from django.urls import path

from . import views as frontend\_views

from . import api\_views

app\_name = 'frontend'

urlpatterns = [

    path('login/', frontend\_views.FrontendLoginView.as\_view(), name='frontend-login'),

    path('register/', frontend\_views.FrontendRegisterView.as\_view(), name='frontend-register'),

    path('send-verification-code/', frontend\_views.SendVerificationCodeView.as\_view(), name='send-verification-code'),

    path('get-csrf-token/', frontend\_views.csrf\_token\_view, name='get-csrf-token'),

    path('profile/', frontend\_views.user\_profile, name='user-profile'),

    path('history/', frontend\_views.user\_history, name='user-history'),  # 新增 history 路径

    path('check\_login\_status/', frontend\_views.check\_login\_status, name='check-login-status'),  # 新增 check\_login\_status 路径

]

from django.contrib.auth import authenticate, login

from django.contrib.admin.models import LogEntry

import json

import random

from django.contrib.auth.models import User

from django.contrib.auth.password\_validation import validate\_password

from django.core.cache import cache

from django.core.mail import send\_mail

from django.http import JsonResponse

from django.views import View

from django.core.exceptions import ValidationError

from django.middleware.csrf import get\_token

from django.views.decorators.csrf import csrf\_exempt

from django.utils.decorators import method\_decorator

from rest\_framework.decorators import api\_view

# 发送验证码视图

@method\_decorator(csrf\_exempt, name='dispatch')

class SendVerificationCodeView(View):

    def post(self, request, \*args, \*\*kwargs):

        try:

            data = json.loads(request.body)

            email = data.get('email')

            if not email:

                return JsonResponse({'success': False, 'message': '電子郵件地址不能為空'}, status=400)

            # 生成6位數的驗證碼

            verification\_code = random.randint(100000, 999999)

            # 將驗證碼存儲在緩存中，有效期為5分鐘

            cache.set(email, verification\_code, timeout=300)

            # 發送電子郵件

            send\_mail(

                '您的驗證碼',

                f'您的驗證碼是 {verification\_code}，有效期為5分鐘。',

                'Leewesley527@gmail.com',  # 發件人郵箱

                [email],

                fail\_silently=False,

            )

            return JsonResponse({'success': True, 'message': '驗證碼已發送'}, status=200)

        except json.JSONDecodeError:

            return JsonResponse({'success': False, 'message': '無效的數據格式'}, status=400)

# 用户注册视图

class FrontendRegisterView(View):

    def post(self, request, \*args, \*\*kwargs):

        try:

            data = json.loads(request.body)

            username = data.get('username')

            email = data.get('email')

            password = data.get('password')

            confirm\_password = data.get('confirm\_password')

            phone = data.get('phone')

            verification\_code = data.get('verification\_code')

            if password != confirm\_password:

                return JsonResponse({'success': False, 'message': '密碼和確認密碼不匹配'}, status=400)

            try:

                validate\_password(password)

            except ValidationError as e:

                return JsonResponse({'success': False, 'message': str(e)}, status=400)

            cached\_verification\_code = cache.get(email)

            if not cached\_verification\_code or str(cached\_verification\_code) != verification\_code:

                return JsonResponse({'success': False, 'message': '驗證碼不正確或已過期'}, status=400)

            if User.objects.filter(username=username).exists():

                return JsonResponse({'success': False, 'message': '用戶名已存在'}, status=400)

            if User.objects.filter(email=email).exists():

                return JsonResponse({'success': False, 'message': '電子郵件已被註冊'}, status=400)

            user = User.objects.create\_user(username=username, email=email, password=password)

            user.save()

            cache.delete(email)

            send\_mail(

                '歡迎註冊',

                '感謝您的註冊！',

                'from@example.com',

                [email],

                fail\_silently=False,

            )

            return JsonResponse({'success': True, 'message': '註冊成功！'}, status=201)

        except json.JSONDecodeError:

            return JsonResponse({'success': False, 'message': '無效的數據格式'}, status=400)

# 用户登录视图

class FrontendLoginView(View):

    def post(self, request, \*args, \*\*kwargs):

        try:

            data = json.loads(request.body)

            email = data.get('email')

            password = data.get('password')

            user = authenticate(request, username=email, password=password)

            if user is not None:

                login(request, user)

                return JsonResponse({'success': True, 'message': '登錄成功'}, status=200)

            else:

                return JsonResponse({'success': False, 'message': '用戶名或密碼錯誤'}, status=400)

        except json.JSONDecodeError:

            return JsonResponse({'success': False, 'message': '無效的數據格式'}, status=400)

    def get(self, request, \*args, \*\*kwargs):

        csrf\_token = get\_token(request)

        response = JsonResponse({'csrfToken': csrf\_token})

        response.set\_cookie('csrftoken', csrf\_token)

        return response

# 获取 CSRF 令牌视图

def csrf\_token\_view(request):

    csrf\_token = get\_token(request)

    return JsonResponse({'csrfToken': csrf\_token})

