GitHub & Django Project Setup Walkthrough

Project Name: POS Web System

Step 1: Create a New GitHub Repository

1. Go to GitHub and log in to your account
2. Click the "+" icon in the top right corner and select "New repository"
3. Name your repository (e.g., "django-pos-system")
4. Add a description (optional)
5. Choose "Public" or "Private" visibility
6. **DO NOT** initialize with README (we'll create this manually)
7. **DO NOT** initialize with .gitignore (we'll create this manually)
8. Click "Create repository"

Step 2: Set Up Local Repository

```
# Create project directory
mkdir django-pos-system
cd django-pos-system

# Initialize git repository
git init

# Connect to your GitHub repository
git remote add origin https://github.com/your-username/django-pos-system.git
```

Step 3: Create **README.md** File

```
# Create and switch to a new branch for README
git checkout -b setup/project/readme_file
```

Create README.md file with this content:

```
# Django POS System

A Point of Sale system built with Django featuring role-based access for admin, manager, and teller users.

## Features
```

- User authentication with different roles
- Admin user management
- Manager product management and sales reports
- Teller POS system with transaction processing
- Responsive web interface

## Installation

1. Clone the repository
2. Create virtual environment: `python -m venv env`
3. Activate environment: `source env/bin/activate` (Linux/Mac) or `env\Scripts\activate` (Windows)
4. Install requirements: `pip install -r requirements.txt`
5. Run migrations: `python manage.py migrate`
6. Create superuser: `python manage.py createsuperuser`
7. Run server: `python manage.py runserver`

## Technology Stack

- Django
- SQLite (can be configured for other databases)
- HTML/CSS/JavaScript

# Add, commit, and push README

```
git add README.md
git commit -m "setup(project): readme_file"
git push origin setup/project/readme_file:setup/project/readme_file
```

Go to GitHub, create a Pull Request for this branch, have it reviewed, then merge it.

Step 4: Set Up .gitignore and Environment

# Switch back to main and pull changes

```
git checkout main
git pull origin main
```

```
# Create new branch for gitignore
git checkout -b setup/project/gitignore_file
```

Create .gitignore file with this content:

gitignore

```
# Django
*.log
*.pot
*.pyc
__pycache__/
db.sqlite3
media/

# Environment
env/
venv/
ENV/
env.bak/
venv.bak/

# IDE
.vscode/
.idea/
*.swp
*.swo

# OS
.DS_Store
Thumbs.db

# Coverage reports
htmlcov/
.coverage
.coverage.*
coverage.xml
*.cover
.hypothesis/
```

```
.pytest_cache/

# Jupyter Notebook
.ipynb_checkpoints

# Package files
*.egg
*.egg-info
dist/
build/
eggs/
parts/
var/
sdist/
develop-eggs/
.installed.cfg
lib/
lib64/

# Installer logs
pip-log.txt
pip-delete-this-directory.txt

# Unit test / coverage reports
htmlcov/
.tox/
.coverage
.coverage.*
.cache
nosetests.xml
coverage.xml
*.cover
.hypothesis/
.pytest_cache/

# Translations
*.mo

# Static files
/staticfiles/

# macOS
.DS_Store
```

```
# Add, commit, and push .gitignore
git add .gitignore
git commit -m "setup(project): gitignore_file"
git push origin setup/project/gitignore_file:setup/project/gitignore_file
```

Create PR, get it reviewed, then merge.

Step 5: Set Up Virtual Environment and Django

```
# Switch back to main and pull changes
git checkout main
git pull origin main

# Create new branch for environment setup
git checkout -b setup/project/virtualenv_django

# Create virtual environment
python -m venv env

# Activate environment
# On Windows:
env\Scripts\activate

# Install Django
pip install django

# Create requirements.txt
pip freeze > requirements.txt
bash
# Add, commit, and push requirements
git add requirements.txt
git commit -m "setup(project): virtualenv_django"
git push origin setup/project/virtualenv_django:setup/project/virtualenv_django
```

Create PR, get it reviewed, then merge.

Step 6: Create Django Project

```
bash
# Switch back to main and pull changes
git checkout main
git pull origin main
```

```
# Create new branch for project creation
git checkout -b setup/project/django_project
```

bash

```
# Create Django project
django-admin startproject pos_project .

# Test the project works
python manage.py runserver
# Press Ctrl+C to stop the server
```

bash

```
# Add, commit, and push project files
git add pos_project/ manage.py
git commit -m "setup(project): django_project"
git push origin setup/project/django_project:setup/project/django_project
```

Create PR, get it reviewed, then merge.

Step 7: Create Django App and Basic Structure

bash

```
# Switch back to main and pull changes
git checkout main
git pull origin main

# Create new branch for app creation
git checkout -b feature/app/pos_app_structure
```

bash

```
# Create the app
python manage.py startapp pos_app

# Create directories for static files and templates
mkdir -p static/pos_app/css static/pos_app/js static/pos_app/images templates/pos_app
```

bash

```
# Add, commit, and push app structure
git add pos_app/ static/ templates/
git commit -m "feature(app): pos_app_structure"
git push origin feature/app/pos_app_structure:feature/app/pos_app_structure
```

Create PR, get it reviewed, then merge.

Step 8: Configure Settings

bash

```
# Switch back to main and pull changes
git checkout main
git pull origin main

# Create new branch for settings configuration
git checkout -b tech/settings/configure_settings
```

Update pos_project/settings.py as shown in the previous response.

bash

```
# Add, commit, and push settings changes
git add pos_project/settings.py
git commit -m "tech(settings): configure_settings"
git push origin tech/settings/configure_settings:tech/settings/configure_settings
```

Create PR, get it reviewed, then merge.

