ChatGPT

Project Structure:

```
myproject/
├── accounting/  # or journals/ if a separate app
│   ├── admin.py
│   ├── forms.py
│   ├── views.py
│   ├── urls.py
│   └── templates/
│       └── journals/
│           └── general_journal_form.html
└── tests/
    └── test_general_journal.py
```

Below are the contents of each file:

```python
# accounting/admin.py
from django.contrib import admin
from .models import Journal, JournalLine, JournalType, VoucherModeConfig,
VoucherModeDefault, VoucherUDFConfig

class JournalLineInline(admin.TabularInline):
    model = JournalLine
    extra = 0

@admin.register(Journal)
class JournalAdmin(admin.ModelAdmin):
    inlines = [JournalLineInline]
    # Optional: configure list display, filters, etc., if needed.
    # list_display = ('journal_number', 'journal_type', 'journal_date',
'status', ...)
    # list_filter = ('journal_type', 'status')

# Register other models in the admin
admin.site.register(JournalType)
admin.site.register(VoucherModeConfig)
admin.site.register(VoucherModeDefault)
admin.site.register(VoucherUDFConfig)
# (JournalLine is managed via inline, but you can register it separately if
desired)
```

```python
# accounting/forms.py
from django import forms
from django.forms import inlineformset_factory, modelformset_factory
from .models import Journal, JournalLine, VoucherModeConfig
from .utils import get_active_currency_choices
from .forms_factory import build_form, build_formset  # using the dynamic
FormBuilder utilities
from .forms_mixin import BootstrapFormMixin

# (Other form classes above ...)

# Dynamic Voucher entry form builders
def get_voucher_forms(config: VoucherModeConfig, user):
    """
    Build header form and line formset classes for a given voucher
configuration.
    Returns a tuple (HeaderFormClass, LineFormSetClass).
    """
    org = getattr(user, 'get_active_organization', None) and
user.get_active_organization() or getattr(user, 'organization', None)
    ui_schema = config.resolve_ui()
    # Optionally inject currency fields if multiple currencies allowed
    if config.allow_multiple_currencies:
        # Ensure currency_code and exchange_rate appear in header schema if not
already present
        if 'currency_code' not in ui_schema['header']:
            ui_schema['header']['currency_code'] = {
                "type": "select",
                "label": "Currency",
                "required": True,
                "choices": get_active_currency_choices()
            }
        if 'exchange_rate' not in ui_schema['header']:
            ui_schema['header']['exchange_rate'] = {
                "type": "decimal",
                "label": "Exchange Rate",
                "required": True
            }
    # Build ModelForm for Journal (header) and FormSet for JournalLine (lines)
    user_perms = {
        # Example: field-level permission flags, if applicable. For simplicity,
assume full edit:
        f"can_edit_{field}": True for field in ui_schema['header'].keys()
    }
    HeaderFormClass = build_form(ui_schema["header"], organization=org,
user_perms=user_perms, prefix="hdr", model=Journal)
    LineFormSetClass = build_formset(ui_schema["lines"], organization=org,
```

```
user_perms=user_perms, model=JournalLine, extra=0, prefix="lines")
    return HeaderFormClass, LineFormSetClass

# If any static ModelForms for Journal or JournalLine are needed (e.g.,
fallback), they could be defined here.
# For example, a basic JournalLineForm in case dynamic schema is not available:
class JournalLineForm(BootstrapFormMixin, forms.ModelForm):
    class Meta:
        model = JournalLine
        fields = ['account', 'description', 'debit_amount', 'credit_amount',
'department', 'project', 'cost_center', 'tax_code']
```

```
# accounting/views.py
from django.shortcuts import render, redirect
from django.http import JsonResponse, Http404
from django.views import View
from django.contrib.auth.mixins import LoginRequiredMixin
from django.core.exceptions import ValidationError
from .models import Journal, JournalLine, JournalType, VoucherModeConfig,
VoucherModeDefault
from .forms import get_voucher_forms, JournalLineForm
from .services import create_voucher
# Assume hooks.py exists with before_save_voucher & after_save_voucher; import
if available
try:
    from . import hooks
except ImportError:
    hooks = None

class GeneralJournalCreateView(LoginRequiredMixin, View):
    """
    View to render and handle the General Journal "New Entry" form.
