```
README.md
This directory contains data from the pytest's cache plugin,
which provides the `--lf` and `--ff` options, as well as the `cache` fixture.
See [the docs](https://docs.pytest.org/en/stable/how-to/cache.html) for more information.
forms.py
from flask_login import current_user
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField
from wtforms.validators import DataRequired, Email, Length, Optional, ValidationError, EqualTo
from app.models.models import User
class LoginForm(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Email(message='请输入有效的邮箱地址')])
    password = PasswordField('Password', validators=[Optional(), Length(min=8, message='密码至少 8 个字符')])
    submit = SubmitField('Login')
    def validate(self):
        if not super(LoginForm, self).validate():
            return False
        if not self.password.data:  # 不需要检查验证码字段，因为这是密码登录
            self.password.errors.append('请输入密码')
            return False
        return True
class CodeLoginForm(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Email(message='请输入有效的邮箱地址')])
    verification_code = StringField('Verification Code', validators=[DataRequired(message='请输入验证码')])
    submit = SubmitField('Login')
def validate_email(form, field):
    user = User.query.filter_by(email=field.data).first()
    if user:
        raise ValidationError('该邮箱已被注册')
class RegistrationForm(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Email(message='请输入有效的邮箱地址'), validate_email])
    verification_code = StringField('Verification Code', validators=[Optional()])
    password = PasswordField('Password', validators=[Optional(), Length(min=8, message='密码至少 8 个字符')])
    submit = SubmitField('Register')
    send_code = SubmitField('Send Verification Code')
class ResetPasswordForm(FlaskForm):
    password = PasswordField('新密码', validators=[DataRequired(), Length(min=8, message="密码至少 8 个字符")])
    confirm_password = PasswordField('确认新密码', validators=[DataRequired(), EqualTo('password', message="密码不匹配
")])
    submit = SubmitField('重置密码')
class RequestResetForm(FlaskForm):
    email = StringField('Email', validators=[DataRequired(), Email(message='请输入有效的邮箱地址')])
    submit = SubmitField('Request Password Reset')
class ChangePasswordForm(FlaskForm):
    old_password = PasswordField('Old Password', validators=[DataRequired()])
    new_password = PasswordField('New Password', validators=[DataRequired(), Length(min=8, message='密码至少 8 个字符
')])
    confirm_password = PasswordField('Confirm New Password', validators=[DataRequired(), EqualTo('new_password',
message='密码不匹配，请重新输入')])
    submit = SubmitField('Change Password')
class ChangeEmailForm(FlaskForm):
    email = StringField('New Email', validators=[DataRequired(), Email(message='请输入有效的邮箱地址'), validate_email])
    verification_code = StringField('Verification Code', validators=[Optional()])
    submit = SubmitField('Change Email')
    send_code = SubmitField('Send Verification Code')
class DeleteAccountForm(FlaskForm):
    password = PasswordField('Password', validators=[DataRequired()])
    verification_code = StringField('Verification Code', validators=[DataRequired(), Length(min=6, max=6, message='验证码应
为 6 位')])
    submit = SubmitField('Delete Account')
routes.py
import logging
import random
import string
import time
from datetime import timedelta
from email.utils import formataddr
from random import randint
from flask_jwt_extended import jwt_required, get_jwt_identity, create_access_token, verify_jwt_in_request, decode_token
from flask_jwt_extended.exceptions import NoAuthorizationError
from flask_mail import Message
from jwt import ExpiredSignatureError, InvalidTokenError
from werkzeug.security import check_password_hash, generate_password_hash
from . import auth_bp
from flask import render_template, redirect, url_for, flash, request, session, current_app, jsonify
from flask_login import login_user, logout_user, login_required, current_user
from app.models.models import User
from ..extensions import db, mail, login_manager
from .forms import LoginForm, RegistrationForm, ResetPasswordForm, ChangePasswordForm, ChangeEmailForm, \
    DeleteAccountForm, CodeLoginForm, RequestResetForm
from functools import wraps
def manage_verification_code(email, operation="generate", input_code=None):
    """
    Handles verification code generation, storage, and validation.
    :param email: User's email to send the verification code.
    :param operation: Either 'generate' or 'validate'.
    :param input_code: Used in 'validate' mode to check the input against the stored code.
    :return: Boolean (for validation) or None (for generation).