Step 9: Create Models

bash

```
# Switch back to main and pull changes
git checkout main
git pull origin main

# Create new branch for models
git checkout -b feature/models/create_models
```

Create models in pos_app/models.py as shown in the previous response.

bash

```
# Make migrations
python manage.py makemigrations

# Add, commit, and push models and migrations
git add pos_app/models.py pos_app/migrations/
git commit -m "feature(models): create_models"
git push origin feature/models/create_models:feature/models/create_models
```

Create PR, get it reviewed, then merge.

Step 10: Continue This Pattern

Continue this process for each component:

1. Forms
2. Views
3. URLs
4. Templates
5. Static files

6. Admin configuration

For each component:

1. Switch to main and pull latest changes
2. Create a new branch with proper naming convention
3. Make your changes
4. Test your changes
5. Add, commit, and push with proper commit message
6. Create PR on GitHub
7. Have it reviewed
8. Merge after approval
9. Repeat for the next component

Step 11: Final Steps

After all components are implemented:

bash

```
# Switch to main and pull all changes
git checkout main
git pull origin main

# Create superuser
python manage.py createsuperuser

# Run migrations
python manage.py migrate

# Run the server
python manage.py runserver
```

Visit http://127.0.0.1:8000 in your browser to see the application.

Important Tips for Students

1. **Always start from the main branch**: git checkout main && git pull origin main
2. **Create a new branch for each feature**: Use the proper naming convention
3. **Test your code**: Make sure it works before committing
4. **Write clear commit messages**: Follow the format in the PDF
5. **Push to the correct branch**: Use the full push command
6. **Create PRs for review**: Don't merge your own code without review
7. **Resolve conflicts early**: If there are conflicts, resolve them in your branch before merging

POS System Project Walkthrough

Step 1: Environment Setup

bash

```
# Create project directory
mkdir django-pos-system
cd django-pos-system

# Create virtual environment
python -m venv env

# Activate virtual environment
# On Windows:
env\Scripts\activate
# On macOS/Linux:
source env/bin/activate

# Install Django
pip install django
```

Step 2: Create Django Project and App

bash

```
# Create Django project
django-admin startproject pos_project .

# Create main app
python manage.py startapp pos_app

# Create directories for static files and templates
mkdir -p static/pos_app/css static/pos_app/js static/pos_app/images templates/pos_app
```

Step 3: Project Structure

Your project structure should look like this:

text

```
django-pos-system/
├── env/
├── pos_project/
├── pos_app/
├── static/
│   └── pos_app/
│      ├── css/
│      ├── js/
│     └── images/
├── templates/
│   └── pos_app/
├── manage.py
└── requirements.txt
```

Step 4: Configure Settings (pos_project/settings.py)

python

```
import os
from pathlib import Path

BASE_DIR = Path(__file__).resolve().parent.parent

SECRET_KEY = 'your-secret-key-here'  # Change this in production

DEBUG = True

ALLOWED_HOSTS = []

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'pos_app',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'pos_project.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'pos_project.wsgi.application'

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_TZ = True

STATIC_URL = '/static/'
STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static')]

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

LOGIN_REDIRECT_URL = 'dashboard'
LOGOUT_REDIRECT_URL = 'login'
```

Step 5: Define Models (pos_app/models.py)

python

```
from django.db import models
from django.contrib.auth.models import User
from django.utils import timezone

class Profile(models.Model):
    ROLE_CHOICES = (
        ('admin', 'Admin'),
        ('manager', 'Manager'),
        ('teller', 'Teller'),
    )
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    role = models.CharField(max_length=10, choices=ROLE_CHOICES)
    is_active = models.BooleanField(default=True)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return f"{self.user.username} - {self.role}"
```

```
class Product(models.Model):
    STATUS_CHOICES = (
        ('available', 'Available'),
        ('not_available', 'Not Available'),
    )
    name = models.CharField(max_length=100)
    description = models.TextField(blank=True)
    price = models.DecimalField(max_digits=10, decimal_places=2)
    status = models.CharField(max_length=15, choices=STATUS_CHOICES,
default='available')
    created_by = models.ForeignKey(User, on_delete=models.CASCADE)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.name

class Transaction(models.Model):
    teller = models.ForeignKey(User, on_delete=models.CASCADE,
related_name='transactions')
    customer_name = models.CharField(max_length=100)
    total_amount = models.DecimalField(max_digits=10, decimal_places=2)
    payment_method = models.CharField(max_length=50)
    transaction_date = models.DateTimeField(default=timezone.now)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"Transaction #{self.id} - {self.customer_name}"

class TransactionItem(models.Model):
    transaction = models.ForeignKey(Transaction, on_delete=models.CASCADE,
related_name='items')
    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    quantity = models.PositiveIntegerField()
    price = models.DecimalField(max_digits=10, decimal_places=2)

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

Step 6: Create Forms (pos_app/forms.py)

python

```
from django import forms
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm
from .models import Profile, Product, Transaction, TransactionItem

class UserRegisterForm(UserCreationForm):
    email = forms.EmailField()
    first_name = forms.CharField(max_length=30)
    last_name = forms.CharField(max_length=30)
    role = forms.ChoiceField(choices=Profile.ROLE_CHOICES)

    class Meta:
        model = User
        fields = ['username', 'email', 'first_name', 'last_name', 'password1', 'password2', 'role']

class UserUpdateForm(forms.ModelForm):
    email = forms.EmailField()
    first_name = forms.CharField(max_length=30)
    last_name = forms.CharField(max_length=30)

    class Meta:
        model = User
        fields = ['username', 'email', 'first_name', 'last_name']

class ProfileUpdateForm(forms.ModelForm):
    class Meta:
        model = Profile
        fields = ['role', 'is_active']

class ProductForm(forms.ModelForm):
    class Meta:
        model = Product
        fields = ['name', 'description', 'price', 'status']

class TransactionForm(forms.ModelForm):
    class Meta:
        model = Transaction
        fields = ['customer_name', 'payment_method']

class TransactionItemForm(forms.ModelForm):
    class Meta:
        model = TransactionItem
        fields = ['product', 'quantity']
```

Step 7: Create Views (pos_app/views.py)

python

