**Collaborations.html**

<!-- templates/authentication/dashboard/collaborations.html -->

{% extends 'authentication/dashboard/base.html' %}

{% load static %}

{% block extra\_css %}

<link href="https://cdn.jsdelivr.net/npm/quill@2.0.3/dist/quill.snow.css" rel="stylesheet">

<style>

    .collaboration-card {

        transition: transform 0.2s;

        border: 1px solid #e9ecef;

    }

    .collaboration-card:hover {

        transform: translateY(-3px);

    }

    .participant-avatar {

        width: 32px;

        height: 32px;

        object-fit: cover;

    }

    .status-badge {

        font-size: 0.75rem;

        padding: 0.35em 0.65em;

    }

    .update-item {

        padding: 1rem;

        border-radius: 8px;

        background-color: #f8f9fa;

        margin-bottom: 0.75rem;

    }

    .stat-icon {

        width: 48px;

        height: 48px;

        border-radius: 12px;

        display: flex;

        align-items: center;

        justify-content: center;

    }

    @media (max-width: 768px) {

        .search-input-group {

            width: 100% !important;

        }

        .stat-card {

            margin-bottom: 1rem;

        }

        .collaboration-actions {

            flex-wrap: wrap;

            gap: 0.5rem;

        }

    }

</style>

{% endblock %}

{% block content %}

<div class="container-fluid">

    <!-- Header Section -->

    <div class="d-flex flex-column flex-md-row justify-content-between align-items-start align-items-md-center mb-4 gap-3">

        <h2 class="mb-0">Collaborations</h2>

        <div class="search-input-group w-50">

            <div class="input-group">

                <span class="input-group-text bg-transparent border-end-0">

                    <i class="fas fa-search text-muted"></i>

                </span>

                <input type="text" class="form-control search-input border-start-0"

                       placeholder="Search collaborations...">

            </div>

        </div>

    </div>

    <!-- Statistics Cards -->

    <div class="row g-4 mb-4">

        <div class="col-md-4 stat-card">

            <div class="card h-100 border-0 shadow-sm">

                <div class="card-body">

                    <div class="d-flex align-items-center justify-content-between">

                        <div class="d-flex align-items-center gap-3">

                            <div class="stat-icon bg-primary bg-opacity-10 text-primary">

                                <i class="fas fa-users fa-lg"></i>

                            </div>

                            <div>

                                <h6 class="text-muted mb-2">Total Collaborations</h6>

                                <h2 class="mb-0">{{ total\_collaborations }}</h2>

                            </div>

                        </div>

                        <span class="badge bg-success bg-opacity-10 text-success fs-6">

                            <i class="fas fa-arrow-up me-1"></i>12%

                        </span>

                    </div>

                </div>

            </div>

        </div>

        <div class="col-md-4 stat-card">

            <div class="card h-100 border-0 shadow-sm">

                <div class="card-body">

                    <div class="d-flex align-items-center justify-content-between">

                        <div class="d-flex align-items-center gap-3">

                            <div class="stat-icon bg-info bg-opacity-10 text-info">

                                <i class="fas fa-folder-open fa-lg"></i>

                            </div>

                            <div>

                                <h6 class="text-muted mb-2">Active Projects</h6>

                                <h2 class="mb-0">{{ active\_projects }}</h2>

                            </div>

                        </div>

                        <span class="badge bg-success bg-opacity-10 text-success fs-6">

                            <i class="fas fa-arrow-up me-1"></i>8%

                        </span>

                    </div>

                </div>

            </div>

        </div>

        <div class="col-md-4 stat-card">

            <div class="card h-100 border-0 shadow-sm">

                <div class="card-body">

                    <div class="d-flex align-items-center justify-content-between">

                        <div class="d-flex align-items-center gap-3">

                            <div class="stat-icon bg-warning bg-opacity-10 text-warning">

                                <i class="fas fa-user-plus fa-lg"></i>

                            </div>

                            <div>

                                <h6 class="text-muted mb-2">Total Collaborators</h6>

                                <h2 class="mb-0">{{ total\_collaborators }}</h2>

                            </div>

                        </div>

                        <span class="badge bg-success bg-opacity-10 text-success fs-6">

                            <i class="fas fa-arrow-up me-1"></i>24%

                        </span>

                    </div>

                </div>

            </div>

        </div>

    </div>

    <!-- Control Elements Section -->

    <div class="d-flex flex-column flex-md-row justify-content-between align-items-start align-items-md-center mb-4 gap-3">

