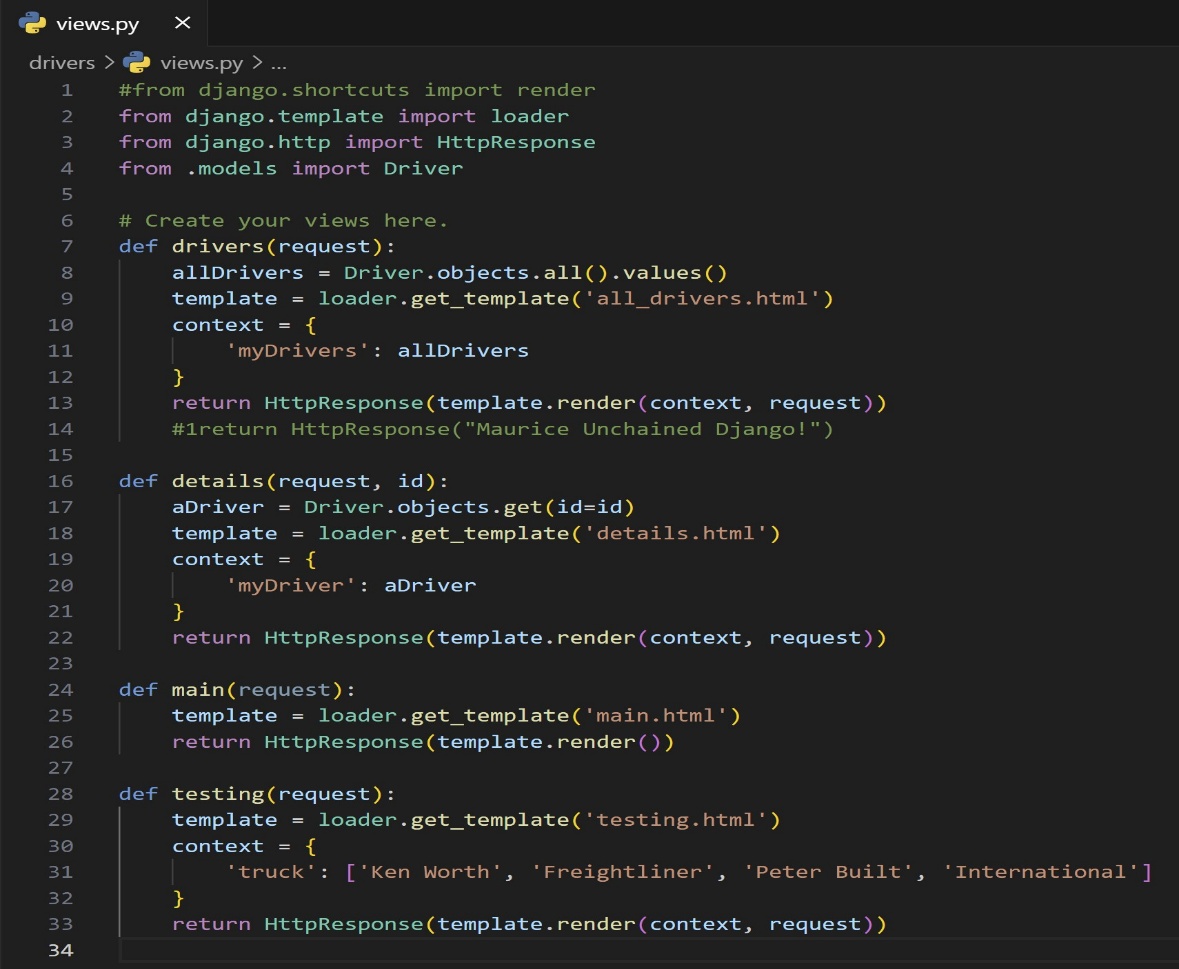
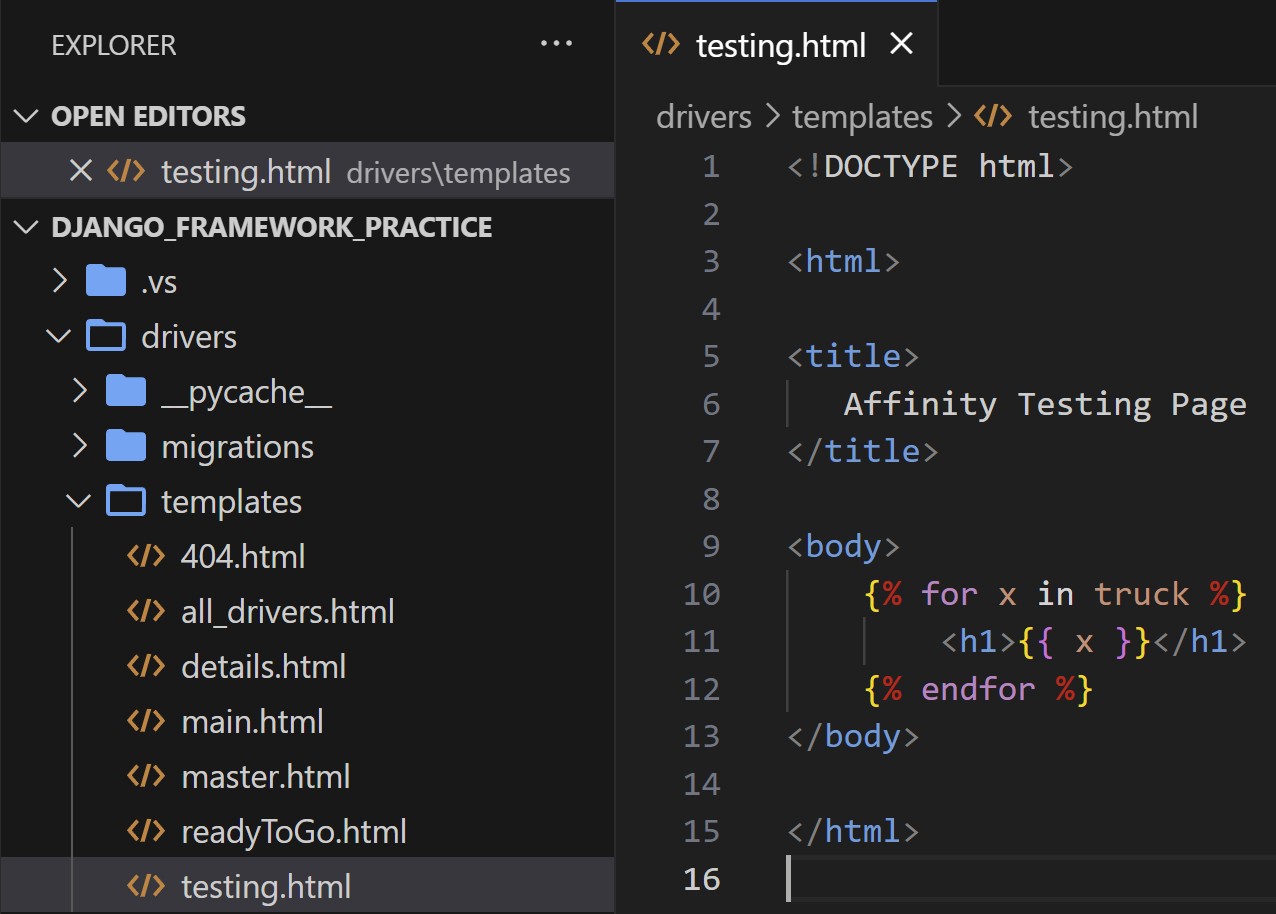
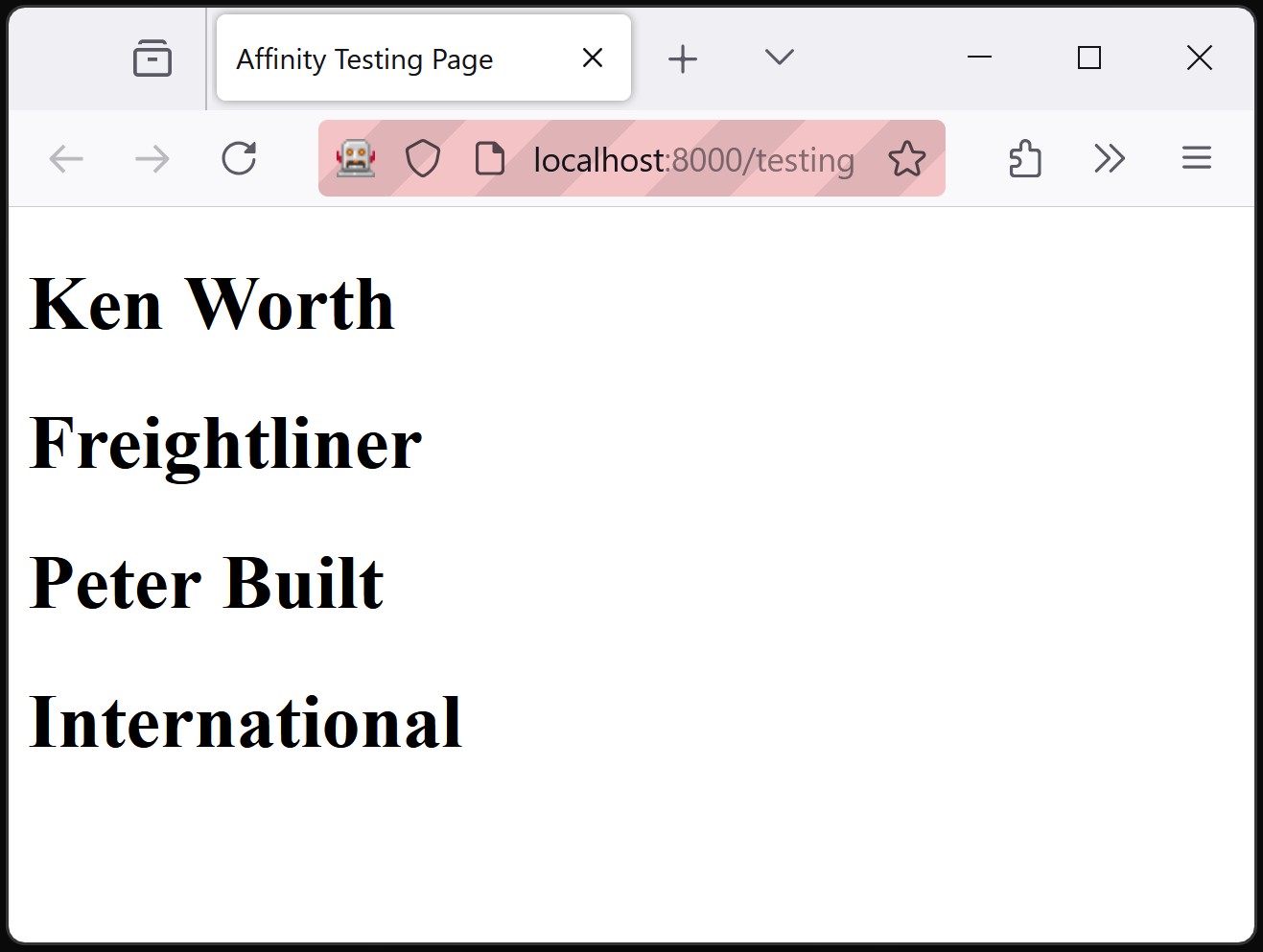
Got Goat Chapter 4 Displaying Database Data

We have created the model schema in the Driver app, the table in database, and entered records. We also discovered how to perform popular CRUD operations. In this chapter, we will create a template and view for displaying dynamic content on the screen.

1. Create HTML file
   1. Navigate to the drivers app/folder and create a new HTML file inside the templates folder similar to below.
   2. 
      1. Line 6: The double “%” instructs the view to perform programming logic inside the brackets. More specifically, the iterator will list the specified attributes of each record.
2. Update View
   1. We need to make the database model available to the template.
   2. 
      1. Line 4: We are importing the Driver model from driver app models folder
      2. Line 8: We are assigning the values of the model to a variable passed in response
      3. Line 11: The programming logic (myDriver) has to reference an object(allDrivers) for the iterator (for loop in html).
3. Display Records
   1. Run: python manage.py runserver
   2. 
4. Details HTML
   1. Navigate to the drivers app/folder and create a new HTML file called ‘details’ inside the templates folder similar to below.
   2. 
5. Link To Details
   1. Modify the all\_drivers html to link the first and last name to the details page
