Phase 2: MVP Business Modules - TWIST ERP

Implementation Guide

Duration: 10–12 weeks
Version: 1.0
Date: October 2025
Project: TWIST ERP - Visual Drag-and-Drop Multi-Company ERP

1. Phase Overview

Phase 2 focuses on implementing core business modules that deliver immediate value to SMEs: Finance, Inventory, Sales/CRM, and Procurement. All modules are built with multi-company support, inter-company transactions, and real-time integration.

Key Objectives

- Implement Financial Management (GL, AP, AR, Multi-currency)
- Build Inventory & Warehouse Management
- Create Sales & CRM module with pipeline visualization
- Develop Procurement & Supplier Management
- Enable inter-company transactions
- Implement consolidated reporting
- Integrate all modules with event bus

Success Criteria

- Complete Order-to-Cash (O2C) flow functional
- Complete Procure-to-Pay (P2P) flow functional
- Real-time inventory updates from all transactions
- Multi-company consolidation working
- 90%+ test coverage for business logic
- API response time <300ms for complex queries

2. Financial Management Module

2.1 Data Models

Chart of Accounts

```python
# backend/apps/finance/models/accounts.py
from django.db import models
from shared.models import CompanyAwareModel

class AccountType(models.TextChoices):
    ASSET = 'ASSET', 'Asset'
    LIABILITY = 'LIABILITY', 'Liability'
    EQUITY = 'EQUITY', 'Equity'
    REVENUE = 'REVENUE', 'Revenue'
    EXPENSE = 'EXPENSE', 'Expense'

class Account(CompanyAwareModel):
    """
    Chart of Accounts with hierarchical structure
    """
    code = models.CharField(max_length=20)
    name = models.CharField(max_length=255)
    account_type = models.CharField(
        max_length=20,
        choices=AccountType.choices
    )
    parent_account = models.ForeignKey(
        'self',
        on_delete=models.PROTECT,
        null=True,
        blank=True,
        related_name='sub_accounts'
    )

    # Configuration
    is_active = models.BooleanField(default=True)
    is_bank_account = models.BooleanField(default=False)
    is_control_account = models.BooleanField(default=False)
    allow_direct_posting = models.BooleanField(default=True)

    # Balance tracking
    current_balance = models.DecimalField(
        max_digits=20,
        decimal_places=2,
        default=0
    )

    # Currency
    currency = models.CharField(max_length=3, default='BDT')

    class Meta:
        unique_together = [['company', 'code']]
        ordering = ['code']
        indexes = [
            models.Index(fields=['company', 'account_type']),
            models.Index(fields=['company', 'is_active']),
        ]

    def __str__(self):
        return f"{self.code} - {self.name}"

    def get_balance(self, date=None):
        """Calculate account balance up to a date"""
        from .journal import JournalEntry

        entries = JournalEntry.objects.filter(
            company=self.company,
            account=self,
            status='POSTED'
        )

        if date:
            entries = entries.filter(entry_date__lte=date)

        debit_total = entries.aggregate(
            total=models.Sum('debit_amount')
        )['total'] or 0

        credit_total = entries.aggregate(
            total=models.Sum('credit_amount')
        )['total'] or 0

        # Asset, Expense: Debit increases
        # Liability, Equity, Revenue: Credit increases
        if self.account_type in [AccountType.ASSET, AccountType.EXPENSE]:
            return debit_total - credit_total
        else:
            return credit_total - debit_total
```