        <button class="btn btn-primary" data-bs-toggle="modal" data-bs-target="#newCollaborationModal">

            <i class="fas fa-plus me-2"></i>New Collaboration

        </button>

        <div class="d-flex flex-column flex-md-row gap-2">

            <div class="dropdown">

                <button class="btn btn-light border text-start"

                    type="button"

                    data-bs-toggle="dropdown"

                    aria-expanded="false"

                    data-status-button>

                    Status: All <i class="fas fa-chevron-down ms-2"></i>

                </button>

                <ul class="dropdown-menu">

                    <li><a class="dropdown-item" href="#" data-status="all">All</a></li>

                    <li><a class="dropdown-item" href="#" data-status="active">Active</a></li>

                    <li><a class="dropdown-item" href="#" data-status="pending">Pending</a></li>

                </ul>

            </div>

            <div class="dropdown">

                <button class="btn btn-light border text-start"

                    type="button"

                    data-bs-toggle="dropdown"

                    aria-expanded="false"

                    data-sort-button>

                    Sort by: Last Updated <i class="fas fa-chevron-down ms-2"></i>

                </button>

                <ul class="dropdown-menu">

                    <li><a class="dropdown-item" href="#" data-sort="updated">Last Updated</a></li>

                    <li><a class="dropdown-item" href="#" data-sort="created">Date Created</a></li>

                    <li><a class="dropdown-item" href="#" data-sort="name">Project Name</a></li>

                </ul>

            </div>

        </div>

    </div>

    <!-- Main Content -->

    <div class="row g-4">

        <!-- Active Collaborations -->

        <div class="col-lg-8">

            <div class="card border-0 shadow-sm">

                <div class="card-header bg-white border-0 py-3">

                    <h5 class="mb-0">Active Collaborations</h5>

                </div>

                <div class="card-body">

                    <div id="collaborationsContainer" class="row g-3">

                        {% for collab in collaborations %}

                        <div class="col-12 collaboration-item">

                            <div class="card collaboration-card">

                                <div class="card-body">

                                    <div class="d-flex justify-content-between align-items-start mb-3">

                                        <div>

                                            <h5 class="card-title mb-1">{{ collab.title }}</h5>

                                            <p class="card-text text-muted small mb-0">

                                                {{ collab.description }}

                                            </p>

                                        </div>

                                        <span class="badge status-badge

                                            {% if collab.status == 'active' %}bg-success{% else %}bg-warning{% endif %}">

                                            {{ collab.status|title }}

                                        </span>

                                    </div>

                                    <div class="d-flex justify-content-between align-items-center">

                                        <div class="d-flex align-items-center">

                                            <div class="avatar-group">

                                                {% for participant in collab.participants.all|slice:":3" %}

                                                <div class="avatar-sm me-2">

                                                    {% if participant.profile\_picture %}

                                                    <img src="{{ participant.profile\_picture.url }}"

                                                         class="rounded-circle participant-avatar"

                                                         alt="{{ participant.get\_full\_name }}">

                                                    {% else %}

                                                    <div class="rounded-circle bg-primary text-white d-flex

                                                        align-items-center justify-content-center participant-avatar">

                                                        {{ participant.get\_full\_name|default:participant.email|make\_list|first|upper }}

                                                    </div>

                                                    {% endif %}

                                                </div>

                                                {% endfor %}

                                                {% if collab.participants.count > 3 %}

                                                <div class="avatar-sm bg-light text-muted rounded-circle

                                                    d-flex align-items-center justify-content-center participant-avatar">

                                                    +{{ collab.participants.count|add:"-3" }}

                                                </div>

                                                {% endif %}

                                            </div>

                                        </div>

                                        <div class="text-end">

                                            <small class="text-muted d-block">Updated</small>

                                            <small class="text-muted">{{ collab.updated\_at|timesince }} ago</small>

                                        </div>

                                    </div>

                                </div>

                                <div class="card-footer bg-transparent border-0 d-flex justify-content-end gap-2 pt-0 collaboration-actions">

