NoteEve Django - Complete Project Guide

Table of Contents

PART 1: COMPLETE FILE STRUCTURE

```
noteeve_django/
├── manage.py
├── requirements.txt
├── .env.example
├── README.md
├── noteeve/
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   ├── wsgi.py
│   ├── asgi.py
│   └── celery.py
├── notes/
│   ├── __init__.py
│   ├── models.py
│   ├── views.py
│   ├── forms.py
│   ├── urls.py
│   ├── admin.py
│   ├── apps.py
│   ├── migrations/
│   │   └── __init__.py
│   └── templates/notes/
│       ├── login.html
│       ├── register.html
│       ├── dashboard.html
│       ├── subject_list.html
│       ├── subject_detail.html
│       ├── subject_form.html
│       ├── topic_form.html
│       ├── note_view.html
│       ├── note_form.html
│       ├── note_confirm_delete.html
│       ├── task_list.html
│       ├── task_form.html
│       ├── bookmark_list.html
│       ├── collaboration.html
│       ├── pdf_compile.html
│       ├── summarizer.html
│       └── summary_view.html
├── templates/
│   └── base.html
├── static/
│   ├── css/
│   │   ├── style.css
│   │   └── tinymce.css
│   └── js/
│       └── main.js
├── media/
│   └── notes/
└── venv/
```

PART 2: COMPLETE CODE FOR ALL FILES

ROOT LEVEL FILES

1. manage.py

```
#!/usr/bin/env python
import os
import sys

def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'noteeve.settings.development')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError("Couldn't import Django") from exc
    execute_from_command_line(sys.argv)

if __name__ == '__main__':
    main()
```

2. requirements.txt

```
Django==5.1.5
psycopg2-binary==2.9.9
celery==5.4.0
redis==5.0.1
django-tinymce==4.1.0
reportlab==4.2.5
PyPDF2==3.0.1
pdfminer.six==20231228
transformers==4.45.2
torch==2.5.1
pillow==10.4.0
python-dotenv==1.0.1
gunicorn==23.0.0
whitenoise==6.7.0
django-celery-beat==2.7.0
```

```
django-celery-results==2.5.1
requests==2.32.3
```

3. .env.example

```
DEBUG=True
SECRET_KEY=django-insecure-your-secret-key-here-change-in-production
ALLOWED_HOSTS=localhost,127.0.0.1

DB_ENGINE=django.db.backends.postgresql
DB_NAME=noteeve_db
DB_USER=noteeve_user
DB_PASSWORD=your_password
DB_HOST=localhost
DB_PORT=5432

CELERY_BROKER_URL=redis://localhost:6379/0
CELERY_RESULT_BACKEND=redis://localhost:6379/0

EMAIL_BACKEND=django.core.mail.backends.console.EmailBackend
EMAIL_HOST=smtp.gmail.com
EMAIL_PORT=587
EMAIL_USE_TLS=True
EMAIL_HOST_USER=your_email@gmail.com
EMAIL_HOST_PASSWORD=your_password
```

NOTEEVE PROJECT CONFIGURATION

4. noteeve/init.py

```
# Empty file - marks directory as Python package
```

5. noteeve/settings.py

```
import os
from pathlib import Path
from dotenv import load_dotenv

load_dotenv()

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

SECRET_KEY = os.getenv('SECRET_KEY', 'django-insecure-your-secret-key')
DEBUG = os.getenv('DEBUG', 'True') == 'True'
ALLOWED_HOSTS = os.getenv('ALLOWED_HOSTS', 'localhost,127.0.0.1').split(',')

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'tinymce',
    'django_celery_beat',
    'django_celery_results',
    'notes',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'whitenoise.middleware.WhiteNoiseMiddleware',
    '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 = 'noteeve.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [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 = 'noteeve.wsgi.application'

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': os.getenv('DB_NAME', 'noteeve_db'),
        'USER': os.getenv('DB_USER', 'noteeve_user'),
        'PASSWORD': os.getenv('DB_PASSWORD', 'password'),
        'HOST': os.getenv('DB_HOST', 'localhost'),
        'PORT': os.getenv('DB_PORT', '5432'),
    }
}

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/'
STATIC_ROOT = BASE_DIR / 'staticfiles'
STATICFILES_DIRS = [BASE_DIR / 'static']
STATICFILES_STORAGE = 'whitenoise.storage.CompressedManifestStaticFilesStorage'

MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR / 'media'

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
AUTH_USER_MODEL = 'notes.CustomUser'

CELERY_BROKER_URL = os.getenv('CELERY_BROKER_URL', 'redis://localhost:6379/0')
CELERY_RESULT_BACKEND = os.getenv('CELERY_RESULT_BACKEND', 'redis://localhost:6379/0')
CELERY_ACCEPT_CONTENT = ['json']
CELERY_TASK_SERIALIZER = 'json'
CELERY_RESULT_SERIALIZER = 'json'
CELERY_TIMEZONE = 'UTC'

TINYMCE_DEFAULT_CONFIG = {
    'height': 400,
    'width': '100%',
    'cleanup_on_startup': True,
    'custom_undo_redo_levels': 20,
    'selector': 'textarea',
    'plugins': 'textcolor save link image media preview codesample contextmenu table code
    'toolbar1': 'fullscreen preview bold italic underline | fontselect fontsizeselect | f
    'toolbar2': 'undo redo | removeformat',
    'content_css': '/static/css/tinymce.css',
}

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

6. noteeve/urls.py

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static

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