Journal Entry System

```
# backend/apps/finance/models/journal.py
from django.db import models
from django.db import transaction
from shared.models import CompanyAwareModel

class JournalStatus(models.TextChoices):
    DRAFT = 'DRAFT', 'Draft'
    POSTED = 'POSTED', 'Posted'
    CANCELLED = 'CANCELLED', 'Cancelled'

class Journal(CompanyAwareModel):
    """
    Journal (book of accounts)
    """
    code = models.CharField(max_length=20)
    name = models.CharField(max_length=255)
    type = models.CharField(
        max_length=20,
        choices=[
            ('GENERAL', 'General Journal'),
            ('SALES', 'Sales Journal'),
            ('PURCHASE', 'Purchase Journal'),
            ('CASH', 'Cash Journal'),
            ('BANK', 'Bank Journal'),
        ]
    )
    is_active = models.BooleanField(default=True)

    class Meta:
        unique_together = [['company', 'code']]

class JournalVoucher(CompanyAwareModel):
    """
    Journal Voucher - header for multiple entries
    """
    voucher_number = models.CharField(max_length=50)
    journal = models.ForeignKey(Journal, on_delete=models.PROTECT)
    entry_date = models.DateField()
    period = models.CharField(max_length=7)  # YYYY-MM

    reference = models.CharField(max_length=100, blank=True)
    description = models.TextField()

    status = models.CharField(
        max_length=20,
        choices=JournalStatus.choices,
        default=JournalStatus.DRAFT
    )

    # Linked document
    source_document_type = models.CharField(max_length=50, blank=True)
    source_document_id = models.IntegerField(null=True, blank=True)

    posted_by = models.ForeignKey(
        'users.User',
        on_delete=models.PROTECT,
        null=True,
        related_name='posted_vouchers'
    )
    posted_at = models.DateTimeField(null=True, blank=True)

    class Meta:
        unique_together = [['company', 'voucher_number']]
        indexes = [
            models.Index(fields=['company', 'entry_date']),
            models.Index(fields=['company', 'status']),
        ]

class JournalEntry(models.Model):
    """
    Individual journal entry line (debit or credit)
    """
    voucher = models.ForeignKey(
        JournalVoucher,
        on_delete=models.CASCADE,
        related_name='entries'
    )
    line_number = models.IntegerField()
    account = models.ForeignKey('Account', on_delete=models.PROTECT)

    debit_amount = models.DecimalField(
        max_digits=20,
        decimal_places=2,
        default=0
    )
    credit_amount = models.DecimalField(
        max_digits=20,
        decimal_places=2,
        default=0
    )

    description = models.CharField(max_length=255, blank=True)

    # Analytical dimensions
    cost_center = models.ForeignKey(
        'CostCenter',
        on_delete=models.PROTECT,
        null=True,
        blank=True
    )
    project = models.ForeignKey(
        'projects.Project',
        on_delete=models.PROTECT,
        null=True,
        blank=True
    )

    class Meta:
        ordering = ['voucher', 'line_number']
        indexes = [
            models.Index(fields=['account', 'voucher__entry_date']),
        ]

    @property
    def company(self):
        return self.voucher.company
```

Accounts Payable/Receivable

```
# backend/apps/finance/models/payables.py
from django.db import models
from shared.models import CompanyAwareModel

class Invoice(CompanyAwareModel):
    """
    Base invoice model (AP and AR)
    """
    invoice_number = models.CharField(max_length=50)
    invoice_type = models.CharField(
        max_length=10,
        choices=[
            ('AR', 'Accounts Receivable'),
            ('AP', 'Accounts Payable'),
        ]
    )

    # Partner (Customer or Supplier)
    partner_type = models.CharField(max_length=20)
    partner_id = models.IntegerField()

    invoice_date = models.DateField()
    due_date = models.DateField()

    # Amounts
    subtotal = models.DecimalField(max_digits=20, decimal_places=2)
    tax_amount = models.DecimalField(max_digits=20, decimal_places=2, default=0)
    discount_amount = models.DecimalField(max_digits=20, decimal_places=2, default=0)
    total_amount = models.DecimalField(max_digits=20, decimal_places=2)
    paid_amount = models.DecimalField(max_digits=20, decimal_places=2, default=0)

    currency = models.CharField(max_length=3, default='BDT')
    exchange_rate = models.DecimalField(max_digits=10, decimal_places=6, default=1)

    status = models.CharField(
        max_length=20,
        choices=[
            ('DRAFT', 'Draft'),
            ('POSTED', 'Posted'),
            ('PARTIAL', 'Partially Paid'),
            ('PAID', 'Fully Paid'),
            ('CANCELLED', 'Cancelled'),
        ],
        default='DRAFT'
    )

    # Linked documents
    journal_voucher = models.ForeignKey(
        'JournalVoucher',
        on_delete=models.PROTECT,
        null=True,
        blank=True
    )

    notes = models.TextField(blank=True)

    class Meta:
        unique_together = [['company', 'invoice_number']]
        indexes = [
            models.Index(fields=['company', 'invoice_type', 'status']),
            models.Index(fields=['company', 'due_date']),
        ]

    @property
    def balance_due(self):
        return self.total_amount - self.paid_amount

    @property
    def is_overdue(self):
        from django.utils import timezone
        return (
            self.balance_due &gt; 0 and
            self.due_date &lt; timezone.now().date() and
            self.status not in ['PAID', 'CANCELLED']
        )

class InvoiceLine(models.Model):
    """
    Invoice line items
    """
    invoice = models.ForeignKey(
        Invoice,
        on_delete=models.CASCADE,
        related_name='lines'
    )
    line_number = models.IntegerField()

    description = models.CharField(max_length=255)
    quantity = models.DecimalField(max_digits=15, decimal_places=3, default=1)
    unit_price = models.DecimalField(max_digits=20, decimal_places=2)
    tax_rate = models.DecimalField(max_digits=5, decimal_places=2, default=0)
    discount_percent = models.DecimalField(max_digits=5, decimal_places=2, default=0)

    line_total = models.DecimalField(max_digits=20, decimal_places=2)

    # Link to product/service
    product_id = models.IntegerField(null=True, blank=True)
    account = models.ForeignKey('Account', on_delete=models.PROTECT)

    class Meta:
        ordering = ['invoice', 'line_number']
```