                                    <button class="btn btn-sm btn-light">

                                        <i class="fas fa-eye me-2"></i>View

                                    </button>

                                    <button class="btn btn-sm btn-light">

                                        <i class="fas fa-pencil-alt me-2"></i>Edit

                                    </button>

                                    <button class="btn btn-sm btn-light">

                                        <i class="fas fa-share me-2"></i>Share

                                    </button>

                                </div>

                            </div>

                        </div>

                        {% endfor %}

                    </div>

                </div>

            </div>

        </div>

        <!-- Recent Updates -->

        <div class="col-lg-4">

            <div class="card border-0 shadow-sm h-100">

                <div class="card-header bg-white border-0 py-3">

                    <h5 class="mb-0">Recent Updates</h5>

                </div>

                <div class="card-body">

                    <div id="updatesContainer" class="updates-list">

                        {% for activity in recent\_activities %}

                        <div class="update-item">

                            <div class="d-flex align-items-start">

                                <div class="avatar-sm me-3">

                                    {% if activity.user.profile\_picture %}

                                    <img src="{{ activity.user.profile\_picture.url }}"

                                         class="rounded-circle participant-avatar"

                                         alt="{{ activity.user.get\_full\_name }}">

                                    {% else %}

                                    <div class="rounded-circle bg-primary text-white d-flex

                                        align-items-center justify-content-center participant-avatar">

                                        {{ activity.user.get\_full\_name|default:activity.user.email|make\_list|first|upper }}

                                    </div>

                                    {% endif %}

                                </div>

                                <div class="flex-grow-1">

                                    <p class="mb-1 small">

                                        <strong>{{ activity.user.get\_full\_name }}</strong>

                                        {{ activity.get\_activity\_type\_display }}

                                        {% if activity.details %}<span class="text-muted">- {{ activity.details }}</span>{% endif %}

                                    </p>

                                    <small class="text-muted">{{ activity.created\_at|timesince }} ago</small>

                                </div>

                            </div>

                        </div>

                        {% endfor %}

                    </div>

                </div>

            </div>

        </div>

    </div>

</div>

<!-- New Collaboration Modal -->

<div class="modal fade" id="newCollaborationModal" tabindex="-1">

    <div class="modal-dialog modal-dialog-centered">

        <div class="modal-content">

            <div class="modal-header border-0">

                <h5 class="modal-title">Start New Collaboration</h5>

                <button type="button" class="btn-close" data-bs-dismiss="modal"></button>

            </div>

            <div class="modal-body">

                <form id="newCollaborationForm">

                    <div class="mb-3">

                        <label class="form-label">Project Title</label>

                        <input type="text" class="form-control" required>

                    </div>

                    <div class="mb-3">

                        <label class="form-label">Description</label>

                        <textarea class="form-control" rows="3"></textarea>

                    </div>

                    <div class="mb-3">

                        <label class="form-label">Add Participants</label>

                        <select class="form-select" multiple>

                            {% for user in users %}

                            <option value="{{ user.id }}">{{ user.get\_full\_name|default:user.email }}</option>

                            {% endfor %}

                        </select>

                    </div>

                </form>

                <div id="editor"></div>

            </div>

            <div class="modal-footer border-0">

                <button type="button" class="btn btn-light" data-bs-dismiss="modal">Cancel</button>

                <button type="submit" form="newCollaborationForm" class="btn btn-primary">Create Project</button>

            </div>

        </div>

    </div>

</div>

{% endblock %}

{% block extra\_js %}

<script src="https://cdn.jsdelivr.net/npm/quill@2.0.3/dist/quill.js"></script>

<script>

    // Filter and Sort Functionality

    let currentStatus = 'all';

    let currentSort = 'updated';

    // For status dropdown

    document.querySelectorAll('.dropdown-item[data-status]').forEach(item => {

        item.addEventListener('click', (e) => {

            e.preventDefault();

            currentStatus = e.target.dataset.status;

            const statusButton = document.querySelector('[data-bs-toggle="dropdown"][data-status-button]');

            if (statusButton) {

                statusButton.textContent = `Status: ${e.target.textContent}`;