    """
    def _get_config(self, request):
        # Fetch the VoucherModeConfig for General Journal (code "GJ")
        org = request.user.get_active_organization() if hasattr(request.user,
'get_active_organization') else getattr(request.user, 'organization', None)
        try:
            journal_type = JournalType.objects.get(code="GJ", organization=org)
        except JournalType.DoesNotExist:
            raise Http404("General Journal type not found")
        config =
VoucherModeConfig.objects.filter(journal_type=journal_type).first()
        if not config:
            raise Http404("General Journal configuration not found")
        return config
```

```
    def get(self, request, *args, **kwargs):
        config = self._get_config(request)
        HeaderFormClass, LineFormSetClass = get_voucher_forms(config,
request.user)
        # Prepare initial data for header form (defaults)
        header_initial = {}
        # Pre-fill currency and period defaults
        header_initial['journal_date'] = None  # default to blank (or use
timezone.now().date() for current date)
        header_initial['currency_code'] = getattr(config, 'default_currency',
None)

# If there's a current open period, could prefill (period not on form by
default, it's auto-determined on save)
        header_initial['reference'] = ""  # no default unless provided
        header_initial['description'] = ""  # could use
default_narration_template if desired
        if hasattr(config, 'default_narration_template') and
config.default_narration_template:
            header_initial['description'] = config.default_narration_template

        # Prepare initial data for line forms from VoucherModeDefault
        default_lines =
list(VoucherModeDefault.objects.filter(config=config).order_by('display_order'))
        initial_lines = []
        for default in default_lines:
            line_data = {}
            if default.account:
                line_data['account'] = default.account.pk

# If default amount is specified, assign to debit or credit based on flags
            if default.default_amount:
                if default.default_debit:
                    line_data['debit_amount'] = default.default_amount
                if default.default_credit:
                    line_data['credit_amount'] = default.default_amount
            if default.default_description:
                line_data['description'] = default.default_description
            # If defaults for dimensions or tax are provided (stored as IDs)
            if default.default_department:
                line_data['department'] = default.default_department
            if default.default_project:
                line_data['project'] = default.default_project
            if default.default_cost_center:
                line_data['cost_center'] = default.default_cost_center
            if default.default_tax_code_id:
                line_data['tax_code'] = default.default_tax_code_id
```

```
            initial_lines.append(line_data)
        # Instantiate forms
        header_form = HeaderFormClass(initial=header_initial)
        # For lines, use modelformset (LineFormSetClass is a ModelFormSet class)
        if initial_lines:
            line_formset = LineFormSetClass(queryset=JournalLine.objects.none(),
initial=initial_lines, prefix='lines')
        else:
            # If no defaults, provide at least one blank line
            line_formset = LineFormSetClass(queryset=JournalLine.objects.none(),
initial=[], prefix='lines', extra=1)
        return render(request, 'journals/general_journal_form.html', {
            'header_form': header_form,
            'line_formset': line_formset,
            'voucher_config': config
        })

    def post(self, request, *args, **kwargs):
        config = self._get_config(request)
        HeaderFormClass, LineFormSetClass = get_voucher_forms(config,
request.user)
        header_form = HeaderFormClass(request.POST)
        line_formset = LineFormSetClass(request.POST,
queryset=JournalLine.objects.none(), prefix='lines')
        if not (header_form.is_valid() and line_formset.is_valid()):
            # On validation errors, re-render the form with errors
            return render(request, 'journals/general_journal_form.html', {
                'header_form': header_form,
                'line_formset': line_formset,
                'voucher_config': config
            })
        # Both forms valid, extract data
        header_data = header_form.cleaned_data
        lines_data = [form.cleaned_data for form in line_formset.forms if not
form.cleaned_data.get('DELETE', False)]
        udf_header = {}  # Assuming header UDF fields (if any) are included in
header_form.cleaned_data or handled similarly
        udf_lines = []
        # Handle UDF fields from POST (they might come as "udf_fieldname_index")
        # We gather any fields in POST that start with "udf_" and organize per
line index.