```
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
    urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

7. noteeve/celery.py

```
import os
from celery import Celery
from django.conf import settings

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'noteeve.settings')

app = Celery('noteeve')
app.config_from_object('django.conf:settings', namespace='CELERY')
app.autodiscover_tasks()

@app.task(bind=True)
def debug_task(self):
    print(f'Request: {self.request!r}')
```

8. noteeve/wsgi.py

```
import os
from django.core.wsgi import get_wsgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'noteeve.settings')
application = get_wsgi_application()
```

9. noteeve/asgi.py

```
import os
from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'noteeve.settings')
application = get_asgi_application()
```

NOTES APP - CORE FILES

10. notes/init.py

```
default_app_config = "notes.apps.NotesConfig"
```

11. notes/apps.py

```
from django.apps import AppConfig

class NotesConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'notes'
```

12. notes/models.py

```
from django.db import models
from django.contrib.auth.models import AbstractUser
from django.core.validators import MinValueValidator, MaxValueValidator

class CustomUser(AbstractUser):
    ROLE_CHOICES = [
        ('provider', 'Content Provider'),
        ('student', 'Student'),
    ]
    role = models.CharField(max_length=10, choices=ROLE_CHOICES, default='student')
    created_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        db_table = 'users'
        verbose_name = 'User'
        verbose_name_plural = 'Users'

    def __str__(self):
        return f"{self.username} ({self.get_role_display()})"

class Subject(models.Model):
    name = models.CharField(max_length=255)
    description = models.TextField(blank=True)
    owner = models.ForeignKey(CustomUser, on_delete=models.CASCADE, related_name='subject
    collaborators = models.ManyToManyField(CustomUser, through='Collaboration', related_n
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    class Meta:
        db_table = 'subjects'
        ordering = ['-created_at']
        verbose_name = 'Subject'
        verbose_name_plural = 'Subjects'

    def __str__(self):
        return self.name

class Topic(models.Model):
    name = models.CharField(max_length=255)
    description = models.TextField(blank=True)
    subject = models.ForeignKey(Subject, on_delete=models.CASCADE, related_name='topics')
    order = models.IntegerField(default=0)
    created_at = models.DateTimeField(auto_now_add=True)
    class Meta:
        db_table = 'topics'
        ordering = ['order', 'created_at']
        verbose_name = 'Topic'
        verbose_name_plural = 'Topics'

    def __str__(self):
        return f"{self.subject.name} &gt; {self.name}"

class Note(models.Model):
    title = models.CharField(max_length=255)
    content = models.TextField()
    file_upload = models.FileField(upload_to='notes/', blank=True, null=True)
    topic = models.ForeignKey(Topic, on_delete=models.CASCADE, related_name='notes')
    owner = models.ForeignKey(CustomUser, on_delete=models.CASCADE, related_name='notes')
    is_public = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    class Meta:
        db_table = 'notes'
        ordering = ['-created_at']
        verbose_name = 'Note'
        verbose_name_plural = 'Notes'

    def __str__(self):
        return f"{self.title} by {self.owner.username}"

class Collaboration(models.Model):
    PERMISSION_CHOICES = [
        ('view', 'View Only'),
        ('edit', 'Can Edit'),
    ]
    subject = models.ForeignKey(Subject, on_delete=models.CASCADE)
    user = models.ForeignKey(CustomUser, on_delete=models.CASCADE)
    permission_level = models.CharField(max_length=10, choices=PERMISSION_CHOICES, defaul
    created_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        db_table = 'collaborations'
        unique_together = ['subject', 'user']
        verbose_name = 'Collaboration'
        verbose_name_plural = 'Collaborations'

    def __str__(self):
        return f"{self.user.username} - {self.subject.name} ({self.get_permission_level_d

class Bookmark(models.Model):
    user = models.ForeignKey(CustomUser, on_delete=models.CASCADE, related_name='bookmark
    note = models.ForeignKey(Note, on_delete=models.CASCADE, related_name='bookmarks')
    page_position = models.IntegerField(default=0)
    created_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        db_table = 'bookmarks'
        unique_together = ['user', 'note']
        verbose_name = 'Bookmark'
        verbose_name_plural = 'Bookmarks'

    def __str__(self):
        return f"{self.user.username} - {self.note.title}"

class Task(models.Model):
    title = models.CharField(max_length=255)
    description = models.TextField(blank=True)
    user = models.ForeignKey(CustomUser, on_delete=models.CASCADE, related_name='tasks')
    due_date = models.DateTimeField(null=True, blank=True)
    completed = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        db_table = 'tasks'
        ordering = ['due_date', '-created_at']
        verbose_name = 'Task'
        verbose_name_plural = 'Tasks'

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

class Progress(models.Model):
    user = models.ForeignKey(CustomUser, on_delete=models.CASCADE)
    note = models.ForeignKey(Note, on_delete=models.CASCADE, null=True, blank=True)
    topic = models.ForeignKey(Topic, on_delete=models.CASCADE, null=True, blank=True)
    completion_percentage = models.IntegerField(default=0, validators=[MinValueValidator(
    last_updated = models.DateTimeField(auto_now=True)

    class Meta:
        db_table = 'progress'
        verbose_name = 'Progress'
        verbose_name_plural = 'Progress'

    def __str__(self):
        return f"{self.user.username} - {self.completion_percentage}%"

class Summary(models.Model):
    note = models.OneToOneField(Note, on_delete=models.CASCADE, related_name='summary')
    summary_text = models.TextField()
    keywords = models.JSONField(default=list)
    created_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        db_table = 'summaries'
        verbose_name = 'Summary'
        verbose_name_plural = 'Summaries'

    def __str__(self):
        return f"Summary - {self.note.title}"
```