            }

            loadCollaborations();

        });

    });

    document.querySelectorAll('.dropdown-item[data-sort]').forEach(item => {

        item.addEventListener('click', (e) => {

            e.preventDefault();

            currentSort = e.target.dataset.sort;

            // Better selector for the sort dropdown button

            const sortButton = document.querySelector('[data-bs-toggle="dropdown"][data-sort-button]');

            if (sortButton) {

                sortButton.textContent = `Sort by: ${e.target.textContent}`;

            } else {

                console.error('Sort button not found');

            }

            loadCollaborations();

        });

    });

    async function loadCollaborations() {

        const container = document.getElementById('collaborationsContainer');

        container.innerHTML = '<div class="col-12 text-center py-5"><div class="spinner-border text-primary" role="status"></div></div>';

        const response = await fetch(`/auth/api/collaborations/?status=${currentStatus}&sort=${currentSort}`);

        const data = await response.json();

        container.innerHTML = data.html;

        initInfiniteScroll();

    }

    // Infinite Scroll Initialization

    function initInfiniteScroll() {

        const observer = new IntersectionObserver((entries) => {

            entries.forEach(entry => {

                if (entry.isIntersecting) {

                    loadMoreContent();

                }

            });

        });

        const lastItem = document.querySelector('#collaborationsContainer .collaboration-item:last-child');

        if (lastItem) observer.observe(lastItem);

    }

    let isLoading = false;

    async function loadMoreContent() {

        if (isLoading) return;

        isLoading = true;

        const offset = document.querySelectorAll('.collaboration-item').length;

        const response = await fetch(

            `/auth/api/collaborations/?status=${currentStatus}&sort=${currentSort}&offset=${offset}`

        );

        const data = await response.json();

        const container = document.getElementById('collaborationsContainer');

        container.insertAdjacentHTML('beforeend', data.html);

        isLoading = false;

        initInfiniteScroll();

    }

    // Initial setup

    document.addEventListener('DOMContentLoaded', () => {

        initInfiniteScroll();

    });

    // WebSocket Connection

    let collaborationSocket = null;

    // Initialize WebSocket only AFTER successful collaboration creation

    function connectToCollaboration(collabId) {

        const wsProtocol = window.location.protocol === 'https:' ? 'wss://' : 'ws://';

        const wsUrl = `${wsProtocol}${window.location.host}/ws/collaboration/${collabId}/`;

        collaborationSocket = new WebSocket(wsUrl);

        // Add your WebSocket handlers here

        collaborationSocket.onopen = function(e) {

            console.log('WebSocket connection established');

        };

        collaborationSocket.onerror = function(e) {

            console.error('WebSocket error:', e);

        };

    }

    // Real-time Document Handling

    let localDocumentState = {};

    function handleDocumentUpdate(data) {

        if(data.user !== currentUserEmail) {

            // Apply remote changes to editor

            quill.updateContents(data.operations);

        }

    }

    // Initialize Quill Editor

    {% comment %} document.addEventListener('DOMContentLoaded', function() {

        const quill = new Quill('#editor', {

            modules: { toolbar: [

                ['bold', 'italic', 'underline'],

                ['code-block', 'blockquote'],

                [{ 'list': 'ordered'}, { 'list': 'bullet' }],

                ['link', 'image'],

                ['clean']

            ]},

            theme: 'snow'

        });

    }); {% endcomment %}

    const quill = new Quill('#editor', {

        theme: 'snow'

    });

    quill.on('text-change', (delta, oldDelta, source) => {

        if (source === 'user') {

            collaborationSocket.send(JSON.stringify({

                type: 'document\_update',

                operations: delta,

                documentId: currentDocumentId

            }));

        }

    });

    //Add Select2 initialization

    $(document).ready(function() {

        $('select[multiple]').select2({

            placeholder: 'Select participants...',

            width: '100%'

        });

    });

    // New Collaboration Form Submission

    document.getElementById('newCollaborationForm').addEventListener('submit', async (e) => {

        e.preventDefault();

        const formData = new FormData(e.target);

        try {

            const response = await fetch('{% url "authentication:create\_collaboration" %}', {

                method: 'POST',

                body: formData,

                headers: {

                    'X-CSRFToken': '{{ csrf\_token }}'