    """
    if operation == "generate":
        verification_code = generate_verification_code()
        session['verification_code'] = verification_code
        session['last_code_sent'] = time.time()
        return send_verification_email(email, verification_code)
    elif operation == "validate":
        return verify_code(input_code)
def generate_verification_code(length=6):
    """Generates a random numeric verification code of a specified length."""
    return ''.join(random.choices(string.digits, k=length))
def generate_reset_token(user, expires_sec=3600):
    token = create_access_token(identity=user.id, expires_delta=timedelta(seconds=expires_sec))
    return token
def send_verification_email(email, code):
    """Sends an email with the verification code to the user's email address."""
    try:
        msg = Message('lanshi 云盘:您的验证码', recipients=[email])
        msg.body = f'您的验证码是 {code}'
        mail.send(msg)
        return True
    except Exception as e:
        current_app.logger.error(f'发送邮件时出错: {e}')
        return False
def verify_code(input_code):
    """Verifies if the input code matches the stored session code."""
    stored_code = session.get('verification_code')
    return stored_code and input_code == stored_code
def is_cooldown_active():
    """Checks if the cooldown period for sending a new verification code is still active."""
    last_sent = session.get('last_code_sent', 0)
    current_time = time.time()
    cooldown_time = 60  # 冷却时间
    if current_time - last_sent < cooldown_time:
        return True, cooldown_time - int(current_time - last_sent)
    return False, None
def handle_db_operation(operation):
    """Handle database operation with error handling."""
    try:
        operation()
        db.session.commit()
    except Exception as e:
        db.session.rollback()
        current_app.logger.error(f"Database operation error: {e}")
        return False
    return True
def handle_user_login(user, form=None, use_code=False):
    """Unified user login handler for password and code login."""
    if use_code:
        if not verify_code(form.verification_code.data):
            flash('验证码错误', 'danger')
            return redirect(url_for('auth.email_login'))
    else:
        if not check_password_hash(user.password_hash, form.password.data):
            flash('邮箱或密码错误', 'danger')
            return redirect(url_for('auth.login'))
    login_user(user)
    access_token = create_access_token(identity=user.id)
    flash('登录成功', 'success')
    return redirect(url_for('main.index'))
def cooldown_required(func):
    """Decorator to enforce cooldown period for sending verification code."""
    @wraps(func)
    def wrapper(*args, **kwargs):
        is_cooldown, time_left = is_cooldown_active()
        if is_cooldown:
            return jsonify({'success': False, 'message': f'请等待 {time_left} 秒后再请求验证码。'}), 429
        return func(*args, **kwargs)
    return wrapper
```

```
@auth_bp.route('/register', methods=['GET', 'POST'])
def register():
    form = RegistrationForm()
    if form.validate_on_submit():
        email = form.email.data
        password_hash = generate_password_hash(form.password.data)
        if form.send_code.data:
            manage_verification_code(email, operation="generate")
            flash('验证码已发送，请检查您的邮箱。', 'info')
            return redirect(url_for('auth.register'))
        if form.submit.data:
            if verify_code(form.verification_code.data):
                user = User(email=email, password_hash=password_hash)
                success = handle_db_operation(lambda: db.session.add(user))
                if success:
                    session.pop('verification_code', None)
                    login_user(user)
                    flash('注册成功，请登录。', 'success')
                    return redirect(url_for('main.index'))
            else:
                flash('验证码错误，请重新输入。', 'danger')
    return render_template('register.html', form=form)
@auth_bp.route('/send_code', methods=['POST'])
@cooldown_required
def send_code():
    email = request.form.get('email')
    if not email:
        return jsonify({'success': False, 'message': '邮箱地址是必需的。'}), 400
    manage_verification_code(email, operation="generate")
    return jsonify({'success': True, 'message': '验证码已发送，请检查您的邮箱。'})
@auth_bp.route('/email_login', methods=['GET', 'POST'])
def email_login():
    form = CodeLoginForm()
    if form.validate_on_submit():
        email = form.email.data
        user = User.query.filter_by(email=email).first()
        if not user:
            flash('用户不存在', 'danger')
            return redirect(url_for('auth.email_login'))
        return handle_user_login(user, form, use_code=True)
    return render_template('code_login.html', form=form)
@auth_bp.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        email = form.email.data
        user = User.query.filter_by(email=email).first()
        if user:
            return handle_user_login(user, form, use_code=False)