        for key, value in request.POST.items():
            if key.startswith("udf_"):
                # Format expected: "udf_<field_name>_<index>"
                try:
                    _, field, idx = key.split('_', 2)
                except ValueError:
                    continue
```

```
                # Ensure the list is long enough
                idx = int(idx)
                # Expand udf_lines list to hold index
                while len(udf_lines) <= idx:
                    udf_lines.append({})
                udf_lines[idx][field] = value

        # Attempt to save voucher via service
        try:
            # Optional: call before_save hook
            if hooks and hasattr(hooks, 'before_save_voucher'):
                hooks.before_save_voucher(None, request.user)
# If hook expects a voucher object, could pass header_data
            journal = create_voucher(request.user, config.config_id,
header_data, lines_data, udf_header=udf_header, udf_lines=udf_lines)
            # Call after_save hook
            if hooks and hasattr(hooks, 'after_save_voucher'):
                hooks.after_save_voucher(journal, request.user)
        except ValidationError as e:
            # If create_voucher raised validation error (e.g., imbalance),
handle accordingly
            error_msg = "; ".join(e.messages) if hasattr(e, 'messages') else
str(e)
            if request.headers.get('HX-Request') or request.headers.get('Hx-
Request'):
                # Return JSON errors for HTMX requests
                return JsonResponse({"errors": error_msg}, status=400)
            # For normal form submission, add as non-field error and re-render
form
            header_form.add_error(None, error_msg)
            return render(request, 'journals/general_journal_form.html', {
                'header_form': header_form,
                'line_formset': line_formset,
                'voucher_config': config
            })
        # On success, redirect to the new journal's detail page
        return redirect('accounting:journal_detail', pk=journal.journal_id)


class LineRowHXView(LoginRequiredMixin, View):
    """
    HTMX view to return a blank journal line row (table row) form fragment.
    """
    def get(self, request, *args, **kwargs):
        voucher_type = request.GET.get("type")  # optional voucher code (e.g.,
"GJ")
        index = request.GET.get("index", "0")
        # Determine which form to use for a new line
```

```
        if voucher_type:
            # Load schema by voucher type code and build a one-off formset
            try:
                org = request.user.get_active_organization() if
hasattr(request.user, 'get_active_organization') else request.user.organization
            except Exception:
                org = None
            try:
                config = VoucherModeConfig.objects.get(code=voucher_type,
organization=org)
            except VoucherModeConfig.DoesNotExist:
                raise Http404("Voucher config not found")
            HeaderFormClass, LineFormSetClass = get_voucher_forms(config,
request.user)
            LineFormSet = LineFormSetClass  # already a FormSet class
        else:
            # Fallback: use General Journal config by default
            config = self._get_default_config(request) if hasattr(self,
'_get_default_config') else None
            if not config:
                # If no default config method, just reuse
GeneralJournalCreateView logic
                try:
                    journal_type = JournalType.objects.get(code="GJ",
organization=request.user.organization)
                    config =
VoucherModeConfig.objects.filter(journal_type=journal_type).first()
                except Exception:
                    config = None
            HeaderFormClass, LineFormSetClass = get_voucher_forms(config,
request.user) if config else (None, None)
            LineFormSet = LineFormSetClass
        if LineFormSet is None:
            # If still not determined, use a static fallback form
            FormClass = JournalLineForm
            LineFormSet = modelformset_factory(JournalLine, form=FormClass,
extra=1)
        # Get an empty form (unbound) for a single new line
        empty_form = LineFormSet(queryset=JournalLine.objects.none()).empty_form
        empty_form.prefix = f"lines-{index}"
        return render(request, "accounting/partials/journal_line_form.html", {
            "form": empty_form,
            "form_index": index,
            "udf_configs": list(config.udf_configs.filter(scope='line',
is_active=True)) if config else [],
            "show_dimensions": getattr(config, "show_dimensions", False),
            "show_tax_details": getattr(config, "show_tax_details", False),
        })
```

```
class ValidateVoucherHXView(LoginRequiredMixin, View):
    """
    HTMX view to validate the entire voucher (header + lines) and return JSON
with any errors.