```
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth.decorators import login_required, user_passes_test
from django.contrib import messages
from django.db.models import Sum, Count
from django.utils import timezone
from datetime import timedelta
from .models import Profile, Product, Transaction, TransactionItem
from .forms import UserRegisterForm, UserUpdateForm, ProfileUpdateForm,
ProductForm, TransactionForm, TransactionItemForm
from django.contrib.auth.models import User

def role_check(role):
    def check(user):
        try:
            return user.profile.role == role
        except Profile.DoesNotExist:
            return False
    return check

admin_required = user_passes_test(role_check('admin'))
manager_required = user_passes_test(role_check('manager'))
teller_required = user_passes_test(role_check('teller'))

def register(request):
    if request.method == 'POST':
        form = UserRegisterForm(request.POST)
        if form.is_valid():
            user = form.save()
            Profile.objects.create(
                user=user,
                role=form.cleaned_data.get('role')
            )
            messages.success(request, 'Account created successfully!')
            return redirect('login')
    else:
        form = UserRegisterForm()
    return render(request, 'pos_app/register.html', {'form': form})

@login_required
def dashboard(request):
    user = request.user
    context = {}

    if user.profile.role == 'admin':
        # Admin dashboard
        total_users = User.objects.count()
        active_users = User.objects.filter(profile__is_active=True).count()
        context.update({
            'total_users': total_users,
            'active_users': active_users,
        })

    elif user.profile.role == 'manager':
        # Manager dashboard
        today = timezone.now().date()
        week_ago = today - timedelta(days=7)
        month_ago = today - timedelta(days=30)

        daily_sales = Transaction.objects.filter(
            transaction_date__date=today
        ).aggregate(total=Sum('total_amount'))['total'] or 0

        weekly_sales = Transaction.objects.filter(
            transaction_date__date__gte=week_ago
        ).aggregate(total=Sum('total_amount'))['total'] or 0
        monthly_sales = Transaction.objects.filter(
            transaction_date__date__gte=month_ago
        ).aggregate(total=Sum('total_amount'))['total'] or 0

        total_sales = Transaction.objects.aggregate(total=Sum('total_amount'))['total'] or 0

        context.update({
            'daily_sales': daily_sales,
            'weekly_sales': weekly_sales,
            'monthly_sales': monthly_sales,
            'total_sales': total_sales,
        })

    elif user.profile.role == 'teller':
        # Teller dashboard
        today = timezone.now().date()
        today_sales = Transaction.objects.filter(
            teller=user,
            transaction_date__date=today
        ).aggregate(total=Sum('total_amount'))['total'] or 0

        today_transactions = Transaction.objects.filter(
            teller=user,
            transaction_date__date=today
        ).count()

        context.update({
            'today_sales': today_sales,
            'today_transactions': today_transactions,
        })

    return render(request, 'pos_app/dashboard.html', context)

@login_required
@admin_required
def user_management(request):
    users = User.objects.all()
    return render(request, 'pos_app/user_management.html', {'users': users})
```

```
@login_required
@admin_required
def edit_user(request, user_id):
    user = get_object_or_404(User, id=user_id)

    if request.method == 'POST':
        u_form = UserUpdateForm(request.POST, instance=user)
        p_form = ProfileUpdateForm(request.POST, instance=user.profile)

        if u_form.is_valid() and p_form.is_valid():
            u_form.save()
            p_form.save()
            messages.success(request, 'User updated successfully!')
            return redirect('user_management')
    else:
        u_form = UserUpdateForm(instance=user)
        p_form = ProfileUpdateForm(instance=user.profile)

    return render(request, 'pos_app/edit_user.html', {
        'u_form': u_form,
        'p_form': p_form
    })

@login_required
@manager_required
def product_management(request):
    products = Product.objects.all()
    return render(request, 'pos_app/product_management.html', {'products': products})

@login_required
@manager_required
def add_product(request):
    if request.method == 'POST':
        form = ProductForm(request.POST)
        if form.is_valid():
            product = form.save(commit=False)
            product.created_by = request.user
            product.save()
            messages.success(request, 'Product added successfully!')
            return redirect('product_management')
    else:
        form = ProductForm()

    return render(request, 'pos_app/add_product.html', {'form': form})

@login_required
@manager_required
def edit_product(request, product_id):
    product = get_object_or_404(Product, id=product_id)

    if request.method == 'POST':
        form = ProductForm(request.POST, instance=product)
        if form.is_valid():
            form.save()
            messages.success(request, 'Product updated successfully!')
            return redirect('product_management')
    else:
        form = ProductForm(instance=product)

    return render(request, 'pos_app/edit_product.html', {'form': form})

@login_required
@teller_required
def pos_system(request):
    products = Product.objects.filter(status='available')

    if request.method == 'POST':
        transaction_form = TransactionForm(request.POST)

        if transaction_form.is_valid():
            transaction = transaction_form.save(commit=False)
            transaction.teller = request.user

            # Calculate total amount from items
            total_amount = 0
            items = []

            for key, value in request.POST.items():
                if key.startswith('product_'):
                    product_id = key.split('_')[1]
                    quantity = int(value)

                    if quantity > 0:
                        product = Product.objects.get(id=product_id)
                        item_total = product.price * quantity
                        total_amount += item_total

                        items.append({
                            'product': product,
                            'quantity': quantity,
                            'price': product.price
                        })

            transaction.total_amount = total_amount
            transaction.save()