    return render_template('login.html', form=form)
@auth_bp.route('/protected', methods=['GET'])
@jwt_required()
```

```
def protected():
    current_user_id = get_jwt_identity()
    return jsonify(logged_in_as=current_user_id), 200
def send_reset_email(user):
    """发送密码重置邮件"""
    token = generate_reset_token(user)
    reset_url = url_for('auth.reset_password', token=token, _external=True)
    sender_name = "lanshi 云盘"
    sender_email = current_app.config['MAIL_USERNAME']
    msg = Message('重置密码请求',
                  sender=formataddr((sender_name, sender_email)),
                  recipients=[user.email])
    msg.body = f'''要重置您的密码，请访问以下链接：
{reset_url}
如果您没有请求此操作，请忽略此邮件，无需采取进一步操作。
'''
    mail.send(msg)
@auth_bp.route('/reset_password_request', methods=['GET', 'POST'])
def reset_password_request():
    if current_user.is_authenticated:
        return redirect(url_for('main.index'))
    form = RequestResetForm()
    if form.validate_on_submit():
        user = User.query.filter_by(email=form.email.data).first()
        if user:
            send_reset_email(user)
        flash('如果该邮箱存在，重置密码链接将发送到该邮箱中。', 'info')
        return redirect(url_for('auth.login'))
    return render_template('reset_password_request.html', title='重置密码', form=form)
@auth_bp.route('/reset_password/<token>', methods=['GET', 'POST'])
def reset_password(token):
    if current_user.is_authenticated:
        return redirect(url_for('main.index'))
    try:
        decoded_token = decode_token(token)
        user_id = decoded_token['sub']
        user = User.query.get(user_id)
    except ExpiredSignatureError:
        flash('该重置密码链接已过期。', 'warning')
        return redirect(url_for('auth.reset_password_request'))
    except InvalidTokenError:
        flash('该重置密码链接无效。', 'warning')
        return redirect(url_for('auth.reset_password_request'))
    if not user:
        flash('无效的用户。', 'danger')
        return redirect(url_for('auth.reset_password_request'))
    form = ResetPasswordForm()
    if form.validate_on_submit():
        print("表单验证成功")
        hashed_password = generate_password_hash(form.password.data)
        user.password_hash = hashed_password
        try:
            db.session.commit()
        except Exception as e:
            db.session.rollback()
            print(f"Error while updating password: {e}")
            flash('更新密码时发生错误，请稍后再试。', 'danger')
        flash('您的密码已更新！您现在可以登录了。', 'success')
        return redirect(url_for('auth.login'))
    else:
        print("表单验证失败: ", form.errors)
    return render_template('reset_password.html', form=form,token=token)
@auth_bp.route('/change_email', methods=['GET', 'POST'])
@login_required
def change_email():
    form = ChangeEmailForm()
    if form.validate_on_submit():
        if form.send_code.data:
            manage_verification_code(form.email.data, operation="generate")
            flash('验证码已发送，请查收您的电子邮件。', 'info')
        elif form.submit.data and verify_code(form.verification_code.data):
            current_user.email = form.email.data
            db.session.commit()
            flash('您的邮箱已更新，请重新登录！', 'success')
            logout_user()
            return redirect(url_for('auth.login'))
    return render_template('change_email.html', form=form)
@auth_bp.route('/delete_account', methods=['GET', 'POST'])
@login_required
def delete_account():
    form = DeleteAccountForm()
    if form.validate_on_submit():
        if verify_code(form.verification_code.data):
            success = handle_db_operation(lambda: db.session.delete(current_user))
            if success:
                logout_user()
                flash('您的账户已成功删除。', 'success')
                return redirect(url_for('auth.login'))
        flash('验证码错误，请重新输入。', 'danger')
    return render_template('delete_account.html', form=form)
@auth_bp.route('/change_password', methods=['GET', 'POST'])
@login_required
def change_password():
    form = ChangePasswordForm()
    if form.validate_on_submit():
        if check_password_hash(current_user.password_hash, form.old_password.data):
            current_user.set_password(form.new_password.data)
            success = handle_db_operation(lambda: None)
            if success:
                flash('密码已更改，请重新登录。', 'success')
                logout_user()
                return redirect(url_for('auth.login'))
        flash('旧密码错误。', 'danger')
    return render_template('change_password.html', form=form)
```

```
@auth_bp.route('/logout')
@login_required
def logout():
   logout_user()
   flash('您已成功登出。', 'info')
   return redirect(url_for('auth.login'))
@auth_bp.route('/user_center', methods=['GET'])
@login_required
def user_center():
   return render_template('user_center.html')
__init__.py
from flask import Blueprint
auth_bp = Blueprint('auth', __name__, template_folder='templates')
from . import routes
config.py
import os
from datetime import timedelta
class Config:
   SECRET_KEY = os.environ.get('SECRET_KEY') or os.urandom(24).hex()