   2. 
   3. Navigate over to the terminal to see the changes
   4. 
   5. Clicking on the link will generate an error. We have not set up the routing to pass the data from the model to a view.
6. Create Detail View
   1. Navigate to the views.py file and add the following
   2. 
7. Add URL Routing
   1. The details URL has to link to the correct view. Open the URLs file and add the following.
   2. 
8. Display Drivers and link to details
   1. Run: python manage.py runserver
   2. 
      1. We can celebrate the above. The URL displays the page and contents. The driver’s table attribute values are displayed with the templated format.
   3. Pressing the link ‘Drivers’ will navigate the user back to the home page.
   4. 
      1. The first driver has a different color meaning that the link has been visited.
9. Master Template
   1. Django provides functionality to wrap similar content into blocks. Both the details HTML and all\_drivers HTML have similar structures. We can leverage this similarity into a single template.
   2. Create a new HTML file in the template folder called ‘master.html’. Insert the following.
   3. 
   4. Now that we have the master template. We can wrap the detail and all\_drivers HTMLs in programming language blocks. See below.
10. Modify the details HTML
    1. 
    2. The model data output and format are wrapped in blocks. Proceed to edit the all\_drivers HTML.
    3. 
    4. 
11. Main Index Page
    1. The main landing page will show an error because our current application does not have a main page.
    2. Create a main HTML file in the template folder
    3. 
    4. Add a main function to the view to load the main template.
    5. 
    6. Add path to the main page
    7. 
    8. Update the all\_drivers template to include a link to the main landing page. See below.
    9. 
    10. Run: python manage.py runserver
    11. 
        1. Success!!
12. 404 Error Page
    1. Navigating to an unspecified path will yield either a built-in Django error or an error page. To edit the error page, we have to set Debug = False in the settings.py file. See below.
    2. 
    3. If the user navigates to an unspecified path (localhost:8000/oiuj), they will see the below.
    4. 
    5. Django looks for a 404-html template when Django gets a 404 error. The above is displayed when it cannot find a template. Create a 404 html.
    6. 
    7. Navigating back to the open window, we can see the reflected changes on the page.
    8. 
13. Testing Page
    1. Following the same process described above, create a new function in the view called testing that will load a testing template. See below
    2. 
    3. Now create the testing template in templates folder under the driver app.
    4. 
    5. We added a view, created the template that the view calls, and now we add the routing to the URL pattern in the url.py file.
    6. 
       1. Success!!