                }

            });

            const data = await response.json();

            if (data.redirect) {

                // Connect WebSocket AFTER successful creation

                const collabId = data.redirect.split('/').slice(-2)[0];

                connectToCollaboration(collabId);

                window.location.href = data.redirect;

            }

        } catch (error) {

            console.error('Submission error:', error);

        }

    });

</script>

{% endblock %}

**Consumers.py**

# authentication/consumers.py

import json

import logging

from channels.generic.websocket import AsyncWebsocketConsumer

from channels.db import database\_sync\_to\_async

from django.contrib.auth.models import AnonymousUser

from .models import Collaboration, User

from django.utils import timezone

logger = logging.getLogger(\_\_name\_\_)

class CollaborationConsumer(AsyncWebsocketConsumer):

    async def connect(self):

        # Reject unauthenticated users immediately

        if isinstance(self.scope["user"], AnonymousUser):

            await self.close(code=4001)

            return

        self.user = self.scope["user"]

        self.collab\_id = self.scope['url\_route']['kwargs']['collab\_id']

        try:

            if await self.validate\_access():

                await self.channel\_layer.group\_add(

                    self.collab\_group\_name,

                    self.channel\_name

                )

                await self.accept()

                await self.send\_user\_list()

                logger.info(f"User {self.user.email} connected to collaboration {self.collab\_id}")

            else:

                await self.close(code=4003)

        except Exception as e:

            logger.error(f"Connection error: {str(e)}")

            await self.close(code=4002)

    @property

    def collab\_group\_name(self):

        return f"collab\_{self.collab\_id}"

    @database\_sync\_to\_async

    def validate\_access(self):

        """Verify user has access to this collaboration"""

        return Collaboration.objects.filter(

            id=self.collab\_id,

            participants=self.user

        ).exists()

    @database\_sync\_to\_async

    def get\_participants(self):

        """Get list of participant emails with safe exception handling"""

        try:

            collab = Collaboration.objects.get(id=self.collab\_id)

            return list(collab.participants.values\_list('email', flat=True))

        except Collaboration.DoesNotExist:

            logger.error(f"Collaboration {self.collab\_id} not found")

            return []

        except Exception as e:

            logger.error(f"Error getting participants: {str(e)}")

            return []

    async def send\_user\_list(self):

        """Send updated participant list to all group members"""

        participants = await self.get\_participants()

        try:

            await self.channel\_layer.group\_send(

                self.collab\_group\_name,

                {

                    'type': 'user\_list',

                    'participants': participants

                }

            )

        except Exception as e:

            logger.error(f"Error sending user list: {str(e)}")

    async def user\_list(self, event):

        """Handle user list updates"""

        try:

            await self.send(text\_data=json.dumps({

                'type': 'user\_list',

                'participants': event['participants']

            }))

        except Exception as e:

            logger.error(f"Error sending user list to client: {str(e)}")

    async def receive(self, text\_data):

        """Handle incoming WebSocket messages"""

        try:

            data = json.loads(text\_data)

            data['user'] = self.user.email

            # Validate message type

            if data.get('type') not in ['document\_update', 'cursor\_position']:

                raise ValueError("Invalid message type")

            # Broadcast to collaboration group

            await self.channel\_layer.group\_send(

                self.collab\_group\_name,

                {

                    'type': data['type'],

                    'data': data

                }

            )

        except json.JSONDecodeError:

            logger.error("Received invalid JSON data")

        except ValueError as ve:

            logger.error(f"Validation error: {str(ve)}")

        except Exception as e:

            logger.error(f"Error processing message: {str(e)}")

    async def document\_update(self, event):

        """Handle document update events"""

        try:

            # Add timestamp before sending

            event['data']['timestamp'] = timezone.now().isoformat()

            await self.send(text\_data=json.dumps(event['data']))

        except Exception as e:

            logger.error(f"Error sending document update: {str(e)}")

    async def cursor\_position(self, event):

        """Handle cursor position updates"""

        try:

            await self.send(text\_data=json.dumps(event['data']))

        except Exception as e:

            logger.error(f"Error sending cursor position: {str(e)}")

    async def disconnect(self, close\_code):