   JWT_SECRET_KEY = os.environ.get('JWT_SECRET_KEY', os.urandom(24).hex())
   JWT_ALGORITHM = 'HS256'
   JWT_ACCESS_TOKEN_EXPIRES = timedelta(hours=1)
   SQLALCHEMY_DATABASE_URI = os.environ.get('DATABASE_URL')
   SQLALCHEMY_TRACK_MODIFICATIONS = False
   UPLOAD_FOLDER = os.path.join(os.getcwd(), 'uploads')
   ALLOWED_EXTENSIONS = {'txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif', 'doc', 'docx', 'ppt', 'pptx', 'xls', 'xlsx', 'zip',
                  'rar', '7z', 'gz', 'tar', 'mp4', 'avi', 'mkv', 'flv', 'mov', 'wmv', 'mp3', 'wav', 'flac',
                  'wma',
                  'aac', 'ogg', 'csv', 'json', 'xml', 'html', 'css', 'js', 'py', 'c', 'cpp', 'java', 'md',
                  'yml',
                  'yaml', 'toml', 'ini', 'cfg', 'conf', 'log', 'sql', 'db', 'sqlite', 'psd', 'ai', 'svg', 'eps',
                  'ttf', 'otf', 'woff', 'woff2', 'eot', 'apk', 'exe', 'dmg', 'iso', 'img', 'bin', 'dll', 'deb',
                  'rpm', 'sh', 'bat', 'cmd', 'vbs', 'ps1', 'reg', 'jar'}
   MAX_CONTENT_LENGTH = 1024 * 1024 * 1024  # 1024 MB
   MAIL_SERVER = 'smtp.vip.163.com'
   MAIL_PORT = 465
   MAIL_USE_SSL = True
   MAIL_USERNAME = os.environ.get('MAIL_USERNAME')
   MAIL_PASSWORD = os.environ.get('MAIL_PASSWORD')
   MAIL_DEFAULT_SENDER = MAIL_USERNAME
   from flask import make_response
class DevelopmentConfig(Config):
   DEBUG = True
class ProductionConfig(Config):
   DEBUG = False
class TestConfig(Config):
   TESTING = True
   SQLALCHEMY_DATABASE_URI = 'sqlite:///:memory:'  # Use in-memory database for tests
   WTF_CSRF_ENABLED = False  # Typically disable CSRF protection for tests
extensions.py
from flask import current_app
from flask_migrate import Migrate
from flask_sqlalchemy import SQLAlchemy
from flask_login import LoginManager
from flask_mail import Mail, Message
from itsdangerous import URLSafeTimedSerializer as Serializer
from flask_wtf import CSRFProtect
db = SQLAlchemy()
migrate = Migrate()
login_manager = LoginManager()
login_manager.login_view = 'auth.login'
mail = Mail()
csrf = CSRFProtect()
def init_extensions(app):
    db.init_app(app)
    migrate.init_app(app, db)
    login_manager.init_app(app)
    mail.init_app(app)
    csrf.init_app(app)
    login_manager.user_loader(load_user)
def load_user(user_id):
    from app.models.models import User  # 防止循环导入
    return User.query.get(int(user_id))
def generate_token(email):
    serializer = Serializer(current_app.config['SECRET_KEY'])
    return serializer.dumps(email, salt='email-salt')
def confirm_token(token, expiration=600):
    serializer = Serializer(current_app.config['SECRET_KEY'])
    try:
        email = serializer.loads(token, salt='email-confirm', max_age=expiration)
    except:
        return False
    return email
def send_verification_email(email, code):
    msg = Message('Your Verification Code', recipients=[email])
    msg.body = f'Your verification code is: {code}'
    mail.send(msg)
```

forms.py

```
from flask import flash, url_for, redirect
from flask_wtf import FlaskForm
from flask_wtf.file import FileField, FileRequired
from wtforms import SubmitField
from wtforms.fields.simple import StringField
from wtforms.validators import DataRequired
class UploadForm(FlaskForm):
    files = FileField('Upload Files', validators=[FileRequired()])
    submit = SubmitField('Upload')
class ShareFileForm(FlaskForm):
    filename = StringField('Filename', validators=[DataRequired()])
    submit = SubmitField('Generate Share Link')
class SaveSharedFileForm(FlaskForm):
    share_token = StringField('分享链接或代码', validators=[DataRequired()])
```

routes.py

```
import logging
import os
import secrets
import re
from collections import Counter
from urllib.parse import unquote
import jieba
from pypinyin import lazy_pinyin
from os.path import join
from flask import (
   Blueprint, current_app, render_template, redirect, url_for, flash,
   jsonify, send_file, request
)
from flask_login import login_required, current_user
from sqlalchemy import func
from werkzeug.utils import secure_filename
from ..extensions import db
from app.main import main_bp
from app.models.file import File
from app.main.forms import UploadForm, ShareFileForm, SaveSharedFileForm
from ..utils.utils import format_file_size
def allowed_file(filename):
   return '.' in filename and filename.rsplit('.', 1)[1].lower() in current_app.config['ALLOWED_EXTENSIONS']
def delete_file(file_id):
   file_record = File.query.filter_by(id=file_id, user_id=current_user.id).first()
   if not file_record:
      return False, "文件不存在或无权删除该文件"
   file_path = file_record.get_full_path()
   if os.path.exists(file_path):
      try:
         os.remove(file_path)
         db.session.delete(file_record)
         db.session.commit()
         return True, "文件删除成功"
      except Exception as e:
         db.session.rollback()
         return False, f"文件删除失败: {str(e)}"
   else:
      return False, "文件不存在"
def custom_secure_filename(filename):
   """
   Custom version of secure_filename to keep Chinese characters and file extensions.