            # Save transaction items
            for item in items:
                TransactionItem.objects.create(
                    transaction=transaction,
                    product=item['product'],
                    quantity=item['quantity'],
                    price=item['price']
                )

            messages.success(request, f'Transaction completed successfully! Total: ${total_amount}')
            return redirect('pos_system')
    else:
        transaction_form = TransactionForm()

    return render(request, 'pos_app/pos_system.html', {
        'products': products,
        'form': transaction_form
    })

@login_required
@teller_required
def today_sales(request):
    today = timezone.now().date()
    transactions = Transaction.objects.filter(
        teller=request.user,
        transaction_date__date=today
    )

    return render(request, 'pos_app/today_sales.html', {'transactions': transactions})

@login_required
@manager_required
def sales_reports(request):
    today = timezone.now().date()
    week_ago = today - timedelta(days=7)
    month_ago = today - timedelta(days=30)

    # Daily sales
    daily_sales = Transaction.objects.filter(
        transaction_date__date=today
    ).values('teller__username').annotate(
        total_sales=Sum('total_amount'),
        transaction_count=Count('id')
    )

    # Weekly sales
    weekly_sales = Transaction.objects.filter(
        transaction_date__date__gte=week_ago
    ).values('teller__username').annotate(
        total_sales=Sum('total_amount'),
        transaction_count=Count('id')
    )

    # Monthly sales
    monthly_sales = Transaction.objects.filter(
        transaction_date__date__gte=month_ago
    ).values('teller__username').annotate(
        total_sales=Sum('total_amount'),
        transaction_count=Count('id')
    )

    # All-time sales
    all_time_sales = Transaction.objects.values('teller__username').annotate(
        total_sales=Sum('total_amount'),
        transaction_count=Count('id')
    )

    context = {
        'daily_sales': daily_sales,
        'weekly_sales': weekly_sales,
        'monthly_sales': monthly_sales,
        'all_time_sales': all_time_sales,
    }

    return render(request, 'pos_app/sales_reports.html', context)
```

Step 8: Create URLs (pos_app/urls.py)

python

```
from django.urls import path
from django.contrib.auth import views as auth_views
from . import views

urlpatterns = [
    path('', auth_views.LoginView.as_view(template_name='pos_app/login.html'),
name='login'),
    path('logout/', auth_views.LogoutView.as_view(), name='logout'),
    path('register/', views.register, name='register'),
    path('dashboard/', views.dashboard, name='dashboard'),

    # Admin URLs
    path('user-management/', views.user_management, name='user_management'),
    path('edit-user/<int:user_id>/', views.edit_user, name='edit_user'),
    # Manager URLs
    path('product-management/', views.product_management, name='product_management'),
    path('add-product/', views.add_product, name='add_product'),
    path('edit-product/<int:product_id>/', views.edit_product, name='edit_product'),
    path('sales-reports/', views.sales_reports, name='sales_reports'),

    # Teller URLs
    path('pos-system/', views.pos_system, name='pos_system'),
    path('today-sales/', views.today_sales, name='today_sales'),
]
```

Step 9: Update Project URLs (pos_project/urls.py)

python

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('pos_app.urls')),
]
```

Step 10: Create Templates

Base Template (templates/pos_app/base.html)

html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>POS System - {% block title %}Home{% endblock %}</title>
    {% load static %}
    <link rel="stylesheet" href="{% static 'pos_app/css/styles.css' %}">
</head>
<body>
    <nav class="navbar">
        <div class="nav-container">
            <div class="nav-logo">
                <img src="{% static 'pos_app/images/logo.png' %}" alt="POS Logo">
                <span>POS System</span>
    </div>
    <div class="nav-menu">
      {% if user.is_authenticated %}
        <a href="{% url 'dashboard' %}">Dashboard</a>
        {% if user.profile.role == 'admin' %}
          <a href="{% url 'user_management' %}">User Management</a>
        {% endif %}
        {% if user.profile.role == 'manager' %}
          <a href="{% url 'product_management' %}">Product Management</a>
          <a href="{% url 'sales_reports' %}">Sales Reports</a>
        {% endif %}
        {% if user.profile.role == 'teller' %}
          <a href="{% url 'pos_system' %}">POS System</a>
          <a href="{% url 'today_sales' %}">Today's Sales</a>
        {% endif %}
        <a href="{% url 'logout' %}">Logout ({{ user.username }})</a>
      {% else %}
        <a href="{% url 'login' %}">Login</a>
        <a href="{% url 'register' %}">Register</a>
      {% endif %}
    </div>
  </div>
</nav>

<div class="container">
  {% if messages %}
    {% for message in messages %}
      <div class="alert alert-{{ message.tags }}">
        {{ message }}
      </div>
    {% endfor %}
  {% endif %}

  {% block content %}
  {% endblock %}
</div>

<script src="{% static 'pos_app/js/pos_app.js' %}"></script>
```

```
</body>
</html>
```

Login Template (templates/pos_app/login.html)

html

```
{% extends 'pos_app/base.html' %}

{% block title %}Login{% endblock %}

{% block content %}
<div class="auth-container">
  <div class="auth-form">
    <h2>Login to POS System</h2>
    <form method="post">
      {% csrf_token %}
      <div class="form-group">
        <label for="username">Username</label>
        <input type="text" id="username" name="username" required>
      </div>
      <div class="form-group">
        <label for="password">Password</label>
        <input type="password" id="password" name="password" required>
      </div>
      <button type="submit" class="btn btn-primary">Login</button>
    </form>
    <p>Don't have an account? <a href="{% url 'register' %}">Register here</a></p>
  </div>
</div>
{% endblock %}
```

Register Template (templates/pos_app/register.html)

html

