**Asset Management Module - Installation Guide**

**📋 Overview**

The Asset Management module adds comprehensive tracking for:

* Fixed assets (IT equipment, machinery, tools)
* Serialized items
* Asset assignments to users/departments
* Warranty management
* Maintenance scheduling
* Depreciation calculation (Straight-line & Declining Balance)
* Software license tracking

**🚀 Installation Steps**

**1. Create the Assets App Directory Structure**

mkdir -p assets/management/commands

mkdir -p assets/migrations

touch assets/\_\_init\_\_.py (ni assets/\_\_init\_\_.py)

touch assets/apps.py (ni assets/apps.py)

touch assets/models.py (ni assets/models.py)

touch assets/admin.py (ni assets/admin.py)

touch assets/signals.py (ni assets/signals.py)

touch assets/tests.py (ni assets/tests.py)

touch assets/views.py (ni assets/views.py)

touch assets/management/\_\_init\_\_.py (ni assets/management/\_\_init\_\_.py)

touch assets/management/commands/\_\_init\_\_.py (ni assets/management/commands/\_\_init\_\_.py)

For windows, instead of “touch” use “ni” means New-Item to create files.

**2. Copy the Code Files**

Copy the following files from the artifacts:

1. **assets/models.py** - Core models (Asset, AssetAssignmentHistory, AssetMaintenanceRecord, AssetDepreciationSchedule)
2. **assets/admin.py** - Django admin configuration
3. **assets/apps.py** - App configuration
4. **assets/signals.py** - Auto-create assets from GRN
5. **assets/management/commands/\*.py** - Management commands

**3. Update Django Settings**

Edit pdl\_ims\_backend/settings.py:

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

# Third-party apps

'rest\_framework',

'corsheaders',

'django\_filters',

# Our apps

'users.apps.UsersConfig',

'core.apps.CoreConfig',

'inventory.apps.InventoryConfig',

'procurement.apps.ProcurementConfig',

'production.apps.ProductionConfig',

'assets.apps.AssetsConfig', # ← ADD THIS

'reports.apps.ReportsConfig',

]

**4. Install Additional Dependencies**

pip install python-dateutil

Update requirements.txt:

# ... existing packages ...

python-dateutil==2.8.2

**5. Create and Run Migrations**

# Create migration files

python manage.py makemigrations assets

# Review the migration file

# It should create 4 tables:

# - assets

# - asset\_assignment\_history

# - asset\_maintenance\_records

# - asset\_depreciation\_schedule

# Apply migrations

python manage.py migrate assets

**6. Verify Installation**

# Check if models are registered

python manage.py shell

from assets.models import Asset, AssetAssignmentHistory, AssetMaintenanceRecord

from django.contrib.contenttypes.models import ContentType

# Should return 4 content types

ContentType.objects.filter(app\_label='assets').count()

# Expected: 4

# Exit shell

exit()

**7. Test Admin Interface**

# Start development server

python manage.py runserver

Navigate to: http://127.0.0.1:8000/admin/assets/

You should see:

* Assets
* Asset assignment histories
* Asset maintenance records
* Asset depreciation schedules

**🧪 Testing the Module**

**Test 1: Create an Asset Manually**

python manage.py shell

from assets.models import Asset

from inventory.models import Item

from core.models import Location

# Get a serialized item (or create one)

item = Item.objects.filter(is\_serialized=True).first()

if not item:

# Create a test item

from inventory.models import ItemCategory, UnitOfMeasurement

category = ItemCategory.objects.first()

uom = UnitOfMeasurement.objects.first()

item = Item.objects.create(

sku='IT-LAPTOP-001',

item\_name='Dell Latitude 5520',

category=category,

default\_uom=uom,

is\_serialized=True,

is\_asset=True

)

# Create an asset

asset = Asset.objects.create(

item=item,

serial\_no='SN123456789',

model='Latitude 5520',

manufacturer='Dell',

purchase\_date='2025-01-15',

purchase\_price=85000,

depreciation\_method='STRAIGHT\_LINE',

useful\_life\_years=5,

current\_status='IN\_STOCK'

)

print(f"✓ Created asset: {asset.asset\_tag}")

print(f" Book Value: {asset.current\_book\_value}")

print(f" Depreciation: {asset.accumulated\_depreciation}")