    """
    def post(self, request, *args, **kwargs):
        # Identify the voucher config (pass config_id in POST or infer from
context)
        config_id = request.POST.get('config_id')
        config = None
        if config_id:
            try:
                config = VoucherModeConfig.objects.get(pk=config_id,
organization=request.user.get_active_organization())
            except VoucherModeConfig.DoesNotExist:
                pass
        if not config:
            try:
                # Fallback to General Journal config by code
                journal_type = JournalType.objects.get(code="GJ",
organization=request.user.get_active_organization())
                config =
VoucherModeConfig.objects.filter(journal_type=journal_type).first()
            except JournalType.DoesNotExist:
                config = None
        if not config:
            return JsonResponse({"valid": False, "errors": {"__all__":
["Configuration not found"]}}, status=400)
        HeaderFormClass, LineFormSetClass = get_voucher_forms(config,
request.user)
        header_form = HeaderFormClass(request.POST)
        line_formset = LineFormSetClass(request.POST,
queryset=JournalLine.objects.none(), prefix='lines')
        valid = header_form.is_valid() and line_formset.is_valid()
        errors = {
            "header": header_form.errors,
            "lines": line_formset.errors,
        }
        return JsonResponse({"valid": valid, "errors": errors})
```

```
# accounting/urls.py
from django.urls import path
from .views import GeneralJournalCreateView, LineRowHXView,
ValidateVoucherHXView, HTMXJournalLineFormView
```

```
app_name = "journals"  # using a separate namespace for journal entries
urlpatterns = [
    # General Journal entry page
    path('general/new/', GeneralJournalCreateView.as_view(),
name='general_journal_new'),
    # HTMX endpoints for dynamic interactions
    path('general/new/line/', LineRowHXView.as_view(),
name='general_journal_add_line'),
    path('general/new/validate/', ValidateVoucherHXView.as_view(),
name='general_journal_validate'),
    path('general/new/udf/', HTMXJournalLineFormView.as_view(),
name='general_journal_udf_form'),
]
```

```
<!-- accounting/templates/journals/general_journal_form.html -->
{% extends "dason/base.html" %}
{% load static %}
{% block content %}
<div class="container-fluid">
  <h1 class="mb-4">New General Journal Entry</h1>
  <!-- Alert for non-field errors -->
  {% if header_form.non_field_errors %}
    <div class="alert alert-danger">{{ header_form.non_field_errors|join:"; " }}
</div>
  {% endif %}
  <form method="post" action="">
    {% csrf_token %}
    <div class="card mb-3 p-3">
      <h5 class="card-title">Journal Details</h5>
      {{ header_form.as_p }}
    </div>
    <div class="card p-3">
      <h5 class="card-title">Journal Lines</h5>
      <div class="table-responsive">
        <table class="table table-bordered">
          <thead>
            <tr>
              <th>Account</th>
              <th>Description</th>
              <th>Debit</th>
              <th>Credit</th>
              {% if voucher_config.show_dimensions %}
                <th>Department</th>
                <th>Project</th>
                <th>Cost Center</th>
              {% endif %}
```

```
              {% if voucher_config.show_tax_details %}
                <th>Tax Code</th>
              {% endif %}
              {% for udf in voucher_config.udf_configs.all|default:None %}
                {% if udf.scope == 'line' %}
                  <th>{{ udf.display_name }}</th>
                {% endif %}
              {% endfor %}
              <th>Actions</th>
            </tr>
          </thead>
          <tbody id="lines">
            {% for form in line_formset.forms %}
              {% include "accounting/partials/journal_line_form.html" with
form=form form_index=forloop.counter0 %}
            {% endfor %}
          </tbody>
          <tfoot>
            <tr>
              <td colspan="2" class="text-end"><strong>Totals:</strong></td>
              <td class="fw-bold"><span id="total-debit">0.00</span></td>
              <td class="fw-bold"><span id="total-credit">0.00</span></td>
              {% if voucher_config.show_dimensions %}<td colspan="3"></td>{%
endif %}
              {% if voucher_config.show_tax_details %}<td></td>{% endif %}
              {% if voucher_config.udf_configs.filter(scope='line').exists %}
                  <td
colspan="{{ voucher_config.udf_configs.filter(scope='line').count }}"></td>
              {% endif %}
              <td></td>
            </tr>
          </tfoot>
        </table>
        <button type="button" id="add-line-btn" class="btn btn-primary btn-sm"
                hx-get="{% url 'journals:general_journal_add_line' %}?