   """
   filename = filename.strip().replace(' ', '_')  # Replace spaces with underscores
   name, ext = filename.rsplit('.', 1) if '.' in filename else (filename, '')
   name = re.sub(r'[^\w\u4e00-\u9fff]', '_', name)
   if ext:
      return f"{name}.{ext}"
   return name
@main_bp.route('/')
def index():
   return render_template('index.html')
@main_bp.context_processor
def utility_processor():
   return dict(format_file_size=format_file_size)
@main_bp.route('/upload', methods=['GET', 'POST'])
```

```
@login_required
def upload_file():
    form = UploadForm()
    if request.method == 'POST':
        try:
            if 'files' not in request.files:
                return jsonify(success=False, message="没有文件上传"), 400
            files = request.files.getlist('files')
            upload_results = []
            for file in files:
                if file and allowed_file(file.filename):
                    filename = custom_secure_filename(file.filename)
                    file_path = os.path.join(current_app.config['UPLOAD_FOLDER'], filename)
                    if File.query.filter_by(filename=filename, user_id=current_user.id).first():
                        upload_results.append({'file': filename, 'success': False, 'message': '此文件已存在！'})
                        continue
                    file.save(file_path)
                    new_file = File(filename=filename, user_id=current_user.id, path=file_path)
                    db.session.add(new_file)
                    db.session.commit()
                    upload_results.append({'file': filename, 'success': True, 'message': '文件上传成功！'})
                else:
                    upload_results.append(
                        {'file': file.filename, 'success': False, 'message': '无效文件格式，请检查格式！'})
            return jsonify(success=True, upload_results=upload_results), 200
        except Exception as e:
            db.session.rollback()
            return jsonify(success=False, message=f"服务器错误: {str(e)}"), 500
    return render_template('upload.html', form=form)
@main_bp.route('/share/<int:file_id>', methods=['POST'])
@login_required
def share_file(file_id):
    file = File.query.filter_by(id=file_id, user_id=current_user.id).first()
    if not file:
        return jsonify({'success': False, 'message': '文件不存在！'}), 404
    if not file.share_token:
        file.share_token = secrets.token_urlsafe(16)
        db.session.commit()
    share_link = url_for('main.download_file', file_id=file.id, share_token=file.share_token, _external=True)
    return jsonify({'success': True, 'share_link': share_link})
@main_bp.route('/batch_share_file', methods=['GET', 'POST'])
@login_required
def batch_share_file():
    data = request.get_json()
    file_ids = data.get('file_ids')
    if not file_ids:
        return jsonify(success=False, message="没有文件被选中")
    try:
        for file_id in file_ids:
            file_record = File.query.filter_by(id=file_id, user_id=current_user.id).first()
            if file_record:
                if not file_record.share_token:
```

```
            file_record.share_token = secrets.token_urlsafe(16)
            db.session.commit()
      db.session.commit()
      return jsonify(success=True, message="文件分享成功")
   except Exception as e:
      db.session.rollback()
      return jsonify(success=False, message=f"文件分享失败: {str(e)}")
@main_bp.route('/save_shared_file', methods=['GET', 'POST'])
@login_required
def save_shared_file():
   form = SaveSharedFileForm()
   if request.method == 'POST':
      if form.validate_on_submit():
         share_token = form.share_token.data
         if 'share_token=' in share_token:
            share_token = share_token.split('share_token=')[1]
         shared_file = File.query.filter_by(share_token=share_token).first()
         if shared_file:
            new_file = File(
               filename=shared_file.filename,
               user_id=current_user.id,
               path=shared_file.path
            )
            db.session.add(new_file)
            db.session.commit()
            flash('文件已成功保存到您的目录中！', 'success')
            return redirect(url_for('main.file_list'))
         else:
            flash('无效的分享令牌，无法保存文件。', 'danger')
      else:
         print("表单验证失败")
         print(form.errors)
   return render_template('save_shared_file.html', form=form)
@main_bp.route('/download/<int:file_id>', methods=['GET'])
@login_required
def download_file(file_id):
   file = File.query.filter_by(id=file_id, user_id=current_user.id).first()
   if not file:
      return jsonify({'success': False, 'message': '文件不存在'}), 404
   file.download_count += 1
   db.session.commit()
   return send_file(file.path, as_attachment=True, download_name=file.filename)
@main_bp.route('/get_files')
@login_required
def get_files():
   page = request.args.get('page', 1, type=int)
   per_page = 10
   pagination = File.query.filter_by(user_id=current_user.id).paginate(page, per_page, error_out=False)
   files = pagination.items
   file_list = [
      {
         'id': file.id,
            'filename': file.filename,
            'size': file.get_formatted_size(),
            'created_at': file.created_at.strftime("%Y-%m-%d %H:%M:%S"),
            'download_count': file.download_count,
        }
        for file in files
    ]
    return jsonify({
        'files': file_list,
        'has_next': pagination.has_next,
        'has_prev': pagination.has_prev,
        'page': pagination.page,
        'total_pages': pagination.pages
    })
@main_bp.route('/files', methods=['GET', 'POST'])
@login_required
def file_list():
    if request.method == 'POST':
        file_id = request.form.get('file_id')
        success, message = delete_file(file_id)
        if success:
            flash(message, 'success')
        else:
            flash(message, 'danger')