```
{% extends 'pos_app/base.html' %}

{% block title %}Register{% endblock %}

{% block content %}
<div class="auth-container">
  <div class="auth-form">
    <h2>Register for POS System</h2>
    <form method="post">
      {% csrf_token %}
      {% for field in form %}
        <div class="form-group">
          <label for="{{ field.id_for_label }}">{{ field.label }}</label>
          {{ field }}
          {% if field.errors %}
            <div class="error">{{ field.errors }}</div>
          {% endif %}
        </div>
      {% endfor %}
      <button type="submit" class="btn btn-primary">Register</button>
    </form>
    <p>Already have an account? <a href="{% url 'login' %}">Login here</a></p>
  </div>
</div>
{% endblock %}
```

Dashboard Template (templates/pos_app/dashboard.html)

html

```
{% extends 'pos_app/base.html' %}

{% block title %}Dashboard{% endblock %}

{% block content %}
<div class="dashboard">
  <h2>Welcome, {{ user.username }} ({{ user.profile.role|title }})</h2>

  {% if user.profile.role == 'admin' %}
    <div class="stats-container">
      <div class="stat-card">
        <h3>Total Users</h3>
        <p>{{ total_users }}</p>
      </div>
      <div class="stat-card">
        <h3>Active Users</h3>
        <p>{{ active_users }}</p>
      </div>
    </div>
    {% endif %}

    {% if user.profile.role == 'manager' %}
        <div class="stats-container">
            <div class="stat-card">
                <h3>Daily Sales</h3>
                <p>${{ daily_sales }}</p>
            </div>
            <div class="stat-card">
                <h3>Weekly Sales</h3>
                <p>${{ weekly_sales }}</p>
            </div>
            <div class="stat-card">
                <h3>Monthly Sales</h3>
                <p>${{ monthly_sales }}</p>
            </div>
            <div class="stat-card">
                <h3>Total Sales</h3>
                <p>${{ total_sales }}</p>
            </div>
        </div>
    {% endif %}

    {% if user.profile.role == 'teller' %}
        <div class="stats-container">
            <div class="stat-card">
                <h3>Today's Sales</h3>
                <p>${{ today_sales }}</p>
            </div>
            <div class="stat-card">
                <h3>Today's Transactions</h3>
                <p>{{ today_transactions }}</p>
            </div>
        </div>
    {% endif %}
</div>
{% endblock %}
```

User Management Template (templates/pos_app/user_management.html)

html

```
{% extends 'pos_app/base.html' %}

{% block title %}User Management{% endblock %}

{% block content %}
<div class="management-container">
  <h2>User Management</h2>

  <table class="data-table">
    <thead>
      <tr>
        <th>Username</th>
        <th>Email</th>
        <th>First Name</th>
        <th>Last Name</th>
        <th>Role</th>
        <th>Status</th>
        <th>Actions</th>
      </tr>
    </thead>
    <tbody>
      {% for user in users %}
      <tr>
        <td>{{ user.username }}</td>
        <td>{{ user.email }}</td>
        <td>{{ user.first_name }}</td>
        <td>{{ user.last_name }}</td>
        <td>{{ user.profile.role }}</td>
        <td>{% if user.profile.is_active %}Active{% else %}Inactive{% endif %}</td>
        <td>
          <a href="{% url 'edit_user' user.id %}" class="btn btn-edit">Edit</a>
        </td>
      </tr>
      {% endfor %}
    </tbody>
  </table>
</div>
```

```
{% endblock %}
```

Edit User Template (templates/pos_app/edit_user.html)

html

```
{% extends 'pos_app/base.html' %}

{% block title %}Edit User{% endblock %}

{% block content %}
<div class="form-container">
    <h2>Edit User</h2>

    <form method="post">
        {% csrf_token %}
        <div class="form-grid">
            <div class="form-group">
                <label for="{{ u_form.username.id_for_label }}">Username</label>
                {{ u_form.username }}
            </div>
            <div class="form-group">
                <label for="{{ u_form.email.id_for_label }}">Email</label>
                {{ u_form.email }}
            </div>
            <div class="form-group">
                <label for="{{ u_form.first_name.id_for_label }}">First Name</label>
                {{ u_form.first_name }}
            </div>
            <div class="form-group">
                <label for="{{ u_form.last_name.id_for_label }}">Last Name</label>
                {{ u_form.last_name }}
            </div>
            <div class="form-group">
                <label for="{{ p_form.role.id_for_label }}">Role</label>
                {{ p_form.role }}
            </div>
            <div class="form-group">
                <label for="{{ p_form.is_active.id_for_label }}">Active</label>
                {{ p_form.is_active }}
            </div>
    </div>
    <button type="submit" class="btn btn-primary">Update User</button>
    <a href="{% url 'user_management' %}" class="btn btn-secondary">Cancel</a>
  </form>
</div>
{% endblock %}
```

Product Management Template (templates/pos_app/product_management.html)

html

```
{% extends 'pos_app/base.html' %}

{% block title %}Product Management{% endblock %}

{% block content %}
<div class="management-container">
  <h2>Product Management</h2>
  <a href="{% url 'add_product' %}" class="btn btn-primary">Add New Product</a>

  <table class="data-table">
    <thead>
      <tr>
        <th>Name</th>
        <th>Description</th>
        <th>Price</th>
        <th>Status</th>
        <th>Actions</th>
      </tr>
    </thead>
    <tbody>
      {% for product in products %}
      <tr>
        <td>{{ product.name }}</td>
        <td>{{ product.description }}</td>
        <td>${{ product.price }}</td>
        <td>{{ product.status }}</td>
        <td>
          <a href="{% url 'edit_product' product.id %}" class="btn btn-edit">Edit</a>
        </td>
      </tr>
      {% endfor %}
    </tbody>
  </table>
</div>
{% endblock %}
```

Add/Edit Product Template (templates/pos_app/add_product.html)

html