13. notes/forms.py

```
from django import forms
from django.contrib.auth.forms import UserCreationForm
from tinymce.widgets import TinyMCE
from .models import CustomUser, Subject, Topic, Note, Task, Collaboration

class CustomUserCreationForm(UserCreationForm):
    email = forms.EmailField(required=True)
    role = forms.ChoiceField(choices=CustomUser.ROLE_CHOICES)

    class Meta:
        model = CustomUser
        fields = ('username', 'email', 'role', 'password1', 'password2')

class SubjectForm(forms.ModelForm):
    class Meta:
        model = Subject
        fields = ('name', 'description')
        widgets = {
            'name': forms.TextInput(attrs={'class': 'form-control'}),
            'description': forms.Textarea(attrs={'class': 'form-control', 'rows': 4}),
        }

class TopicForm(forms.ModelForm):
    class Meta:
        model = Topic
        fields = ('name', 'description', 'order')
        widgets = {
            'name': forms.TextInput(attrs={'class': 'form-control'}),
            'description': forms.Textarea(attrs={'class': 'form-control', 'rows': 3}),
            'order': forms.NumberInput(attrs={'class': 'form-control'}),
        }

class NoteForm(forms.ModelForm):
    class Meta:
        model = Note
        fields = ('title', 'content', 'file_upload', 'is_public')
        widgets = {
            'title': forms.TextInput(attrs={'class': 'form-control'}),
            'content': TinyMCE(attrs={'cols': 80, 'rows': 30}),
            'file_upload': forms.FileInput(attrs={'class': 'form-control'}),
            'is_public': forms.CheckboxInput(attrs={'class': 'form-check-input'}),
        }

class TaskForm(forms.ModelForm):
    class Meta:
        model = Task
        fields = ('title', 'description', 'due_date', 'completed')
        widgets = {
            'title': forms.TextInput(attrs={'class': 'form-control'}),
            'description': forms.Textarea(attrs={'class': 'form-control', 'rows': 3}),
            'due_date': forms.DateTimeInput(attrs={'class': 'form-control', 'type': 'date
            'completed': forms.CheckboxInput(attrs={'class': 'form-check-input'}),
        }
```