Payment System

```
# backend/apps/finance/models/payments.py
from django.db import models
from shared.models import CompanyAwareModel

class Payment(CompanyAwareModel):
    """
    Payment transactions
    """
    payment_number = models.CharField(max_length=50)
    payment_date = models.DateField()

    payment_type = models.CharField(
        max_length=20,
        choices=[
            ('RECEIPT', 'Customer Payment'),
            ('PAYMENT', 'Supplier Payment'),
        ]
    )

    # Payment method
    payment_method = models.CharField(
        max_length=20,
        choices=[
            ('CASH', 'Cash'),
            ('BANK', 'Bank Transfer'),
            ('CHEQUE', 'Cheque'),
            ('CARD', 'Card'),
            ('MOBILE', 'Mobile Payment'),
        ]
    )

    bank_account = models.ForeignKey(
        'Account',
        on_delete=models.PROTECT,
        null=True,
        blank=True
    )

    amount = models.DecimalField(max_digits=20, decimal_places=2)
    currency = models.CharField(max_length=3, default='BDT')

    # Partner
    partner_type = models.CharField(max_length=20)
    partner_id = models.IntegerField()

    reference = models.CharField(max_length=100, blank=True)
    notes = models.TextField(blank=True)

    status = models.CharField(
        max_length=20,
        choices=[
            ('DRAFT', 'Draft'),
            ('POSTED', 'Posted'),
            ('RECONCILED', 'Reconciled'),
            ('CANCELLED', 'Cancelled'),
        ],
        default='DRAFT'
    )

    journal_voucher = models.ForeignKey(
        'JournalVoucher',
        on_delete=models.PROTECT,
        null=True,
        blank=True
    )

    class Meta:
        unique_together = [['company', 'payment_number']]

class PaymentAllocation(models.Model):
    """
    Payment allocation to invoices
    """
    payment = models.ForeignKey(
        Payment,
        on_delete=models.CASCADE,
        related_name='allocations'
    )
    invoice = models.ForeignKey(
        'Invoice',
        on_delete=models.PROTECT
    )
    allocated_amount = models.DecimalField(max_digits=20, decimal_places=2)

    class Meta:
        unique_together = [['payment', 'invoice']]
```

2.2 Finance API Endpoints

```
# backend/apps/finance/views/account_views.py
from rest_framework import viewsets, status
from rest_framework.decorators import action
from rest_framework.response import Response
from django_filters.rest_framework import DjangoFilterBackend
from apps.finance.models import Account
from apps.finance.serializers import AccountSerializer

class AccountViewSet(viewsets.ModelViewSet):
    """
    Account CRUD and operations
    """
    serializer_class = AccountSerializer
    filter_backends = [DjangoFilterBackend]
    filterset_fields = ['account_type', 'is_active', 'parent_account']

    def get_queryset(self):
        return Account.objects.filter(
            company=self.request.company
        ).select_related('parent_account')

    @action(detail=True, methods=['get'])
    def balance(self, request, pk=None):
        """Get account balance"""
        account = self.get_object()
        date = request.query_params.get('date')
        balance = account.get_balance(date=date)

        return Response({
            'account': account.code,
            'balance': balance,
            'date': date or 'current'
        })

    @action(detail=True, methods=['get'])
    def transactions(self, request, pk=None):
        """Get account transactions"""
        account = self.get_object()
        entries = account.journalentry_set.select_related(
            'voucher'
        ).order_by('-voucher__entry_date')[:100]

        return Response({
            'account': account.code,
            'transactions': [
                {
                    'date': e.voucher.entry_date,
                    'voucher': e.voucher.voucher_number,
                    'description': e.description,
                    'debit': e.debit_amount,
                    'credit': e.credit_amount,
                }
                for e in entries
            ]
        })
```