```
{% extends 'pos_app/base.html' %}

{% block title %}Add Product{% endblock %}

{% block content %}
<div class="form-container">
  <h2>Add Product</h2>

  <form method="post">
    {% csrf_token %}
    <div class="form-grid">
      {% for field in form %}
      <div class="form-group">
        <label for="{{ field.id_for_label }}">{{ field.label }}</label>
        {{ field }}
        {% if field.errors %}
          <div class="error">{{ field.errors }}</div>
        {% endif %}
      </div>
      {% endfor %}
    </div>
    <button type="submit" class="btn btn-primary">Save Product</button>
    <a href="{% url 'product_management' %}" class="btn btn-secondary">Cancel</a>
  </form>
</div>
{% endblock %}
```

POS System Template (templates/pos_app/pos_system.html)

html

```
{% extends 'pos_app/base.html' %}

{% block title %}POS System{% endblock %}
{% block content %}
<div class="pos-container">
  <h2>Point of Sale System</h2>

  <div class="pos-layout">
    <div class="products-section">
      <h3>Available Products</h3>
      <div class="products-grid">
        {% for product in products %}
        <div class="product-card">
          <h4>{{ product.name }}</h4>
          <p>{{ product.description }}</p>
          <p class="price">${{ product.price }}</p>
          <div class="quantity-control">
            <label for="product_{{ product.id }}">Quantity:</label>
            <input type="number" id="product_{{ product.id }}" name="product_{{
product.id }}" min="0" value="0">
          </div>
        </div>
        {% endfor %}
      </div>
    </div>

    <div class="checkout-section">
      <h3>Checkout</h3>
      <form method="post" id="checkout-form">
        {% csrf_token %}
        <div class="form-group">
          <label for="customer_name">Customer Name</label>
          {{ form.customer_name }}
        </div>
        <div class="form-group">
          <label for="payment_method">Payment Method</label>
          {{ form.payment_method }}
        </div>

        <div class="order-summary">
```

```
          <h4>Order Summary</h4>
          <div id="order-items"></div>
          <div class="total-amount">
            <strong>Total: $<span id="total-amount">0.00</span></strong>
          </div>
        </div>

        <button type="submit" class="btn btn-primary btn-large">Complete
Transaction</button>
      </form>
    </div>
  </div>
</div>
{% endblock %}
```

Today's Sales Template (templates/pos_app/today_sales.html)

html

```
{% extends 'pos_app/base.html' %}

{% block title %}Today's Sales{% endblock %}

{% block content %}
<div class="management-container">
  <h2>Today's Sales</h2>

  <table class="data-table">
    <thead>
      <tr>
        <th>Transaction ID</th>
        <th>Customer Name</th>
        <th>Total Amount</th>
        <th>Payment Method</th>
        <th>Date & Time</th>
      </tr>
    </thead>
    <tbody>
      {% for transaction in transactions %}
      <tr>
        <td>#{{ transaction.id }}</td>
        <td>{{ transaction.customer_name }}</td>
        <td>${{ transaction.total_amount }}</td>
        <td>{{ transaction.payment_method }}</td>
        <td>{{ transaction.transaction_date }}</td>
      </tr>
      {% endfor %}
    </tbody>
  </table>
</div>
{% endblock %}
```

Sales Reports Template (templates/pos_app/sales_reports.html)

html

```
{% extends 'pos_app/base.html' %}

{% block title %}Sales Reports{% endblock %}

{% block content %}
<div class="reports-container">
  <h2>Sales Reports</h2>

  <div class="report-section">
    <h3>Daily Sales</h3>
    <table class="data-table">
      <thead>
        <tr>
          <th>Teller</th>
          <th>Total Sales</th>
          <th>Transaction Count</th>
        </tr>
      </thead>
      <tbody>
        {% for sale in daily_sales %}
        <tr>
          <td>{{ sale.teller__username }}</td>
          <td>${{ sale.total_sales }}</td>
          <td>{{ sale.transaction_count }}</td>
        </tr>
        {% endfor %}
        </tbody>
    </table>
</div>

<div class="report-section">
    <h3>Weekly Sales</h3>
    <table class="data-table">
        <thead>
            <tr>
                <th>Teller</th>
                <th>Total Sales</th>
                <th>Transaction Count</th>
            </tr>
        </thead>
        <tbody>
            {% for sale in weekly_sales %}
            <tr>
                <td>{{ sale.teller__username }}</td>
                <td>${{ sale.total_sales }}</td>
                <td>{{ sale.transaction_count }}</td>
            </tr>
            {% endfor %}
        </tbody>
    </table>
</div>

<div class="report-section">
    <h3>Monthly Sales</h3>
    <table class="data-table">
        <thead>
            <tr>
                <th>Teller</th>
                <th>Total Sales</th>
                <th>Transaction Count</th>
            </tr>
        </thead>
        <tbody>
            {% for sale in monthly_sales %}
        <tr>
          <td>{{ sale.teller__username }}</td>
          <td>${{ sale.total_sales }}</td>
          <td>{{ sale.transaction_count }}</td>
        </tr>
        {% endfor %}
      </tbody>
    </table>
  </div>

  <div class="report-section">
    <h3>All-Time Sales</h3>
    <table class="data-table">
      <thead>
        <tr>
          <th>Teller</th>
          <th>Total Sales</th>
          <th>Transaction Count</th>
        </tr>
      </thead>
      <tbody>
        {% for sale in all_time_sales %}
        <tr>
          <td>{{ sale.teller__username }}</td>
          <td>${{ sale.total_sales }}</td>
          <td>{{ sale.transaction_count }}</td>
        </tr>
        {% endfor %}
      </tbody>
    </table>
  </div>
</div>
{% endblock %}
```

Step 11: Create Static Files

CSS (static/pos_app/css/styles.css)

css

```
/* Reset and base styles */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  line-height: 1.6;
  color: #333;
  background-color: #f4f4f4;
}