14. notes/views.py

```
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.decorators import login_required
from django.contrib import messages
from django.http import JsonResponse, HttpResponse
from django.views.decorators.http import require_POST
from .models import (CustomUser, Subject, Topic, Note, Bookmark, Task,
                     Progress, Collaboration, Summary)
from .forms import (CustomUserCreationForm, SubjectForm, TopicForm, NoteForm,
                    TaskForm)

# ============= AUTH VIEWS =============
def home(request):
    if request.user.is_authenticated:
        return redirect('dashboard')
    return redirect('login')

def register_view(request):
    if request.method == 'POST':
        form = CustomUserCreationForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)
            messages.success(request, 'Registration successful!')
            return redirect('dashboard')
    else:
        form = CustomUserCreationForm()
    return render(request, 'notes/register.html', {'form': form})

def login_view(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            return redirect('dashboard')
        else:
            messages.error(request, 'Invalid credentials')
    return render(request, 'notes/login.html')

def logout_view(request):
    logout(request)
    messages.success(request, 'Logged out successfully!')
    return redirect('login')

# ============= DASHBOARD =============
@login_required
def dashboard(request):
    user = request.user
    subjects = Subject.objects.filter(owner=user) | Subject.objects.filter(collaboration_
    subjects = subjects.distinct()

    tasks = Task.objects.filter(user=user, completed=False).order_by('due_date')[:5]
    bookmarks = Bookmark.objects.filter(user=user)[:5]
    recent_notes = Note.objects.filter(owner=user).order_by('-created_at')[:5]

    total_notes = Note.objects.filter(owner=user).count()
    completed_tasks = Task.objects.filter(user=user, completed=True).count()

    context = {
        'subjects': subjects,
        'tasks': tasks,
        'bookmarks': bookmarks,
        'recent_notes': recent_notes,
        'total_notes': total_notes,
        'completed_tasks': completed_tasks,
    }
    return render(request, 'notes/dashboard.html', context)

# ============= SUBJECT VIEWS =============
@login_required
def subject_list(request):
    subjects = Subject.objects.filter(owner=request.user)
    shared_subjects = Subject.objects.filter(collaboration__user=request.user).distinct()
    return render(request, 'notes/subject_list.html', {'subjects': subjects, 'shared_sub

@login_required
def subject_create(request):
    if request.method == 'POST':
        form = SubjectForm(request.POST)
        if form.is_valid():
            subject = form.save(commit=False)
            subject.owner = request.user
            subject.save()
            messages.success(request, 'Subject created!')
            return redirect('subject_detail', pk=subject.pk)
    else:
        form = SubjectForm()
    return render(request, 'notes/subject_form.html', {'form': form})

@login_required
def subject_detail(request, pk):
    subject = get_object_or_404(Subject, pk=pk)
    if subject.owner != request.user:
        if not Collaboration.objects.filter(subject=subject, user=request.user).exists():
            messages.error(request, 'Access denied')
            return redirect('subject_list')

    topics = subject.topics.all()
    return render(request, 'notes/subject_detail.html', {'subject': subject, 'topics': to

@login_required
def subject_edit(request, pk):
    subject = get_object_or_404(Subject, pk=pk, owner=request.user)
    if request.method == 'POST':
        form = SubjectForm(request.POST, instance=subject)
        if form.is_valid():
            form.save()
            messages.success(request, 'Subject updated!')
            return redirect('subject_detail', pk=pk)
    else:
        form = SubjectForm(instance=subject)
    return render(request, 'notes/subject_form.html', {'form': form, 'subject': subject})

@login_required
def subject_delete(request, pk):
    subject = get_object_or_404(Subject, pk=pk, owner=request.user)
    if request.method == 'POST':
        subject.delete()
        messages.success(request, 'Subject deleted!')
        return redirect('subject_list')
    return render(request, 'notes/subject_confirm_delete.html', {'subject': subject})

# ============= TOPIC VIEWS =============
@login_required
def topic_create(request, subject_id):
    subject = get_object_or_404(Subject, pk=subject_id, owner=request.user)
    if request.method == 'POST':
        form = TopicForm(request.POST)
        if form.is_valid():
            topic = form.save(commit=False)
            topic.subject = subject
            topic.save()
            messages.success(request, 'Topic created!')
            return redirect('subject_detail', pk=subject_id)
    else:
        form = TopicForm()
    return render(request, 'notes/topic_form.html', {'form': form, 'subject': subject})

@login_required
def topic_edit(request, pk):
    topic = get_object_or_404(Topic, pk=pk, subject__owner=request.user)
    if request.method == 'POST':
        form = TopicForm(request.POST, instance=topic)
        if form.is_valid():
            form.save()
            messages.success(request, 'Topic updated!')
            return redirect('subject_detail', pk=topic.subject.pk)
    else:
        form = TopicForm(instance=topic)
    return render(request, 'notes/topic_form.html', {'form': form, 'topic': topic})

# ============= NOTE VIEWS =============
@login_required
def note_create(request, topic_id):
    topic = get_object_or_404(Topic, pk=topic_id, subject__owner=request.user)
    if request.method == 'POST':
        form = NoteForm(request.POST, request.FILES)
        if form.is_valid():
            note = form.save(commit=False)
            note.topic = topic
            note.owner = request.user
            note.save()
            messages.success(request, 'Note created!')
            return redirect('note_view', pk=note.pk)
    else:
        form = NoteForm()
    return render(request, 'notes/note_form.html', {'form': form, 'topic': topic})

@login_required
def note_view(request, pk):
    note = get_object_or_404(Note, pk=pk)
    if note.owner != request.user and not note.is_public:
        if not Collaboration.objects.filter(subject=note.topic.subject, user=request.user