**Test 2: Assign Asset to User**

from users.models import User

# Get or create a test user

user = User.objects.first()

# Assign asset

asset.assign\_to\_user(user=user, assigned\_by=user)

print(f"✓ Assigned {asset.asset\_tag} to {user.full\_name}")

print(f" Status: {asset.current\_status}")

# Check assignment history

history = asset.assignment\_history.first()

print(f" History: {history}")

**Test 3: Auto-Create Asset from GRN**

from procurement.models import GRNLine

# Find a GRN line with serialized item

grn\_line = GRNLine.objects.filter(

item\_\_is\_serialized=True,

serial\_no\_\_isnull=False

).first()

if grn\_line:

# Update GRN status to trigger asset creation

grn = grn\_line.grn

grn.status = 'ACCEPTED'

grn.save()

# Check if asset was created

asset = Asset.objects.filter(serial\_no=grn\_line.serial\_no).first()

if asset:

print(f"✓ Auto-created asset: {asset.asset\_tag}")

else:

print("⚠ Asset not created - check signals")

**Test 4: Generate Depreciation Schedule**

# Generate depreciation for all assets

python manage.py generate\_depreciation

# Generate for specific asset

python manage.py generate\_depreciation --asset-tag AST-2025-00001

# Generate for specific year

python manage.py generate\_depreciation --year 2025

**Test 5: Check Maintenance Due**

# Check maintenance due today

python manage.py check\_maintenance\_due

# Check maintenance due in next 7 days

python manage.py check\_maintenance\_due --days 7

**Test 6: Check Warranty Expiring**

# Check warranty expiring in next 30 days

python manage.py check\_warranty\_expiring

# Check warranty expiring in next 60 days

python manage.py check\_warranty\_expiring --days 60

**Test 7: Generate Asset Report**

# Display report on console

python manage.py asset\_report

# Filter by status

python manage.py asset\_report --status ASSIGNED

# Export to CSV

python manage.py asset\_report --export assets\_report.csv

**📊 Usage Examples**

**Example 1: IT Equipment Lifecycle**

# 1. Receive laptop via GRN (auto-creates asset)

# GRN line with serial\_no will trigger asset creation via signal

# 2. Assign to employee

from assets.models import Asset

from users.models import User

laptop = Asset.objects.get(serial\_no='ABC123')

employee = User.objects.get(username='john.doe')

laptop.assign\_to\_user(user=employee, assigned\_by=admin\_user)

# 3. Record maintenance

from assets.models import AssetMaintenanceRecord

maintenance = AssetMaintenanceRecord.objects.create(

asset=laptop,

maintenance\_type='PREVENTIVE',

maintenance\_date='2025-06-15',

performed\_by='IT Support Team',

is\_internal=True,

description='RAM upgrade from 8GB to 16GB',

cost=8000,

next\_maintenance\_date='2025-12-15'

)

# 4. Return to stock

laptop.return\_to\_stock(

returned\_by=employee,

return\_condition='GOOD',

remarks='Employee left company'

)

# 5. Reassign to another employee

new\_employee = User.objects.get(username='jane.smith')

laptop.assign\_to\_user(user=new\_employee, assigned\_by=admin\_user)

**Example 2: Check Assets Needing Attention**

from django.utils import timezone

from datetime import timedelta

# Assets with maintenance due in next 7 days

maintenance\_due = Asset.objects.filter(

next\_maintenance\_date\_\_lte=timezone.now().date() + timedelta(days=7),

next\_maintenance\_date\_\_gte=timezone.now().date(),

is\_active=True

)

# Assets with warranty expiring in next 30 days

warranty\_expiring = Asset.objects.filter(

warranty\_end\_date\_\_lte=timezone.now().date() + timedelta(days=30),

warranty\_end\_date\_\_gte=timezone.now().date(),

is\_active=True

)

# Currently assigned assets

assigned\_assets = Asset.objects.filter(

current\_status='ASSIGNED',

is\_active=True

)

# Available assets

available\_assets = Asset.objects.filter(

current\_status='IN\_STOCK',

is\_active=True

)