/* Navigation */
.navbar {
  background-color: #2c3e50;
  color: white;
  padding: 1rem 0;
}

.nav-container {
  width: 90%;
  max-width: 1200px;
  margin: 0 auto;
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.nav-logo {
  display: flex;
  align-items: center;
  gap: 10px;
}

.nav-logo img {
  height: 40px;
}
```

```
.nav-menu {
   display: flex;
   gap: 20px;
}

.nav-menu a {
   color: white;
   text-decoration: none;
   padding: 5px 10px;
   border-radius: 4px;
   transition: background-color 0.3s;
}

.nav-menu a:hover {
   background-color: #34495e;
}

/* Container */
.container {
   width: 90%;
   max-width: 1200px;
   margin: 2rem auto;
   padding: 1rem;
}

/* Auth forms */
.auth-container {
   display: flex;
   justify-content: center;
   align-items: center;
   min-height: 70vh;
}

.auth-form {
   background: white;
   padding: 2rem;
   border-radius: 8px;
   box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
```

```css
  width: 100%;
  max-width: 400px;
}

.auth-form h2 {
  margin-bottom: 1.5rem;
  text-align: center;
  color: #2c3e50;
}

/* Forms */
.form-group {
  margin-bottom: 1rem;
}

.form-group label {
  display: block;
  margin-bottom: 0.5rem;
  font-weight: 600;
}

.form-group input,
.form-group select,
.form-group textarea {
  width: 100%;
  padding: 0.75rem;
  border: 1px solid #ddd;
  border-radius: 4px;
  font-size: 1rem;
}

.form-grid {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
  gap: 1rem;
}

/* Buttons */
```

```css
.btn {
  display: inline-block;
  padding: 0.75rem 1.5rem;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  text-decoration: none;
  font-size: 1rem;
  transition: background-color 0.3s;
}

.btn-primary {
  background-color: #3498db;
  color: white;
}

.btn-primary:hover {
  background-color: #2980b9;
}

.btn-secondary {
  background-color: #95a5a6;
  color: white;
}

.btn-secondary:hover {
  background-color: #7f8c8d;
}

.btn-edit {
  background-color: #f39c12;
  color: white;
}

.btn-edit:hover {
  background-color: #e67e22;
}
```

```
.btn-large {
  padding: 1rem 2rem;
  font-size: 1.1rem;
}

/* Alerts */
.alert {
  padding: 1rem;
  margin-bottom: 1rem;
  border-radius: 4px;
  border: 1px solid transparent;
}

.alert-success {
  color: #155724;
  background-color: #d4edda;
  border-color: #c3e6cb;
}

.alert-error {
  color: #721c24;
  background-color: #f8d7da;
  border-color: #f5c6cb;
}

/* Dashboard */
.dashboard h2 {
  margin-bottom: 2rem;
  color: #2c3e50;
}

.stats-container {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
  gap: 1.5rem;
}

.stat-card {
  background: white;
  padding: 1.5rem;
  border-radius: 8px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
  text-align: center;
}

.stat-card h3 {
  color: #7f8c8d;
  margin-bottom: 0.5rem;
  font-size: 1rem;
}

.stat-card p {
  font-size: 2rem;
  font-weight: bold;
  color: #2c3e50;
}

/* Management containers */
.management-container,
.form-container,
.reports-container {
  background: white;
  padding: 2rem;
  border-radius: 8px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
}

.management-container h2,
.form-container h2,
.reports-container h2 {
  margin-bottom: 1.5rem;
  color: #2c3e50;
}

/* Data tables */
.data-table {
  width: 100%;
  border-collapse: collapse;
  margin-top: 1rem;
}

.data-table th,
.data-table td {
  padding: 0.75rem;
  text-align: left;
  border-bottom: 1px solid #ddd;
}

.data-table th {
  background-color: #f8f9fa;
  font-weight: 600;
}

.data-table tr:hover {
  background-color: #f8f9fa;
}

/* POS System */
.pos-container {
  background: white;
  padding: 2rem;
  border-radius: 8px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
}

.pos-layout {
  display: grid;
  grid-template-columns: 2fr 1fr;
  gap: 2rem;
  margin-top: 1.5rem;
}

.products-section h3,
.checkout-section h3 {
  margin-bottom: 1rem;
  color: #2c3e50;
}

.products-grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
  gap: 1rem;
}

.product-card {
  border: 1px solid #ddd;
  border-radius: 8px;
  padding: 1rem;
  text-align: center;
}

.product-card h4 {
  margin-bottom: 0.5rem;
  color: #2c3e50;
}

.product-card .price {
  font-weight: bold;
  color: #27ae60;
  margin: 0.5rem 0;
}

.quantity-control {
  margin-top: 1rem;
}

.quantity-control label {
  display: block;
  margin-bottom: 0.25rem;
}

.quantity-control input {
  width: 80px;
  text-align: center;
}

.checkout-section {
  border-left: 1px solid #ddd;
  padding-left: 2rem;
}

.order-summary {
  margin: 1.5rem 0;
  padding: 1rem;
  border: 1px solid #ddd;
  border-radius: 8px;
}

.order-summary h4 {
  margin-bottom: 1rem;
  color: #2c3e50;
}

.total-amount {
  margin-top: 1rem;
  padding-top: 1rem;
  border-top: 2px solid #ddd;
  text-align: right;
  font-size: 1.2rem;
}

/* Reports */
.report-section {
  margin-bottom: 2rem;
}

.report-section h3 {
  margin-bottom: 1rem;
  color: #2c3e50;
}
```

```
/* Error messages */
.error {
  color: #e74c3c;
  font-size: 0.875rem;
  margin-top: 0.25rem;
}

/* Responsive design */
@media (max-width: 768px) {
  .nav-container {
    flex-direction: column;
    gap: 1rem;
  }