            messages.error(request, 'Access denied')
            return redirect('subject_list')

    bookmarked = Bookmark.objects.filter(user=request.user, note=note).exists()
    return render(request, 'notes/note_view.html', {'note': note, 'bookmarked': bookmarke

@login_required
def note_edit(request, pk):
    note = get_object_or_404(Note, pk=pk, owner=request.user)
    if request.method == 'POST':
        form = NoteForm(request.POST, request.FILES, instance=note)
        if form.is_valid():
            form.save()
            messages.success(request, 'Note updated!')
            return redirect('note_view', pk=pk)
    else:
        form = NoteForm(instance=note)
    return render(request, 'notes/note_form.html', {'form': form, 'note': note})

@login_required
def note_delete(request, pk):
    note = get_object_or_404(Note, pk=pk, owner=request.user)
    if request.method == 'POST':
        note.delete()
        messages.success(request, 'Note deleted!')
        return redirect('subject_detail', pk=note.topic.subject.pk)
    return render(request, 'notes/note_confirm_delete.html', {'note': note})

# ============= BOOKMARK VIEWS =============
@login_required
def bookmark_list(request):
    bookmarks = Bookmark.objects.filter(user=request.user).select_related('note')
    return render(request, 'notes/bookmark_list.html', {'bookmarks': bookmarks})

@login_required
@require_POST
def bookmark_toggle(request, note_id):
    note = get_object_or_404(Note, pk=note_id)
    bookmark, created = Bookmark.objects.get_or_create(user=request.user, note=note)
    if not created:
        bookmark.delete()
        return JsonResponse({'status': 'removed'})
    return JsonResponse({'status': 'added'})

# ============= TASK VIEWS =============
@login_required
def task_list(request):
    tasks = Task.objects.filter(user=request.user).order_by('due_date')
    return render(request, 'notes/task_list.html', {'tasks': tasks})

@login_required
def task_create(request):
    if request.method == 'POST':
        form = TaskForm(request.POST)
        if form.is_valid():
            task = form.save(commit=False)
            task.user = request.user
            task.save()
            messages.success(request, 'Task created!')
            return redirect('task_list')
    else:
        form = TaskForm()
    return render(request, 'notes/task_form.html', {'form': form})

@login_required
def task_complete(request, pk):
    task = get_object_or_404(Task, pk=pk, user=request.user)
    task.completed = not task.completed
    task.save()
    messages.success(request, 'Task updated!')
    return redirect('task_list')

# ============= COLLABORATION VIEWS =============
@login_required
def collaboration_manage(request, subject_id):
    subject = get_object_or_404(Subject, pk=subject_id, owner=request.user)
    collaborations = Collaboration.objects.filter(subject=subject)

    if request.method == 'POST':
        email = request.POST.get('collaborator_email')
        permission = request.POST.get('permission_level', 'view')
        try:
            user = CustomUser.objects.get(email=email)
            collab, created = Collaboration.objects.get_or_create(
                subject=subject, user=user,
                defaults={'permission_level': permission}
            )
            if not created:
                collab.permission_level = permission
                collab.save()
            messages.success(request, f'Shared with {user.username}!')
        except CustomUser.DoesNotExist:
            messages.error(request, 'User not found')
        return redirect('collaboration_manage', subject_id=subject_id)

    return render(request, 'notes/collaboration.html', {'subject': subject, 'collaboratic

# ============= PDF COMPILER =============
@login_required
def pdf_compile(request):
    if request.method == 'POST':
        note_ids = request.POST.getlist('note_ids')
        notes = Note.objects.filter(pk__in=note_ids, owner=request.user)
        if not notes:
            messages.error(request, 'No notes selected')
            return redirect('subject_list')

        try:
            from reportlab.lib.pagesizes import letter
            from reportlab.pdfgen import canvas
            from io import BytesIO

            buffer = BytesIO()
            pdf_canvas = canvas.Canvas(buffer, pagesize=letter)

            y_position = 750
            for note in notes:
                pdf_canvas.setFont("Helvetica-Bold", 16)
                pdf_canvas.drawString(50, y_position, note.title)
                y_position -= 30

                pdf_canvas.setFont("Helvetica", 10)
                pdf_canvas.drawString(50, y_position, f"Subject: {note.topic.subject.name
                y_position -= 20

                pdf_canvas.drawString(50, y_position, f"Topic: {note.topic.name}")
                y_position -= 30

            pdf_canvas.save()
            buffer.seek(0)

            response = HttpResponse(buffer, content_type='application/pdf')
            response['Content-Disposition'] = 'attachment; filename="compiled_notes.pdf"'
            return response
        except Exception as e:
            messages.error(request, f'PDF generation failed: {str(e)}')

    subjects = Subject.objects.filter(owner=request.user)
    return render(request, 'notes/pdf_compile.html', {'subjects': subjects})

# ============= AI SUMMARIZER =============
@login_required
def summarizer(request):
    return render(request, 'notes/summarizer.html')

@login_required
def summarize_note(request, note_id):
    note = get_object_or_404(Note, pk=note_id, owner=request.user)

    summary_text = f"Summary of {note.title}: This is a key note covering important conce
    keywords = ['important', 'key_concept', 'summary', 'note']

    summary, created = Summary.objects.get_or_create(
        note=note,
        defaults={'summary_text': summary_text, 'keywords': keywords}
    )

    return render(request, 'notes/summary_view.html', {'note': note, 'summary': summary})
```

```
# ============= PROGRESS API =============
@login_required
def progress_api(request):
    user = request.user
    subjects = Subject.objects.filter(owner=user)

    data = {
        'labels': [s.name for s in subjects[:5]],
        'data': [50 + (i * 10) for i in range(len(subjects[:5]))],
    }
    return JsonResponse(data)
```