3. Inventory & Warehouse Management

3.1 Data Models

```
# backend/apps/inventory/models/product.py
from django.db import models
from shared.models import CompanyAwareModel

class ProductCategory(CompanyAwareModel):
    """Product categorization"""
    code = models.CharField(max_length=20)
    name = models.CharField(max_length=255)
    parent_category = models.ForeignKey(
        'self',
        on_delete=models.PROTECT,
        null=True,
        blank=True
    )
    is_active = models.BooleanField(default=True)

    class Meta:
        unique_together = [['company', 'code']]
        verbose_name_plural = 'Product Categories'

class UnitOfMeasure(CompanyAwareModel):
    """Units for inventory measurement"""
    code = models.CharField(max_length=10)
    name = models.CharField(max_length=50)
    is_active = models.BooleanField(default=True)

    class Meta:
        unique_together = [['company', 'code']]

class Product(CompanyAwareModel):
    """
    Product/Item master
    """
    code = models.CharField(max_length=50)
    name = models.CharField(max_length=255)
    description = models.TextField(blank=True)

    category = models.ForeignKey(
        ProductCategory,
        on_delete=models.PROTECT
    )

    product_type = models.CharField(
        max_length=20,
        choices=[
            ('GOODS', 'Goods'),
            ('SERVICE', 'Service'),
            ('CONSUMABLE', 'Consumable'),
        ],
        default='GOODS'
    )

    # Units
    uom = models.ForeignKey(
        UnitOfMeasure,
        on_delete=models.PROTECT,
        related_name='products'
    )

    # Tracking
    track_inventory = models.BooleanField(default=True)
    track_serial = models.BooleanField(default=False)
    track_batch = models.BooleanField(default=False)

    # Pricing
    cost_price = models.DecimalField(
        max_digits=20,
        decimal_places=2,
        default=0
    )
    selling_price = models.DecimalField(
        max_digits=20,
        decimal_places=2,
        default=0
    )

    # Inventory control
    reorder_level = models.DecimalField(
        max_digits=15,
        decimal_places=3,
        default=0
    )
    reorder_quantity = models.DecimalField(
        max_digits=15,
        decimal_places=3,
        default=0
    )
    # Accounts
    inventory_account = models.ForeignKey(
        'finance.Account',
        on_delete=models.PROTECT,
        related_name='inventory_products'
    )
    income_account = models.ForeignKey(
        'finance.Account',
        on_delete=models.PROTECT,
        related_name='income_products'
    )
    expense_account = models.ForeignKey(
        'finance.Account',
        on_delete=models.PROTECT,
        related_name='expense_products'
    )

    is_active = models.BooleanField(default=True)

    class Meta:
        unique_together = [['company', 'code']]
        indexes = [
            models.Index(fields=['company', 'category']),
            models.Index(fields=['company', 'is_active']),
        ]

    def __str__(self):
        return f"{self.code} - {self.name}"
```