index={{ line_formset.total_form_count }}"
                hx-target="#lines" hx-swap="afterend">
          Add Line
        </button>
      </div>
    </div>
    <div class="mt-3">
      <button type="submit" class="btn btn-success">Save Journal</button>
      <a href="{% url 'accounting:journal_list' %}" class="btn btn-
secondary">Cancel</a>
    </div>
  </form>
</div>
```

```
{% endblock %}

{% block extra_script %}
<script>
  // Using HTMX to validate totals on debit/credit field changes
  document.querySelectorAll('input[name$="debit_amount"],
input[name$="credit_amount"]').forEach(function(elem) {
    elem.setAttribute('hx-post', "{% url 'journals:general_journal_validate'
%}");
    elem.setAttribute('hx-target', 'this.closest("tr")');
    elem.setAttribute('hx-trigger', 'blur change');
    elem.setAttribute('hx-swap', 'outerHTML');  // could highlight row or field
errors
  });
  // Listen for formset changes to update total debit/credit display
  document.getElementById('lines').addEventListener('input', function() {
    let totalDebit = 0.0, totalCredit = 0.0;
    document.querySelectorAll('input[name$="debit_amount"]').forEach(el => {
      totalDebit += parseFloat(el.value) || 0;
    });
    document.querySelectorAll('input[name$="credit_amount"]').forEach(el => {
      totalCredit += parseFloat(el.value) || 0;
    });
    document.getElementById('total-debit').innerText = totalDebit.toFixed(2);
    document.getElementById('total-credit').innerText = totalCredit.toFixed(2);
  });
</script>
{% endblock %}
```

```
# tests/test_general_journal.py
import pytest
from django.urls import reverse
from accounting.models import Journal, JournalLine, JournalType,
VoucherModeConfig, ChartOfAccount

@pytest.mark.django_db
def test_defaults_load(client, django_user_model):
    # Ensure default config and defaults exist (via fixtures or setup)
    user = django_user_model.objects.create_user(username='user1',
password='pass')
    # Assign organization and create JournalType and VoucherModeConfig for "GJ"
    # (Assume user.organization is set via a fixture or a relation)
    org = getattr(user, 'organization', None) or getattr(user,
'get_active_organization', lambda: None)()
    journal_type = JournalType.objects.create(organization=org, code="GJ",
name="General Journal")
```

```
    config = VoucherModeConfig.objects.filter(journal_type=journal_type).first()
    assert config is not None, "VoucherModeConfig for GJ should exist"
    # Create a default account and a VoucherModeDefault line
    account = ChartOfAccount.objects.filter(organization=org).first()
    if not account:
        account = ChartOfAccount.objects.create(organization=org,
account_name="Cash", account_code="1000", is_active=True)

VoucherModeConfig.objects.filter(journal_type=journal_type).update(default_currency="USD")
    # Create two default line configs (one debit, one credit) for balance
    VoucherModeDefault.objects.create(config=config, account=account,
default_debit=True, default_amount=100, display_order=1)
    VoucherModeDefault.objects.create(config=config, account=account,
default_credit=True, default_amount=100, display_order=2)
    client.login(username='user1', password='pass')
    url = reverse('journals:general_journal_new')
    response = client.get(url)
    assert response.status_code == 200
    content = response.content.decode()
    # Check that default currency and two line forms are present
    assert 'value="USD"' in content or 'USD' in content, "Default currency
should be pre-selected"
    assert content.count('name="lines-') >= 2, "Should render default number of
line forms"

@pytest.mark.django_db
def test_post_balanced_lines_success(client, django_user_model):