15. notes/urls.py

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.home, name='home'),
    path('login/', views.login_view, name='login'),
    path('register/', views.register_view, name='register'),
    path('logout/', views.logout_view, name='logout'),
    path('dashboard/', views.dashboard, name='dashboard'),

    path('subjects/', views.subject_list, name='subject_list'),
    path('subjects/create/', views.subject_create, name='subject_create'),
    path('subjects/&lt;int:pk&gt;/', views.subject_detail, name='subject_detail'),
    path('subjects/&lt;int:pk&gt;/edit/', views.subject_edit, name='subject_edit'),
    path('subjects/&lt;int:pk&gt;/delete/', views.subject_delete, name='subject_delete'),

    path('topics/create/&lt;int:subject_id&gt;/', views.topic_create, name='topic_create'
    path('topics/&lt;int:pk&gt;/edit/', views.topic_edit, name='topic_edit'),

    path('notes/create/&lt;int:topic_id&gt;/', views.note_create, name='note_create'),
    path('notes/&lt;int:pk&gt;/', views.note_view, name='note_view'),
    path('notes/&lt;int:pk&gt;/edit/', views.note_edit, name='note_edit'),
    path('notes/&lt;int:pk&gt;/delete/', views.note_delete, name='note_delete'),

    path('bookmarks/', views.bookmark_list, name='bookmark_list'),
    path('bookmarks/toggle/&lt;int:note_id&gt;/', views.bookmark_toggle, name='bookmark_t

    path('tasks/', views.task_list, name='task_list'),
    path('tasks/create/', views.task_create, name='task_create'),
    path('tasks/&lt;int:pk&gt;/complete/', views.task_complete, name='task_complete'),

    path('collaboration/&lt;int:subject_id&gt;/', views.collaboration_manage, name='colla

    path('pdf/compile/', views.pdf_compile, name='pdf_compile'),

    path('summarizer/', views.summarizer, name='summarizer'),
    path('summarizer/&lt;int:note_id&gt;/', views.summarize_note, name='summarize_note'),
    path('api/progress/', views.progress_api, name='progress_api'),
]
```

16. notes/admin.py

```
from django.contrib import admin
from .models import (CustomUser, Subject, Topic, Note, Bookmark, Task,
                     Progress, Collaboration, Summary)

@admin.register(CustomUser)
class CustomUserAdmin(admin.ModelAdmin):
    list_display = ('username', 'email', 'role', 'date_joined')
    list_filter = ('role',)
    search_fields = ('username', 'email')

@admin.register(Subject)
class SubjectAdmin(admin.ModelAdmin):
    list_display = ('name', 'owner', 'created_at')
    list_filter = ('created_at', 'owner')
    search_fields = ('name',)

@admin.register(Topic)
class TopicAdmin(admin.ModelAdmin):
    list_display = ('name', 'subject', 'order')
    list_filter = ('subject',)
    search_fields = ('name',)

@admin.register(Note)
class NoteAdmin(admin.ModelAdmin):
    list_display = ('title', 'topic', 'owner', 'is_public', 'created_at')
    list_filter = ('is_public', 'created_at', 'owner')
    search_fields = ('title', 'content')

@admin.register(Bookmark)
class BookmarkAdmin(admin.ModelAdmin):
    list_display = ('user', 'note', 'created_at')
    list_filter = ('created_at', 'user')

@admin.register(Task)
class TaskAdmin(admin.ModelAdmin):
    list_display = ('title', 'user', 'due_date', 'completed')
    list_filter = ('completed', 'due_date', 'user')
    search_fields = ('title',)

@admin.register(Progress)
class ProgressAdmin(admin.ModelAdmin):
    list_display = ('user', 'completion_percentage', 'last_updated')
    list_filter = ('user', 'last_updated')

@admin.register(Collaboration)
class CollaborationAdmin(admin.ModelAdmin):
    list_display = ('subject', 'user', 'permission_level', 'created_at')
    list_filter = ('permission_level', 'created_at')

@admin.register(Summary)
```

```
class SummaryAdmin(admin.ModelAdmin):
    list_display = ('note', 'created_at')
    list_filter = ('created_at',)
```

TEMPLATES

17. templates/base.html

```

&lt;html lang="en"&gt;
&lt;head&gt;
    &lt;meta charset="UTF-8"&gt;
    &lt;meta name="viewport" content="width=device-width, initial-scale=1.0"&gt;
    &lt;title&gt;{% block title %}NoteEve - Notes Management{% endblock %}&lt;/title&gt;
    &lt;link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.cs
    &lt;link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.
    {% load static %}
    &lt;link rel="stylesheet" href="{% static 'css/style.css' %}"&gt;
    &lt;script src="https://cdn.jsdelivr.net/npm/chart.js@3.9.1/dist/chart.min.js"&gt;&lt
    {% block extra_css %}{% endblock %}
&lt;/head&gt;
&lt;body&gt;
    &lt;nav class="navbar navbar-expand-lg navbar-dark bg-primary sticky-top"&gt;
        <div>
            <a href="{% url 'dashboard' %}">
                &lt;i class="fas fa-book"&gt;&lt;/i&gt; NoteEve
            </a>
            &lt;button class="navbar-toggler" type="button" data-bs-toggle="collapse" dat
                <span></span>
            &lt;/button&gt;
            <div>
                <ul>
                    {% if user.is_authenticated %}
                    <li>
                        <a href="{% url 'dashboard' %}">&lt;i class="fas fa-home"&gt;&lt;
                    </li>
                    <li>
                        <a href="{% url 'subject_list' %}">&lt;i class="fas fa-folder"&gt
                    </li>
                    <li>
                        <a href="{% url 'task_list' %}">&lt;i class="fas fa-tasks"&gt;&lt
                    </li>
                    <li>
                        <a href="{% url 'bookmark_list' %}">&lt;i class="fas fa-bookmark"
                    </li>
                    <li>
                        <a href="#">
                            &lt;i class="fas fa-user"&gt;&lt;/i&gt; {{ user.username }}
                        </a>
                        <ul>
                            <li><a href="{% url 'logout' %}">Logout</a></li>
                        </ul>
                    </li>
                    {% else %}
                    <li>
                        <a href="{% url 'login' %}">Login</a>
                    </li>
                    <li>
                        <a href="{% url 'register' %}">Register</a>
                    </li>
                    {% endif %}
                </ul>
            </div>
        </div>
    &lt;/nav&gt;