Warehouse and Stock

```
# backend/apps/inventory/models/warehouse.py
from django.db import models
from shared.models import CompanyAwareModel

class Warehouse(CompanyAwareModel):
    """
    Warehouse/Location master
    """
    code = models.CharField(max_length=20)
    name = models.CharField(max_length=255)
    address = models.TextField(blank=True)

    warehouse_type = models.CharField(
        max_length=20,
        choices=[
            ('MAIN', 'Main Warehouse'),
            ('TRANSIT', 'Transit Location'),
            ('RETAIL', 'Retail Store'),
            ('VIRTUAL', 'Virtual Location'),
        ],
        default='MAIN'
    )
    is_active = models.BooleanField(default=True)

    class Meta:
        unique_together = [['company', 'code']]

class StockLedger(models.Model):
    """
    Double-entry stock ledger
    Immutable transaction log
    """
    company = models.ForeignKey('companies.Company', on_delete=models.PROTECT)

    transaction_date = models.DateTimeField()
    transaction_type = models.CharField(
        max_length=20,
        choices=[
            ('RECEIPT', 'Stock Receipt'),
            ('ISSUE', 'Stock Issue'),
            ('TRANSFER', 'Transfer'),
            ('ADJUSTMENT', 'Adjustment'),
        ]
    )

    product = models.ForeignKey('Product', on_delete=models.PROTECT)
    warehouse = models.ForeignKey('Warehouse', on_delete=models.PROTECT)

    # Quantities (positive for IN, negative for OUT)
    quantity = models.DecimalField(max_digits=15, decimal_places=3)

    # Valuation
    rate = models.DecimalField(max_digits=20, decimal_places=2)
    value = models.DecimalField(max_digits=20, decimal_places=2)

    # Running totals (updated by trigger)
    balance_qty = models.DecimalField(max_digits=15, decimal_places=3)
    balance_value = models.DecimalField(max_digits=20, decimal_places=2)

    # Source document
    source_document_type = models.CharField(max_length=50)
    source_document_id = models.IntegerField()

    batch_no = models.CharField(max_length=50, blank=True)
    serial_no = models.CharField(max_length=50, blank=True)

    created_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        ordering = ['transaction_date', 'id']
        indexes = [
            models.Index(fields=['company', 'product', 'warehouse']),
            models.Index(fields=['transaction_date']),
        ]
```

Stock Movement Documents

```python
# backend/apps/inventory/models/movement.py
from django.db import models
from shared.models import CompanyAwareModel

class StockMovement(CompanyAwareModel):
    """
    Stock movement document (Receipt, Issue, Transfer)
    """
    movement_number = models.CharField(max_length=50)
    movement_date = models.DateField()

    movement_type = models.CharField(
        max_length=20,
        choices=[
            ('RECEIPT', 'Goods Receipt'),
            ('ISSUE', 'Goods Issue'),
            ('TRANSFER', 'Stock Transfer'),
            ('ADJUSTMENT', 'Stock Adjustment'),
        ]
    )

    from_warehouse = models.ForeignKey(
        'Warehouse',
        on_delete=models.PROTECT,
        null=True,
        blank=True,
        related_name='movements_out'
    )
    to_warehouse = models.ForeignKey(
        'Warehouse',
        on_delete=models.PROTECT,
        related_name='movements_in'
    )

    reference = models.CharField(max_length=100, blank=True)
    notes = models.TextField(blank=True)

    status = models.CharField(
        max_length=20,
        choices=[
            ('DRAFT', 'Draft'),
            ('SUBMITTED', 'Submitted'),
            ('COMPLETED', 'Completed'),
            ('CANCELLED', 'Cancelled'),
        ],
        default='DRAFT'
    )

    posted_at = models.DateTimeField(null=True, blank=True)

    class Meta:
        unique_together = [['company', 'movement_number']]

class StockMovementLine(models.Model):
    """Stock movement line items"""
    movement = models.ForeignKey(
        StockMovement,
        on_delete=models.CASCADE,
        related_name='lines'
    )
    line_number = models.IntegerField()

    product = models.ForeignKey('Product', on_delete=models.PROTECT)
    quantity = models.DecimalField(max_digits=15, decimal_places=3)
    rate = models.DecimalField(max_digits=20, decimal_places=2)

    batch_no = models.CharField(max_length=50, blank=True)
    serial_no = models.CharField(max_length=50, blank=True)

    class Meta:
        ordering = ['movement', 'line_number']
```