    # Setup user, config, accounts
    user = django_user_model.objects.create_user(username='user2',
password='pass')
    org = getattr(user, 'organization', None) or getattr(user,
'get_active_organization', lambda: None)()
    jt = JournalType.objects.create(organization=org, code="GJ", name="General
Journal")
    config = VoucherModeConfig.objects.filter(journal_type=jt).first()
    account1 = ChartOfAccount.objects.create(organization=org,
account_code="2000", account_name="Sales", is_active=True)
    account2 = ChartOfAccount.objects.create(organization=org,
account_code="3000", account_name="Expenses", is_active=True)
    client.login(username='user2', password='pass')
    url = reverse('journals:general_journal_new')
    # Prepare balanced lines data
    post_data = {
        'journal_date': '2025-01-01',
        'reference': 'TestRef',
        'description': 'Test Entry',
        'currency_code': config.default_currency or 'USD',
        'exchange_rate': '1',
```

```
        # Management form data for formset:
        'lines-TOTAL_FORMS': '2',
        'lines-INITIAL_FORMS': '0',
        'lines-MIN_NUM_FORMS': '0',
        'lines-MAX_NUM_FORMS': '1000',
        # Line 0 (debit)
        'lines-0-account': str(account1.pk),
        'lines-0-description': 'Debit line',
        'lines-0-debit_amount': '500.00',
        'lines-0-credit_amount': '',
        # Line 1 (credit)
        'lines-1-account': str(account2.pk),
        'lines-1-description': 'Credit line',
        'lines-1-debit_amount': '',
        'lines-1-credit_amount': '500.00',
    }
    response = client.post(url, post_data, follow=False)
    # Should redirect on success
    assert response.status_code == 302, "Balanced voucher should redirect after
save"
    # Verify Journal created in database with correct totals
    journal = Journal.objects.filter(journal_type=jt).latest('journal_id')
    assert journal.total_debit == journal.total_credit == 500.00
    assert JournalLine.objects.filter(journal=journal).count() == 2

@pytest.mark.django_db
def test_post_unbalanced_returns_json_error(client, django_user_model):
    user = django_user_model.objects.create_user(username='user3',
password='pass')
    org = getattr(user, 'organization', None) or getattr(user,
'get_active_organization', lambda: None)()
    jt = JournalType.objects.create(organization=org, code="GJ", name="General
Journal")
    config = VoucherModeConfig.objects.filter(journal_type=jt).first()
    acct = ChartOfAccount.objects.create(organization=org, account_code="4000",
account_name="Misc", is_active=True)
    client.login(username='user3', password='pass')
    url = reverse('journals:general_journal_new')
    # One-sided entry (debit without matching credit)
    data = {
        'journal_date': '2025-02-01',
        'currency_code': config.default_currency or 'USD',
        'exchange_rate': '1',
        'reference': '',
        'description': 'Unbalanced Entry',
        'lines-TOTAL_FORMS': '1',
        'lines-INITIAL_FORMS': '0',
        'lines-MIN_NUM_FORMS': '0',
```

```
        'lines-MAX_NUM_FORMS': '1000',
        'lines-0-account': str(acct.pk),
        'lines-0-description': 'Only debit',
        'lines-0-debit_amount': '250.00',
        'lines-0-credit_amount': '',
    }
    # Simulate an HTMX request (HX-Request header)
    response = client.post(url, data, HTTP_HX_Request='true')
    assert response.status_code == 400
    json_data = response.json()
    assert not json_data.get('success', True)
    # The error message should indicate imbalance
    error_text = json_data.get('errors') or ''
    assert "Debit and Credit totals must match" in error_text or "must match" in
error_text
```