        return redirect(url_for('main.file_list'))
    page = request.args.get('page', 1, type=int)
    per_page = 10
    files = File.query.filter_by(user_id=current_user.id).paginate(page, per_page, error_out=False)
    return render_template('file_list.html', files=files.items, pagination=files)
@main_bp.route('/batch_delete', methods=['POST'])
@login_required
def batch_delete_files():
    data = request.get_json()
    file_ids = data.get('file_ids')
    if not file_ids:
        return jsonify(success=False, message="没有文件被选中")
    try:
        for file_id in file_ids:
            file_record = File.query.filter_by(id=file_id, user_id=current_user.id).first()
            if file_record:
                file_path = os.path.join(current_app.config['UPLOAD_FOLDER'], file_record.filename)
                if os.path.exists(file_path):
                    os.remove(file_path)
                db.session.delete(file_record)
        db.session.commit()
        flash("文件删除成功！", "success")
        return jsonify(success=True, message="文件删除成功")
    except Exception as e:
        db.session.rollback()
        flash("删除文件失败！", "danger")
        return jsonify(success=False, message=f"删除文件失败: {str(e)}")
def delete_file_from_filesystem(file_record):
    file_path = os.path.join(current_app.config['UPLOAD_FOLDER'], file_record.filename)
    if os.path.exists(file_path):
        try:
            os.remove(file_path)
        except Exception as e:
            return False, f'文件删除失败: {str(e)}'
    return True, '文件删除成功！ '
@main_bp.route('/delete/<int:file_id>', methods=['POST'])
@login_required
def delete_file(file_id):
    current_app.logger.info(f"正在删除文件 ID: {file_id}")
    file_record = File.query.filter_by(id=file_id, user_id=current_user.id).first()
    if not file_record:
        return jsonify({'success': False, 'message': '文件不存在或无权删除该文件'}), 404
    file_path = file_record.get_full_path()
    if os.path.exists(file_path):
        try:
            os.remove(file_path)  # 删除文件
            db.session.delete(file_record)  # 删除数据库记录
            db.session.commit()
            return jsonify({'success': True, 'message': '文件删除成功'}), 200
        except Exception as e:
            db.session.rollback()
            current_app.logger.error(f'文件删除失败: {str(e)}')
            return jsonify({'success': False, 'message': f'文件删除失败: {str(e)}'}), 500
    else:
        return jsonify({'success': False, 'message': '文件不存在'}), 404
@main_bp.route('/stats')
@login_required
def view_stats():
    stats = db.session.query(
        func.count(File.id).label('total_files'),
        func.sum(File.download_count).label('total_downloads'),
        func.max(File.size).label('largest_file_size')
    ).first()
    largest_file = File.query.order_by(File.size.desc()).first()
    largest_file_name = largest_file.filename if largest_file else 'N/A'
    largest_file_size = largest_file.size if largest_file else 0
    return render_template('stats.html', stats={
        'total_files': stats.total_files or 0,
        'total_downloads': stats.total_downloads or 0,
        'largest_file': largest_file_name,
        'largest_file_size': largest_file_size
    })
def process_file_names(files):
    all_words = []
    for file in files:
        file_name = file.filename.rsplit('.', 1)[0]
        words = [word for word in jieba.cut(file_name) if word.isalpha()]
        all_words.extend(words)
    return all_words
@main_bp.route('/upload_stats')
@login_required
```

```
def upload_stats():
    files = File.query.filter_by(user_id=current_user.id).all()
    all_words = process_file_names(files)
    word_freq = Counter(all_words)
    wordcloud_data = [{"name": word, "value": freq} for word, freq in word_freq.items()]
    monthly_uploads = {month: 0 for month in
                    ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']}
    for file in files:
        upload_month = file.created_at.strftime('%b')
        if upload_month in monthly_uploads:
            monthly_uploads[upload_month] += 1
    return jsonify({
        'monthly_uploads': monthly_uploads,
        'wordcloud_data': wordcloud_data
    })
__init__.py
from flask import Blueprint
main_bp = Blueprint('main', __name__)
from . import routes  # Ensure this import is at the end of the file
errors.py
from flask import render_template
def init_app(app):
    @app.errorhandler(404)
    def not_found_error(error):
        return render_template('404.html'), 404
    @app.errorhandler(500)
    def internal_error(error):
        from ..extensions import db
        db.session.rollback()
        return render_template('500.html'), 500
file.py
import os
from datetime import datetime
from os.path import getsize
from flask import current_app
from ..extensions import db
from ..utils.utils import format_file_size
class File(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    filename = db.Column(db.String(256), nullable=False)
    size = db.Column(db.Integer, nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
    created_at = db.Column(db.DateTime, default=datetime.utcnow)
    user = db.relationship('User', back_populates='files', lazy=True)
    share_token = db.Column(db.String(32), unique=True, nullable=True)
    download_count = db.Column(db.Integer, default=0)
    path = db.Column(db.String(256), nullable=False)
    def __init__(self, filename, path, user_id, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.filename = filename
        self.path = path
        self.user_id = user_id
        self.ensure_directory_exists()  # Ensure directory exists
        self.size = self.calculate_size()
    def get_formatted_size(self):
        return format_file_size(self.size)
    def calculate_size(self):
        """Calculate the size of the file and store it in the `size` attribute."""