        """Handle clean disconnection"""

        try:

            await self.channel\_layer.group\_discard(

                self.collab\_group\_name,

                self.channel\_name

            )

            logger.info(f"User {self.user.email} disconnected from {self.collab\_id}")

        except Exception as e:

            logger.error(f"Error during disconnect: {str(e)}")

**Routing.py**

# authentication/routing.py

from django.urls import re\_path

from . import consumers

websocket\_urlpatterns = [

    re\_path(r'ws/collaboration/(?P<collab\_id>\w+)/$', consumers.CollaborationConsumer.as\_asgi()),

]

**Middleware.py**

# authentication/middleware.py

from channels.db import database\_sync\_to\_async

from django.contrib.auth.models import AnonymousUser

class WebSocketAuthMiddleware:

    def \_\_init\_\_(self, app):

        self.app = app

    async def \_\_call\_\_(self, scope, receive, send):

        user = await self.get\_user(scope)

        scope['user'] = user

        return await self.app(scope, receive, send)

    @database\_sync\_to\_async

    def get\_user(self, scope):

        session = scope.get('session')

        if not session:

            return AnonymousUser()

        return session.get('\_auth\_user\_id') or AnonymousUser()

**Models.py**

# from django.db import models

# Create your models here.

# models.py

from django.db import models

from django.contrib.auth.models import User

from django.contrib.auth.models import AbstractUser

from django.core.validators import EmailValidator

from django.core.exceptions import ValidationError

from PIL import Image

import os

# Validator to ensure the image size does not exceed the limit

def validate\_image\_size(image):

    filesize = image.size

    megabyte\_limit = 2.0  # Maximum allowed size is 2 MB

    if filesize > megabyte\_limit \* 1024 \* 1024:

        raise ValidationError(f"Image size cannot exceed {megabyte\_limit}MB")

# Function to generate the upload path for profile pictures

def profile\_picture\_path(instance, filename):

    # Save profile pictures under: media/profile\_pictures/<user\_id>/<filename>

    return f"profile\_pictures/{instance.id}/{filename}"

# Custom User model with profile picture and additional fields

class User(AbstractUser):

    ROLE\_CHOICES = [

        ('faculty', 'Faculty'),

        ('admin', 'Admin'),

        ('external', 'External Partner'),

    ]

    email = models.EmailField(

        unique=True,

        validators=[EmailValidator()],

        error\_messages={

            'unique': 'A user with this email already exists.',

        }

    )

    role = models.CharField(

        max\_length=20,

        choices=ROLE\_CHOICES

        # default='faculty'

    )

    last\_login\_attempt = models.DateTimeField(null=True, blank=True)

    login\_attempts = models.IntegerField(default=0)

    # Profile picture field

    profile\_picture = models.ImageField(

        upload\_to=profile\_picture\_path,

        validators=[validate\_image\_size],

        null=True,

        blank=True,

        help\_text="Upload a JPEG or PNG image (max size: 2MB)."

    )

    DEPARTMENT\_CHOICES = [

        ('Computer Science', 'Computer Science'),

        ('Software Engineering', 'Software Engineering'),

        ('Cyber Security', 'Cyber Security'),

    ]

    department = models.CharField(

        max\_length=100,

        choices=DEPARTMENT\_CHOICES,

        blank=True,

        null=True

    )

    USERNAME\_FIELD = 'email'

    REQUIRED\_FIELDS = ['username', 'role']

    class Meta:

        db\_table = 'auth\_user'

    def \_\_str\_\_(self):

        return self.email

    # Override save method to resize profile picture if necessary

    def save(self, \*args, \*\*kwargs):

        super().save(\*args, \*\*kwargs)

        if self.profile\_picture:

            img = Image.open(self.profile\_picture.path)

            max\_size = (300, 300)  # Resize image to a maximum of 300x300

            if img.height > max\_size[1] or img.width > max\_size[0]:

                img.thumbnail(max\_size)

                img.save(self.profile\_picture.path)