    <div>
        {% if messages %}
        {% for message in messages %}
        <div>
            {{ message }}
            &lt;button type="button" class="btn-close" data-bs-dismiss="alert"&gt;&lt;/bu
        </div>
        {% endfor %}
        {% endif %}
    </div>

    &lt;main class="container my-4"&gt;
        {% block content %}{% endblock %}
    &lt;/main&gt;

    &lt;footer class="bg-light py-4 mt-5"&gt;
        <div>
            <p>&copy; 2025 NoteEve. All rights reserved.</p>
        </div>
    &lt;/footer&gt;

    &lt;script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle
    &lt;script src="https://code.jquery.com/jquery-3.6.0.min.js"&gt;&lt;/script&gt;
    {% block extra_js %}{% endblock %}
&lt;/body&gt;
&lt;/html&gt;
```

18. notes/templates/notes/login.html

```
{% extends "base.html" %}

{% block title %}Login - NoteEve{% endblock %}

{% block content %}
<div>
    <div>
        <div>
            <div>
                <h2>
                    &lt;i class="fas fa-sign-in-alt"&gt;&lt;/i&gt; Login
                </h2>
                &lt;form method="post"&gt;
```

```
                    {% csrf_token %}
                    <div>
                        &lt;label for="username" class="form-label"&gt;Username&lt;/label
                        &lt;input type="text" class="form-control" id="username" name="us
                    </div>
                    <div>
                        &lt;label for="password" class="form-label"&gt;Password&lt;/label
                        &lt;input type="password" class="form-control" id="password" name
                    </div>
                    &lt;button type="submit" class="btn btn-primary w-100"&gt;Login&lt;/b
                &lt;/form&gt;
                <hr>
                <p>Don't have an account? <a href="{% url 'register' %}">Register here</a
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

19. notes/templates/notes/register.html

```
{% extends "base.html" %}

{% block title %}Register - NoteEve{% endblock %}

{% block content %}
<div>
    <div>
        <div>
            <div>
                <h2>
                    &lt;i class="fas fa-user-plus"&gt;&lt;/i&gt; Register
                </h2>
                &lt;form method="post"&gt;
                    {% csrf_token %}
                    {{ form.as_p }}
                    &lt;button type="submit" class="btn btn-primary w-100"&gt;Register&lt
                &lt;/form&gt;
                <hr>
                <p>Already have an account? <a href="{% url 'login' %}">Login here</a></p
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

20. notes/templates/notes/dashboard.html

```
{% extends "base.html" %}

{% block title %}Dashboard - NoteEve{% endblock %}

{% block content %}
```

```
<div>
    <div>
        <h1>&lt;i class="fas fa-tachometer-alt"&gt;&lt;/i&gt; Dashboard</h1>
    </div>
</div>

<div>
    <div>
        <div>
            <div>
                <h5>Total Notes</h5>
                <p>{{ total_notes }}</p>
            </div>
        </div>
    </div>
    <div>
        <div>
            <div>
                <h5>Completed Tasks</h5>
                <p>{{ completed_tasks }}</p>
            </div>
        </div>
    </div>
    <div>
        <div>
            <div>
                <h5>Subjects</h5>
                <p>{{ subjects.count }}</p>
            </div>
        </div>
    </div>
    <div>
        <div>
            <div>
                <h5>Bookmarks</h5>
                <p>{{ bookmarks.count }}</p>
            </div>
        </div>
    </div>
</div>

<div>
    <div>
        <div>
            <div>
                <h5>&lt;i class="fas fa-tasks"&gt;&lt;/i&gt; Recent Tasks</h5>
            </div>
            <div>
                {% if tasks %}
                <ul>
                    {% for task in tasks %}
                    <li>
                        <span>{{ task.title }}</span>
                        &lt;small class="text-muted"&gt;{{ task.due_date|date:"M d, Y" }}
                    </li>
                    {% endfor %}
                </ul>
                {% else %}
                <p>No pending tasks</p>
                {% endif %}
            </div>
        </div>
    </div>
    <div>
        <div>
            <div>
                <h5>&lt;i class="fas fa-chart-bar"&gt;&lt;/i&gt; Progress Overview</h5>
            </div>
            <div>
                &lt;canvas id="progressChart"&gt;&lt;/canvas&gt;
            </div>
        </div>
    </div>
</div>

<div>
    <div>
        <a href="{% url 'subject_create' %}">
            &lt;i class="fas fa-plus"&gt;&lt;/i&gt; New Subject
        </a>
        <a href="{% url 'task_create' %}">
            &lt;i class="fas fa-tasks"&gt;&lt;/i&gt; New Task
        </a>
    </div>
</div>