        file_path = self.get_full_path()
        return getsize(file_path) if os.path.exists(file_path) else 0
    def get_full_path(self):
        """Get the full absolute path of the file."""
        return os.path.abspath(os.path.join(current_app.config['UPLOAD_FOLDER'], self.path))
    def ensure_directory_exists(self):
        """Ensure the upload directory exists."""
        full_path = os.path.dirname(self.get_full_path())
        if not os.path.exists(full_path):
            os.makedirs(full_path)
    def delete_file(self):
        """Delete the file from the file system."""
        file_path = self.get_full_path()
        if os.path.exists(file_path):
            os.remove(file_path)
    def __repr__(self):
        return f"<File(filename='{self.filename}', size='{self.get_formatted_size()}', user_id={self.user_id},
created_at={self.created_at})>"
```

models.py

```
from datetime import datetime
from werkzeug.security import generate_password_hash, check_password_hash
from flask_login import UserMixin
from ..extensions import db
class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password_hash = db.Column(db.String(128), nullable=False)
    is_active = db.Column(db.Boolean, default=True)
    created_at = db.Column(db.DateTime, default=datetime.utcnow)
    verification_code = db.Column(db.String(6))
    verification_sent_at = db.Column(db.DateTime)
    files = db.relationship('File', back_populates='user', lazy=True, cascade="all, delete-orphan")
    def set_password(self, password):
        self.password_hash = generate_password_hash(password)
    def check_password(self, password):
        return check_password_hash(self.password_hash, password)
    def set_verification_code(self, code):
        self.verification_code = code
        self.verification_sent_at = datetime.utcnow()  # 记录发送时间
    def check_verification_code(self, code, expiration=600):
        if self.verification_code != code:
            return False
        if (datetime.utcnow() - self.verification_sent_at).total_seconds() > expiration:
            return False
        return True
```

__init__.py

custom.css

```
.card-container {
  display: flex;
  justify-content: space-around;
  flex-wrap: wrap;
  margin: 30px 0;
  gap: 20px;
}
.card {
  background-color: #fff;
  border-radius: 12px;
  padding: 40px 20px;
  text-align: center;
  box-shadow: 0 8px 20px rgba(0, 0, 0, 0.1);
  transition: transform 0.3s ease, box-shadow 0.3s ease;
  max-width: 250px;
  flex: 1;
}
.card:hover {
  transform: translateY(-8px);
  box-shadow: 0 12px 24px rgba(0, 0, 0, 0.15);
}
.card h2 {
  font-size: 1.4rem;
  margin-bottom: 15px;
  font-weight: 600;
  color: #333;
}
.card p {
  font-size: 1rem;
  color: #666;
  margin-bottom: 20px;
}
.card-icon img {
  width: 50px;
  margin-bottom: 20px;
}
.btn {
  display: inline-block;
  padding: 10px 20px;
  color: white;
  text-decoration: none;
  border-radius: 25px;
  font-size: 0.9rem;
  transition: background-color 0.3s ease, transform 0.3s ease;
}
.btn:hover {
  transform: translateY(-2px);
}
.btn-primary, .btn-info, .btn-danger {
  padding: 12px 24px;
  border-radius: 30px;
}
.btn-primary {
  background-color: #1a73e8;
  border-color: #1a73e8;
}
.btn-primary:hover {
  background-color: #155bb5;
}
.btn-info {
  background-color: #17a2b8;
  border-color: #17a2b8;
}
.btn-info:hover {
  background-color: #138496;
}
.btn-danger {
  background-color: #dc3545;
  border-color: #dc3545;
}
.btn-danger:hover {
  background-color: #c82333;
}
.hero-section {
  text-align: center;
  padding: 60px 20px;
  background-color: #f0f4f8;
}
.hero-title {
  font-size: 2.5rem;
  font-weight: 700;
  color: #333;
  margin-bottom: 20px;
}
.hero-subtitle {
  font-size: 1.2rem;
  color: #555;
  margin-bottom: 40px;
}
.form-control {
  border-radius: 6px;
  padding: 12px;
  font-size: 1rem;
  margin-bottom: 20px;
}
.form-label {
  font-weight: 600;
  font-size: 1.1rem;
}
@media (max-width: 768px) {
  .card-container {
    flex-direction: column;
    align-items: center;
  }
  .card {
    width: 100%;
```

```
        max-width: 400px;
    }
}
```

style.css

```
body {
    font-family: 'Helvetica Neue', Arial, sans-serif;
    background-color: #f5f7fa;
    margin: 0;
    padding: 0;
}
.container {
    max-width: 1200px;
    margin: 0 auto;
    padding: 20px;
}
.header {
    display: flex;
    align-items: center;
    justify-content: space-between;
    padding: 20px 0;
    background-color: #fff;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}
.header img {
    height: 50px;
}
.header h1 {
    font-size: 1.5rem;
    font-weight: 700;
    margin: 0;
}
.header a {
    text-decoration: none;
    color: #1a73e8;