**Example 3: Generate Monthly Depreciation**

from assets.models import generate\_depreciation\_schedule

from datetime import date

# For all active assets

assets = Asset.objects.filter(

is\_active=True,

depreciation\_method\_\_in=['STRAIGHT\_LINE', 'DECLINING\_BALANCE']

)

for asset in assets:

generate\_depreciation\_schedule(

asset=asset,

start\_date=date(2025, 1, 1),

end\_date=date(2025, 12, 31)

)

**🔧 Configuration Options**

**Asset Auto-Creation from GRN**

Edit assets/signals.py to customize when assets are auto-created:

# Only create assets for specific categories

def auto\_create\_asset\_from\_grn(sender, instance, created, \*\*kwargs):

if not instance.item.is\_serialized:

return

# Add custom logic here

if instance.item.category.category\_code not in ['IT', 'MACHINERY']:

return # Don't auto-create for other categories

# ... rest of the code

**Depreciation Settings**

Default useful life by category in assets/models.py:

# In Asset.save() method

if grn\_line.item.is\_asset:

# Customize by category

if grn\_line.item.category.category\_code == 'IT':

useful\_life\_years = 3 # IT equipment: 3 years

elif grn\_line.item.category.category\_code == 'MACHINERY':

useful\_life\_years = 10 # Machinery: 10 years

else:

useful\_life\_years = 5 # Default: 5 years

**🎯 Next Steps**

After installing Asset Management:

1. **Populate Existing Assets**
   * Manually create assets for existing equipment
   * Or import from CSV/Excel
2. **Configure Maintenance Schedules**
   * Set maintenance frequencies for critical assets
   * Schedule regular preventive maintenance
3. **Set Up Notifications** (Next Phase)
   * Maintenance due alerts
   * Warranty expiry notifications
   * Assignment reminders
4. **Generate Reports**
   * Asset register
   * Depreciation reports
   * Assignment history
   * Maintenance logs
5. **Move to Approval Workflows** (Next Module)
   * Asset assignment approvals
   * Maintenance approval workflows
   * Disposal approvals

**🐛 Troubleshooting**

**Issue: Assets Not Auto-Creating from GRN**

**Solution:**

# Check if signals are loaded

from assets import signals

# Verify GRN status is ACCEPTED or POSTED

grn = GoodsReceivedNote.objects.get(grn\_number='GRN-2025-0001')

print(f"GRN Status: {grn.status}")

# Manually trigger asset creation

from assets.models import create\_asset\_from\_grn\_line

for line in grn.lines.all():

asset = create\_asset\_from\_grn\_line(line)

if asset:

print(f"Created: {asset.asset\_tag}")

**Issue: Depreciation Not Calculating**

**Solution:**

asset = Asset.objects.get(asset\_tag='AST-2025-00001')

# Check required fields

print(f"Purchase Date: {asset.purchase\_date}")

print(f"Purchase Price: {asset.purchase\_price}")

print(f"Useful Life: {asset.useful\_life\_years}")

print(f"Method: {asset.depreciation\_method}")

# Manually trigger calculation

asset.calculate\_depreciation()

asset.save()

print(f"Book Value: {asset.current\_book\_value}")

**Issue: Migration Errors**

**Solution:**

# Reset migrations (CAUTION: only in development)

python manage.py migrate assets zero

python manage.py showmigrations assets

python manage.py makemigrations assets

python manage.py migrate assets

**✅ Verification Checklist**

* [ ] Assets app added to INSTALLED\_APPS
* [ ] Migrations created and applied
* [ ] Admin interface accessible
* [ ] Can create assets manually
* [ ] Auto-creation from GRN works
* [ ] Asset assignment works
* [ ] Maintenance records can be created
* [ ] Depreciation calculates correctly
* [ ] Management commands run successfully
* [ ] Signals are firing correctly

**📚 Additional Resources**

* **Models Documentation**: See assets/models.py docstrings
* **Admin Documentation**: See assets/admin.py docstrings
* **Signals Documentation**: See assets/signals.py docstrings
* **Command Documentation**: See assets/management/commands/\*.py docstrings

**🎉 Success!**

If all tests pass, your Asset Management module is ready to use!

**What's Next?**

* Approval Workflows Module
* Reports Module
* Frontend Development

Need help? Check the troubleshooting section or review the test scripts above.