MODULE 5 Enhancement Three: Databases

Author: Elena Ponomareva

In the CS-465: Full Stack Development course, we used the MEAN stack to build a web application for managing travel packages. MEAN stack uses MongoDB database where data stored in JSON documents organized into collections. In the project we used CRUD operations to retrieve data, update and delete it from the database.

For the Databases category, I expanded the complexity of a project and used MySQL database to store information about drivers, trucks, trailers and loads. I demonstrated the ability to use innovative techniques, skills, and tools in computing practices for the purpose of implementing computer solutions that deliver value and accomplish industry-specific goals. Moreover, I expanded the complexity of the datasets and linked loads to a specific driver. Incorporating more advanced MySQL concepts can significantly improve database management skills and allow us to build more efficient, scalable, and secure applications. This is essential for optimizing resources and reducing infrastructure costs while managing the trade-offs involved in design choices.

When improving the project I learned about get_form_kwargs(self) method which is a standard method in Django's class-based views responsible for returning a dictionary of keyword arguments (kwargs) that will be used to instantiate the form class associated with the view (Django, 2025). I used this method in LoadCreate() class in views.py (Pic 1), so I can connect the driver_id with the load on the front-end (Pic 2) and use this driver_id in "New Load" form (Pic 3 and Pic 4). Forms were created using Django forms and method {{ form.as_table }} to render a form in the template.

```python
class LoadCreate(CreateView):
    model = Load
    form_class = LoadForm
    success_url = reverse_lazy("erp_app:load_list")

    def get_form_kwargs(self):
        """Return the keyword arguments for instantiating the form."""
        kwargs = super().get_form_kwargs()
        kwargs["driver_id"] = self.request.GET.get("driver")
        return kwargs
```

Pic 1. src/erp_app/views.py

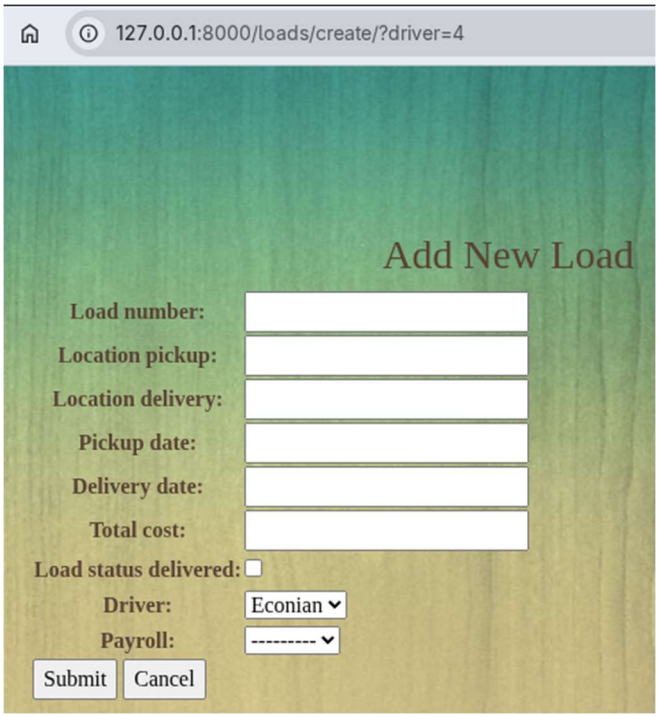
```
<td>
  <a href={% url 'erp_app:driver_update' driver.id %}>Update</a>
  <a href={% url 'erp_app:driver_delete' driver.id %}>Delete</a>
  <a href="{% url 'erp_app:load_create' %}?driver={{ driver.id }}">Load</a>
</td>
```

Pic 2. src/erp_app/templates/erp_app/driver_list.html

```python
class LoadForm(ModelForm):
    class Meta:
        model = Load
        fields = "__all__"

    def __init__(self, *args, **kwargs):
        driver_id = kwargs.pop("driver_id", None)
        super().__init__(*args, **kwargs)
        self.fields["driver"].initial = Driver.objects.get(id=driver_id)
```

Pic 3. src/erp_app/forms.py



Pic 4. Form "New Load" with automatically assigned driver.

References

Django. (2025). *Django Documentation.* https://docs.djangoproject.com/en/5.2/

SNHU Media. (2024, May 23). *CS 499 Milestone Two Code Review Overview Approach Assistance.* https://www.youtube.com/watch?v=229TiLwyipI&ab_channel=SNHUMedia