4. Sales & CRM Module

4.1 Data Models

```
# backend/apps/sales/models/customer.py
from django.db import models
from shared.models import CompanyAwareModel

class Customer(CompanyAwareModel):
    """Customer master"""
    code = models.CharField(max_length=20)
    name = models.CharField(max_length=255)

    # Contact
    email = models.EmailField(blank=True)
    phone = models.CharField(max_length=20, blank=True)
    mobile = models.CharField(max_length=20, blank=True)

    # Address
    billing_address = models.TextField(blank=True)
    shipping_address = models.TextField(blank=True)

    # Credit terms
    credit_limit = models.DecimalField(
        max_digits=20,
        decimal_places=2,
        default=0
    )
    payment_terms = models.IntegerField(
        default=30,
        help_text="Payment terms in days"
    )

    # Accounts
    receivable_account = models.ForeignKey(
        'finance.Account',
        on_delete=models.PROTECT
    )

    # Status
    customer_status = models.CharField(
        max_length=20,
        choices=[
            ('LEAD', 'Lead'),
            ('PROSPECT', 'Prospect'),
            ('ACTIVE', 'Active Customer'),
            ('INACTIVE', 'Inactive'),
        ],
        default='LEAD'
    )

    is_active = models.BooleanField(default=True)

    class Meta:
        unique_together = [['company', 'code']]
        indexes = [
            models.Index(fields=['company', 'customer_status']),
        ]

class SalesOrder(CompanyAwareModel):
    """Sales order document"""
    order_number = models.CharField(max_length=50)
    order_date = models.DateField()

    customer = models.ForeignKey('Customer', on_delete=models.PROTECT)

    # Delivery
    delivery_date = models.DateField()
    shipping_address = models.TextField()

    # Amounts
    subtotal = models.DecimalField(max_digits=20, decimal_places=2, default=0)
    tax_amount = models.DecimalField(max_digits=20, decimal_places=2, default=0)
    discount_amount = models.DecimalField(max_digits=20, decimal_places=2, default=0)
    total_amount = models.DecimalField(max_digits=20, decimal_places=2)

    status = models.CharField(
        max_length=20,
        choices=[
            ('DRAFT', 'Draft'),
            ('CONFIRMED', 'Confirmed'),
            ('PARTIAL', 'Partially Delivered'),
            ('DELIVERED', 'Delivered'),
            ('INVOICED', 'Invoiced'),
            ('CANCELLED', 'Cancelled'),
        ],
        default='DRAFT'
    )

    notes = models.TextField(blank=True)
    class Meta:
        unique_together = [['company', 'order_number']]
        indexes = [
            models.Index(fields=['company', 'order_date']),
            models.Index(fields=['company', 'customer', 'status']),
        ]

class SalesOrderLine(models.Model):
    """Sales order line items"""
    order = models.ForeignKey(
        SalesOrder,
        on_delete=models.CASCADE,
        related_name='lines'
    )
    line_number = models.IntegerField()

    product = models.ForeignKey('inventory.Product', on_delete=models.PROTECT)
    description = models.CharField(max_length=255, blank=True)

    quantity = models.DecimalField(max_digits=15, decimal_places=3)
    unit_price = models.DecimalField(max_digits=20, decimal_places=2)
    discount_percent = models.DecimalField(max_digits=5, decimal_places=2, default=0)
    tax_rate = models.DecimalField(max_digits=5, decimal_places=2, default=0)

    line_total = models.DecimalField(max_digits=20, decimal_places=2)

    delivered_qty = models.DecimalField(max_digits=15, decimal_places=3, default=0)

    warehouse = models.ForeignKey('inventory.Warehouse', on_delete=models.PROTECT)

    class Meta:
        ordering = ['order', 'line_number']
```

5. Procurement Module

```
# backend/apps/procurement/models/supplier.py
from django.db import models
from shared.models import CompanyAwareModel

class Supplier(CompanyAwareModel):
    """Supplier/Vendor master"""
    code = models.CharField(max_length=20)
    name = models.CharField(max_length=255)

    email = models.EmailField(blank=True)
    phone = models.CharField(max_length=20, blank=True)
    address = models.TextField(blank=True)

    # Payment terms
    payment_terms = models.IntegerField(
        default=30,
        help_text="Payment terms in days"
    )
    payable_account = models.ForeignKey(
        'finance.Account',
        on_delete=models.PROTECT
    )

    is_active = models.BooleanField(default=True)

    class Meta:
        unique_together = [['company', 'code']]

class PurchaseOrder(CompanyAwareModel):
    """Purchase order document"""
    order_number = models.CharField(max_length=50)
    order_date = models.DateField()

    supplier = models.ForeignKey('Supplier', on_delete=models.PROTECT)

    expected_delivery_date = models.DateField()
    delivery_address = models.TextField()

    subtotal = models.DecimalField(max_digits=20, decimal_places=2, default=0)
    tax_amount = models.DecimalField(max_digits=20, decimal_places=2, default=0)
    total_amount = models.DecimalField(max_digits=20, decimal_places=2)

    status = models.CharField(
        max_length=20,
        choices=[
            ('DRAFT', 'Draft'),
            ('SUBMITTED', 'Submitted'),
            ('APPROVED', 'Approved'),
            ('PARTIAL', 'Partially Received'),
            ('RECEIVED', 'Received'),
            ('CANCELLED', 'Cancelled'),
        ],
        default='DRAFT'
    )

    notes = models.TextField(blank=True)

    class Meta:
        unique_together = [['company', 'order_number']]
```