    # Override delete method to remove profile picture file when user is deleted

    def delete(self, \*args, \*\*kwargs):

        if self.profile\_picture and os.path.isfile(self.profile\_picture.path):

            os.remove(self.profile\_picture.path)

        super().delete(\*args, \*\*kwargs)

class File(models.Model):

    user = models.ForeignKey(User, on\_delete=models.CASCADE)

    name = models.CharField(max\_length=255)

    file = models.FileField(upload\_to='files/')

    file\_size = models.BigIntegerField()

    created\_at = models.DateTimeField(auto\_now\_add=True)

    modified\_at = models.DateTimeField(auto\_now=True)

    def \_\_str\_\_(self):

        return self.name

    def get\_file\_type(self):

        """Return the file extension for icon mapping"""

        return self.name.split('.')[-1].lower() if '.' in self.name else ''

class Collaboration(models.Model):

    STATUS\_CHOICES = [

        ('active', 'Active'),

        ('pending', 'Pending'),

        ('archived', 'Archived')

    ]

    user = models.ForeignKey(User, on\_delete=models.CASCADE)

    title = models.CharField(max\_length=200)

    description = models.TextField()

    file = models.ForeignKey(File, on\_delete=models.SET\_NULL, null=True)

    status = models.CharField(max\_length=20, choices=STATUS\_CHOICES, default='active')

    participants = models.ManyToManyField(User, related\_name='collaborations')

    created\_at = models.DateTimeField(auto\_now\_add=True)

    updated\_at = models.DateTimeField(auto\_now=True)

    channel\_group = models.CharField(max\_length=255, unique=True)

    def save(self, \*args, \*\*kwargs):

        if not self.channel\_group:

            self.channel\_group = f"collab\_{self.id}"

        super().save(\*args, \*\*kwargs)

    def \_\_str\_\_(self):

        return self.title

class CollaborationActivity(models.Model):

    ACTIVITY\_TYPES = [

        ('file\_add', 'File Added'),

        ('comment', 'Comment'),

        ('join', 'Participant Joined')

    ]

    collaboration = models.ForeignKey(Collaboration, on\_delete=models.CASCADE)

    user = models.ForeignKey(User, on\_delete=models.CASCADE)

    activity\_type = models.CharField(max\_length=20, choices=ACTIVITY\_TYPES)

    details = models.TextField(blank=True)

    created\_at = models.DateTimeField(auto\_now\_add=True)

class Notification(models.Model):

    user = models.ForeignKey(User, on\_delete=models.CASCADE)

    message = models.TextField()

    icon = models.CharField(max\_length=50)

    created\_at = models.DateTimeField(auto\_now\_add=True)

    is\_read = models.BooleanField(default=False)

    def \_\_str\_\_(self):

        return f"{self.user.username} - {self.message[:50]}"

class SharedFile(models.Model):

    file = models.ForeignKey(File, on\_delete=models.CASCADE)

    shared\_by = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name='shared\_by')

    shared\_with = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name='shared\_with')

    permission = models.CharField(max\_length=10, choices=[('view', 'View'), ('edit', 'Edit')])

    shared\_at = models.DateTimeField(auto\_now\_add=True)

    def \_\_str\_\_(self):

        return f"{self.file.name} shared by {self.shared\_by.email} with {self.shared\_with.email}"

**Views.py**

@login\_required

def create\_collaboration(request):

    if request.method == 'POST':

        form = CollaborationForm(request.POST, user=request.user)

        if form.is\_valid():

            collaboration = form.save(commit=False)

            collaboration.user = request.user

            collaboration.save()

            form.save\_m2m()  # Save participants

            # Initialize real-time channel

            collaboration.channel\_group = f"collab\_{collaboration.id}"

            collaboration.save()

            return JsonResponse({

                'status': 'success',

                'redirect': reverse('collaborations', args=[collaboration.id]),

                'collab\_id': collaboration.id  # Add explicit collab\_id

            })

        return JsonResponse({'status': 'error', 'errors': form.errors})

    return JsonResponse({'status': 'invalid\_request'}, status=400)

@login\_required

def collaboration\_session(request, collab\_id):

    collaboration = get\_object\_or\_404(Collaboration, id=collab\_id, participants=request.user)

    return render(request, 'collaborations', {

        'collaboration': collaboration,

        'file\_content': collaboration.file.file.read().decode('utf-8')

    })