forms.py

```
from django import forms
from django.core import exceptions
from django.utils.translation import gettext_lazy as _

from oscar.apps.dashboard.catalogue.forms import ProductForm as CoreProductForm

from myapps.catalogue.models import Metal, Gemstone, ProductAttribute, Product


class ProductForm(CoreProductForm):
    class Meta:
        fields = [
            'title', 'upc', 'description', 'is_public', 'is_discountable', 'structure', 'slug', 'meta_title',
            'meta_description', 'metal', 'primary_gemstone']

    def __init__(self, product_class, metal, data=None, parent=None, *args, **kwargs):
        self.set_initial(product_class, metal, parent, kwargs)
        super().__init__(data, *args, **kwargs)
        if parent:
            self.instance.parent = parent
            # We need to set the correct product structures explicitly to pass
            # attribute validation and child product validation. Note that
            # those changes are not persisted.
            self.instance.structure = Product.CHILD
            self.instance.parent.structure = Product.PARENT

            self.delete_non_child_fields()
        else:
            # Only set product class and metal type for non-child products
            self.instance.product_class = product_class
            self.instance.metal = metal
        self.add_attribute_fields(product_class, metal, self.instance.is_parent)

        if 'slug' in self.fields:
            self.fields['slug'].required = False
            self.fields['slug'].help_text = _('Leave blank to generate from product title')
        if 'title' in self.fields:
            self.fields['title'].widget = forms.TextInput(
                attrs={'autocomplete': 'off'})

    def set_initial(self, product_class, metal, parent, kwargs):
        """
        Set initial data for the form. Sets the correct product structure
        and fetches initial values for the dynamically constructed attribute
        fields.
        """
        if 'initial' not in kwargs:
            kwargs['initial'] = {}
        self.set_initial_attribute_values(product_class, metal, kwargs)
        if parent:
            kwargs['initial']['structure'] = Product.CHILD

    def set_initial_attribute_values(self, product_class, metal, kwargs):
        """
        Update the kwargs['initial'] value to have the initial values based on
        the product instance's attributes
        """
        super().set_initial_attribute_values(product_class, kwargs)
        instance = kwargs.get('instance')
        if instance is None:
            return
        for attribute in metal.product_attribute.all():
            try:
                value = instance.attribute_values.get(
                    attribute=attribute).value
            except exceptions.ObjectDoesNotExist:
                pass
            else:
                kwargs['initial']['attr_%s' % attribute.code] = value

    def add_attribute_fields(self, product_class, metal, is_parent=False):
        """
        For each attribute specified by the product class, metal type,
        this method dynamically adds form fields to the product form.
        """
        super().add_attribute_fields(product_class, is_parent=False)

        for attribute in metal.product_attribute.all():
            field = self.get_attribute_field(attribute)
```

```
if field:
    self.fields['attr_%s' % attribute.code].required = False
```