  .pos-layout {
    grid-template-columns: 1fr;
  }

  .checkout-section {
    border-left: none;
    border-top: 1px solid #ddd;
    padding-left: 0;
    padding-top: 2rem;
  }

  .form-grid {
    grid-template-columns: 1fr;
  }
}
```

JavaScript (static/pos_app/js/pos_app.js)

javascript

```
document.addEventListener('DOMContentLoaded', function() {
  // POS System functionality
  if (document.querySelector('.pos-container')) {
    initPOSSystem();
  }
  // Form validation
  const forms = document.querySelectorAll('form');
  forms.forEach(form => {
    form.addEventListener('submit', function(e) {
      const requiredFields = form.querySelectorAll('[required]');
      let valid = true;

      requiredFields.forEach(field => {
        if (!field.value.trim()) {
          valid = false;
          field.style.borderColor = '#e74c3c';
        } else {
          field.style.borderColor = '';
        }
      });

      if (!valid) {
        e.preventDefault();
        alert('Please fill in all required fields.');
      }
    });
  });
});

function initPOSSystem() {
  const quantityInputs = document.querySelectorAll('.quantity-control input');
  const orderItems = document.getElementById('order-items');
  const totalAmount = document.getElementById('total-amount');
  const checkoutForm = document.getElementById('checkout-form');

  let cart = {};

  // Initialize cart
  quantityInputs.forEach(input => {
    const productId = input.name.split('_')[1];
    cart[productId] = 0;

    input.addEventListener('change', function() {
      updateCart(productId, parseInt(this.value) || 0);
      updateOrderSummary();
    });
  });

  function updateCart(productId, quantity) {
    cart[productId] = quantity;
  }

  function updateOrderSummary() {
    // Clear previous items
    orderItems.innerHTML = '';
    let total = 0;

    // Add items to summary
    for (const productId in cart) {
      if (cart[productId] > 0) {
        const productElement = document.querySelector(` .product-card
input[name="product_${productId}"]` ).closest('.product-card');
        const productName = productElement.querySelector('h4').textContent;
        const productPrice =
parseFloat(productElement.querySelector('.price').textContent.replace('$', ''));
        const itemTotal = productPrice * cart[productId];

        total += itemTotal;

        const itemElement = document.createElement('div');
        itemElement.className = 'order-item';
        itemElement.innerHTML = `
          <div>${productName} x ${cart[productId]}</div>
          <div>$${itemTotal.toFixed(2)}</div>
        `;

        orderItems.appendChild(itemElement);
      }
    }

    // Update total
    totalAmount.textContent = total.toFixed(2);

    // Add hidden fields for products in cart
    document.querySelectorAll('input[type="hidden"][name^="product_"]').forEach(input => {
      input.remove();
    });

    for (const productId in cart) {
      if (cart[productId] > 0) {
        const hiddenInput = document.createElement('input');
        hiddenInput.type = 'hidden';
        hiddenInput.name = `product_${productId}`;
        hiddenInput.value = cart[productId];
        checkoutForm.appendChild(hiddenInput);
      }
    }
  }

  // Initial update
  updateOrderSummary();
}
```

Step 12: Create Admin Interface (pos_app/admin.py)

python

```
from django.contrib import admin
from .models import Profile, Product, Transaction, TransactionItem

@admin.register(Profile)
class ProfileAdmin(admin.ModelAdmin):
  list_display = ['user', 'role', 'is_active', 'created_at']
  list_filter = ['role', 'is_active']
  search_fields = ['user__username', 'user__email']

@admin.register(Product)
class ProductAdmin(admin.ModelAdmin):
  list_display = ['name', 'price', 'status', 'created_by', 'created_at']
  list_filter = ['status']
  search_fields = ['name', 'description']
```

```
@admin.register(Transaction)
class TransactionAdmin(admin.ModelAdmin):
    list_display = ['id', 'teller', 'customer_name', 'total_amount', 'payment_method', 'transaction_date']
    list_filter = ['payment_method', 'transaction_date']
    search_fields = ['customer_name', 'teller__username']

@admin.register(TransactionItem)
class TransactionItemAdmin(admin.ModelAdmin):
    list_display = ['transaction', 'product', 'quantity', 'price']
    list_filter = ['product']
```

Step 13: Create Signals for Profile (pos_app/signals.py)

python

```
from django.db.models.signals import post_save
from django.contrib.auth.models import User
from django.dispatch import receiver
from .models import Profile

@receiver(post_save, sender=User)
def create_user_profile(sender, instance, created, **kwargs):
    if created:
        Profile.objects.create(user=instance)

@receiver(post_save, sender=User)
def save_user_profile(sender, instance, **kwargs):
    try:
        instance.profile.save()
    except Profile.DoesNotExist:
        Profile.objects.create(user=instance)
```

Step 14: Update App Config (pos_app/apps.py)

python

```
from django.apps import AppConfig

class PosAppConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'pos_app'
    def ready(self):
        import pos_app.signals
```

Step 15: Create Requirements File

bash

```
# Create requirements.txt
pip freeze > requirements.txt
```

Step 16: Run Migrations and Create Superuser

bash

```
# Run migrations
python manage.py makemigrations
python manage.py migrate

# Create superuser
python manage.py createsuperuser
```

Step 17: Run the Development Server

bash

```
python manage.py runserver
```

Step 18: Access the Application

1. Open your browser and go to http://127.0.0.1:8000/
2. Login with your superuser credentials
3. Create users with different roles through the admin interface
4. Test the functionality for each role

Additional Notes

1. Make sure to add a logo image at static/pos_app/images/logo.png
2. For production, change the SECRET_KEY and set DEBUG to False
3. You may want to add more security measures and error handling
4. Consider adding more features like receipts printing, inventory management, etc.