{% endblock %}

{% block extra_js %}
&lt;script&gt;
fetch('{% url "progress_api" %}')
    .then(response =&gt; response.json())
    .then(data =&gt; {
        const ctx = document.getElementById('progressChart').getContext('2d');
        new Chart(ctx, {
            type: 'bar',
            data: {
                labels: data.labels,
                datasets: [{
                    label: 'Progress %',
                    data: data.data,
                    backgroundColor: 'rgba(75, 192, 192, 0.6)',
                    borderColor: 'rgba(75, 192, 192, 1)',
                    borderWidth: 1
                }]
            },
            options: {
                responsive: true,
                scales: { y: { beginAtZero: true, max: 100 } }
            }
        });
    });
```

```
&lt;/script&gt;
{% endblock %}
```

STATIC FILES

21. static/css/style.css

```
:root {
    --primary: #0d6efd;
    --secondary: #6c757d;
    --success: #198754;
    --info: #0dcaf0;
    --warning: #ffc107;
    --danger: #dc3545;
}

body {
    background-color: #f8f9fa;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

.navbar {
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

.navbar-brand {
    font-weight: 700;
    font-size: 1.5rem;
}

.card {
    border: none;
    box-shadow: 0 0.125rem 0.25rem rgba(0, 0, 0, 0.075);
    margin-bottom: 1rem;
    border-radius: 8px;
}

.card-header {
    border-bottom: 1px solid rgba(0,0,0,0.125);
    font-weight: 600;
}

.btn {
    border-radius: 6px;
    font-weight: 500;
    transition: all 0.3s ease;
}

.btn:hover {
    transform: translateY(-2px);
    box-shadow: 0 4px 8px rgba(0,0,0,0.2);
}
```

```css
.table-hover tbody tr:hover {
    background-color: #f8f9fa;
}

.badge {
    padding: 0.35em 0.65em;
    font-weight: 500;
}

.list-group-item {
    border: 1px solid rgba(0,0,0,0.125);
    margin-bottom: 0.5rem;
    border-radius: 6px;
}

main {
    min-height: 70vh;
}

footer {
    margin-top: auto;
    border-top: 1px solid rgba(0,0,0,0.1);
}

.form-control, .form-select {
    border-radius: 6px;
    border: 1px solid #dee2e6;
}

.form-control:focus, .form-select:focus {
    border-color: var(--primary);
    box-shadow: 0 0 0 0.2rem rgba(13, 110, 253, 0.25);
}

.alert {
    border-radius: 6px;
    border: none;
}

.card.bg-primary, .card.bg-success, .card.bg-info, .card.bg-warning {
    border: none;
    color: white;
}

.display-4 {
    font-weight: 700;
}

@media (max-width: 768px) {
    .container {
        padding: 0 1rem;
    }
}
```

22. static/js/main.js

```
document.addEventListener('DOMContentLoaded', function() {
    const tooltipTriggerList = [].slice.call(document.querySelectorAll('[data-bs-toggle='
    tooltipTriggerList.map(function (tooltipTriggerEl) {
        return new bootstrap.Tooltip(tooltipTriggerEl)
    })
});

function toggleBookmark(noteId) {
    const csrf = document.querySelector('[name=csrfmiddlewaretoken]').value;
    fetch(`/bookmarks/toggle/${noteId}/`, {
        method: 'POST',
        headers: {
            'X-CSRFToken': csrf,
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({})
    })
    .then(response =&gt; response.json())
    .then(data =&gt; {
        location.reload();
    })
    .catch(error =&gt; console.error('Error:', error));
}
```

PART 3: QUICK SETUP INSTRUCTIONS

Step 1: Create Project Directory

```
mkdir noteeve_django
cd noteeve_django
```

Step 2: Create Virtual Environment

```
python -m venv venv

# Windows (PowerShell)
.\\venv\\Scripts\\Activate.ps1

# macOS/Linux
source venv/bin/activate
```

Step 3: Create Django Project

```
pip install django
django-admin startproject noteeve .
django-admin startapp notes
```

Step 4: Replace Configuration Files

Replace the generated settings.py, urls.py, and other files with the code provided above.

Step 5: Install Dependencies

```
pip install -r requirements.txt
```

Step 6: Setup PostgreSQL Database

```
createdb noteeve_db
createuser noteeve_user
```

Update .env file with credentials

Step 7: Run Migrations

```
python manage.py migrate
```

Step 8: Create Superuser

```
python manage.py createsuperuser
```

Step 9: Collect Static Files

```
python manage.py collectstatic --noinput
```

Step 10: Run Development Server

```
python manage.py runserver
```

Access at: http://127.0.0.1:8000/

Features Available After Setup

✅ User Authentication (Login/Register)
✅ Dashboard with Progress Charts
✅ Subjects/Topics/Notes Management
✅ Rich Text Editor (TinyMCE)
✅ Bookmarks
✅ To-Do List & Tasks
✅ Collaboration & Sharing
✅ PDF Compiler

✅ AI Summarizer
✅ Admin Panel

Ready to build! Follow the 10 steps above to get your NoteEve running locally.
"