# 用户个人资料视图

@api\_view(['GET', 'PUT'])

def user\_profile(request):

    if request.method == 'GET':

        user = request.user

        data = {

            "username": user.username,

            "email": user.email,

            "phone": user.phone,

            "department\_name": user.department\_id,

            "position\_name": user.position\_id

        }

        return JsonResponse(data, safe=False)

    if request.method == "PUT":

        user = request.user

        data = json.loads(request.body)

        user.email = data.get("email", user.email)

        user.phone = data.get("phone", user.phone)

        user.department\_id = data.get("department\_id", user.department\_id)

        user.position\_id = data.get("position\_id", user.position\_id)

        user.save()

        return JsonResponse({"success": True})

# 检查登录状态视图

@api\_view(['GET'])

def check\_login\_status(request):

    if request.user.is\_authenticated:

        return JsonResponse({'loggedIn': True}, status=200)

    else:

        return JsonResponse({'loggedIn': False}, status=200)

# 获取用户操作历史视图

@api\_view(['GET'])

def user\_history(request):

    if request.method == 'GET':

        user = request.user

        logs = LogEntry.objects.filter(user\_id=user.id).order\_by('-action\_time')

        history\_data = []

        for log in logs:

            history\_data.append({

                'id': log.id,

                'date': log.action\_time.strftime('%Y-%m-%d %H:%M:%S'),

                'action': log.object\_repr,  # 动作描述

                'changes': log.change\_message,  # 更改的内容

                'user': user.username

            })

        return JsonResponse(history\_data, safe=False)

<template>

  <div>

    <SidebarPage />

    <div class="history-page">

      <div class="container">

        <div v-if="!isLoggedIn" class="no-history centered-content">

          <p>未登入，無紀錄顯示。</p>

        </div>

        <div v-else>

          <div class="search-container centered-content">

            <input type="text" id="searchInput" placeholder="搜尋紀錄..." v-model="searchQuery">

            <button @click="filterTimeline">搜尋</button>

          </div>

          <ul class="timeline centered-content" id="timeline">

            <li class="timeline-item" v-for="item in filteredItems" :key="item.id" @click="showDetail(item.id)">

              <div class="timeline-panel">

                <div class="timeline-heading">

                  <h4>{{ item.date }}</h4>

                </div>

                <div class="timeline-body">

                  <p>{{ item.action }}</p>

                  <p>{{ item.changes }}</p> <!-- 顯示更改的內容 -->

                  <div class="timeline-user">使用者: {{ item.user }}</div>

                </div>

              </div>

            </li>

          </ul>

        </div>

      </div>

    </div>

  </div>

</template>

<script>

import SidebarPage from '@/components/frontend/SidebarPage.vue';

import axios from 'axios';

export default {

  name: 'HistoryPage',

  components: {

    SidebarPage

  },

  data() {

    return {

      searchQuery: '',

      items: [],

      isLoggedIn: false,

    }

  },

  computed: {

    filteredItems() {

      return this.items.filter(item => {

        const query = this.searchQuery.trim().toUpperCase();

        return item.action.toUpperCase().includes(query) || item.date.includes(query);

      });

    }

  },

  methods: {

    filterTimeline() {

      // 使用计算属性进行过滤，不需要额外的过滤逻辑

    },

    showDetail(id) {

      this.$router.push(`/detail/${id}`);

    },

    async checkLoginStatus() {

      try {

        const response = await axios.get('/api/frontend/check\_login\_status/');

        this.isLoggedIn = response.data.loggedIn;

        if (this.isLoggedIn) {

          this.fetchHistory();

        } else {

          this.$router.push('/login');

        }

      } catch (error) {

        console.error('無法檢查登入狀態:', error);

      }

    },

    async fetchHistory() {

      try {

        const response = await axios.get('/api/frontend/history/');

        this.items = response.data;

      } catch (error) {

        console.error('無法獲取歷史記錄:', error);

      }

    }

  },

  mounted() {

    this.checkLoginStatus();

  }

}

</script>

<style scoped>

.history-page {

  display: flex;

  justify-content: center;

  align-items: center;

  height: 100vh;

}

.container {

  width: 100%;

  max-width: 800px;

  margin: 0 auto;

  padding: 20px;

  text-align: center;

}

.centered-content {

  display: flex;

  justify-content: center;

  align-items: center;

  flex-direction: column;

}

.no-history {

  font-size: 18px;

  color: #555;

}

.search-container {

  margin-bottom: 20px;

}

.timeline {

  list-style: none;

  padding: 0;

}

.timeline-item {

  margin-bottom: 20px;

}

.timeline-panel {

  background: #f8f9fa;

  border-radius: 5px;

  padding: 15px;

  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

  text-align: left;

}

</style>