6. Inter-Company Transactions

```
# backend/apps/intercompany/models.py
from django.db import models
from shared.models import CompanyAwareModel

class InterCompanyTransaction(models.Model):
    """
    Tracks inter-company transactions for consolidation
    """
    transaction_type = models.CharField(
        max_length=20,
        choices=[
            ('SALE', 'Inter-company Sale'),
            ('TRANSFER', 'Inventory Transfer'),
            ('LOAN', 'Inter-company Loan'),
        ]
    )

    from_company = models.ForeignKey(
        'companies.Company',
        on_delete=models.PROTECT,
        related_name='ic_transactions_out'
    )
    to_company = models.ForeignKey(
        'companies.Company',
        on_delete=models.PROTECT,
        related_name='ic_transactions_in'
    )

    transaction_date = models.DateField()
    amount = models.DecimalField(max_digits=20, decimal_places=2)

    # Linked documents
    from_document_type = models.CharField(max_length=50)
    from_document_id = models.IntegerField()
    to_document_type = models.CharField(max_length=50)
    to_document_id = models.IntegerField()

    # Elimination tracking
    is_eliminated = models.BooleanField(default=False)
    elimination_voucher = models.ForeignKey(
        'finance.JournalVoucher',
        on_delete=models.SET_NULL,
        null=True,
        blank=True
    )

    created_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        indexes = [
            models.Index(fields=['from_company', 'to_company', 'transaction_date']),
        ]
```

7. Testing Requirements

Unit Tests Example

```
# backend/apps/finance/tests/test_accounts.py
from django.test import TestCase
from apps.companies.models import Company
from apps.finance.models import Account, AccountType

class AccountTestCase(TestCase):
    def setUp(self):
        self.company = Company.objects.create(
            code='TEST',
            name='Test Company',
            fiscal_year_start='2024-01-01'
        )

    def test_account_creation(self):
        """Test creating an account"""
        account = Account.objects.create(
            company=self.company,
            code='1000',
            name='Cash',
            account_type=AccountType.ASSET
        )
        self.assertEqual(account.code, '1000')
        self.assertEqual(account.current_balance, 0)

    def test_account_hierarchy(self):
        """Test parent-child account relationships"""
        parent = Account.objects.create(
            company=self.company,
            code='1000',
            name='Assets',
            account_type=AccountType.ASSET
        )
        child = Account.objects.create(
            company=self.company,
            code='1010',
            name='Current Assets',
            account_type=AccountType.ASSET,
            parent_account=parent
        )
        self.assertEqual(child.parent_account, parent)
        self.assertIn(child, parent.sub_accounts.all())
```

8. Implementation Checklist

Weeks 1-3: Finance Module

- ☐ Implement Chart of Accounts models
- ☐ Build Journal Entry system
- ☐ Create AR/AP models
- ☐ Implement Payment system
- ☐ Build Finance API endpoints
- ☐ Write unit tests (80%+ coverage)

Weeks 4-6: Inventory Module

- ☐ Implement Product master
- ☐ Build Warehouse models
- ☐ Create Stock Ledger system
- ☐ Implement Stock Movement documents
- ☐ Build Inventory API endpoints
- ☐ Write unit tests

Weeks 7-9: Sales & Procurement

- ☐ Implement Customer/Supplier masters
- ☐ Build Sales Order system
- ☐ Create Purchase Order system
- ☐ Implement order workflows
- ☐ Build Sales/Procurement APIs
- ☐ Write unit tests

Weeks 10-12: Integration & Testing

- ☐ Implement inter-company transactions
- ☐ Build consolidated reporting
- ☐ Create end-to-end workflows (O2C, P2P)
- ☐ Integration testing
- ☐ Performance optimization
- ☐ Documentation

Document Control:

- **Version:** 1.0
- **Dependencies:** Phase 0, Phase 1 complete
- **Next Phase:** Phase 3 - Data Migration Engine