    margin-right: 20px;
    font-weight: bold;
}
.footer {
    text-align: center;
    padding: 20px 0;
    color: #aaa;
}
.footer a {
    color: #1a73e8;
    text-decoration: none;
}
.footer img {
    width: 150px;
    margin-top: 10px;
}
```

404.html

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

```
<div class="container mt-5">
  <h2>服务器错误 404</h2>
  <p>抱歉，服务器遇到了错误。请稍后再试。</p>
</div>
{% endblock %}
500.html
{% extends "base.html" %}
{% block content %}
<div class="container mt-5">
  <h2>服务器错误 505</h2>
  <p>抱歉，服务器遇到了错误。请稍后再试。</p>
</div>
{% endblock %}
base.html
<!DOCTYPE html>
<meta charset="UTF-8">
<html lang="zh-CN">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>{% block title %}Lanshi 云盘{% endblock %}</title>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css">
  <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
  <link rel="stylesheet" href="{{ url_for('static', filename='css/custom.css') }}">
  {% block head %}
  {% endblock %}
</head>
<body>
  <nav class="navbar navbar-expand-lg navbar-light bg-light shadow-sm">
    <div class="container-fluid">
      <a class="navbar-brand" href="{{ url_for('main.index') }}">
        <img src="{{ url_for('static', filename='images/logo.png') }}" alt="Lanshi 云盘" width="69" height="26">
        Lanshi 云盘
      </a>
      <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-
controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav ms-auto">
          {% if current_user.is_authenticated %}
            <li class="nav-item">
              <a class="nav-link" href="{{ url_for('main.file_list') }}">我的文件</a>
            </li>
            <li class="nav-item">
              <a class="nav-link" href="{{ url_for('main.upload_file') }}">上传文件</a>
            </li>
            <li class="nav-item dropdown">
              <a class="nav-link dropdown-toggle" href="#" id="userDropdown" role="button" data-bs-toggle="dropdown"
aria-expanded="false">
                {{ current_user.email }}
              </a>
```

```
                <ul class="dropdown-menu dropdown-menu-end" aria-labelledby="userDropdown">
                  <li><a class="dropdown-item" href="{{ url_for('auth.change_password') }}">修改密码</a></li>
                  <li><a class="dropdown-item" href="{{ url_for('auth.change_email') }}">修改邮箱</a></li>
                  <li><a class="dropdown-item" href="{{ url_for('auth.logout') }}">注销账户</a></li>
                  <li><hr class="dropdown-divider"></li>
                  <li><a class="dropdown-item" href="{{ url_for('auth.logout') }}">退出登录</a></li>
                </ul>
            {% else %}
              <li class="nav-item">
                <a class="nav-link" href="{{ url_for('auth.login') }}">登录</a>
              </li>
              <li class="nav-item">
                <a class="nav-link" href="{{ url_for('auth.register') }}">注册</a>
              </li>
            {% endif %}
          </ul>
        </div>
      </div>
    </nav>
    <main class="container mt-5">
      {% block content %}
      {% endblock %}
    </main>
    <footer class="footer bg-light mt-auto py-3">
      <div class="container">
        <p class="text-center mb-0">网站备案号：<a href="https://beian.miit.gov.cn" target="_blank">湘 ICP 备 2023035150
号-1</a></p>
        <p class="text-center mb-0">如有问题,请联系微信公众号</p>
        <div class="text-center">
          <img src="{{ url_for('static', filename='images/wechat_qr.png') }}" alt="WeChat QR Code" width="80">
        </div>
      </div>
    </footer>
    <script>
      document.addEventListener('DOMContentLoaded', function() {
        setTimeout(function() {
          var flashMessages = document.getElementById('flash-messages-container');
          if (flashMessages) {
            var alerts = flashMessages.getElementsByClassName('alert');
            for (var i = 0; i < alerts.length; i++) {
              alerts[i].classList.add('fade-out');
            }
            setTimeout(function() {
              flashMessages.innerHTML = '';
            }, 1000);  // 1 秒的淡化时间
          }
        }, 2000);  // 5 秒后开始淡化
      });
    </script>
    <style>
      .fade-out {
        opacity: 0;
        transition: opacity 1s ease-out;
      }
    </style>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
    {% block scripts %}
    {% endblock %}
</body>
</html>
```

change_email.html

```
{% extends "base.html" %}
{% block content %}
<div class="container mt-5">
  <h2>修改邮箱</h2>
  <a href="javascript:history.back()" class="btn btn-secondary">返回上一层</a>
  <form id="change-email-form" method="POST" action="{{ url_for('auth.change_email') }}">
    {{ form.hidden_tag() }}
    <div class="form-group">
      <label for="email">新邮箱</label>
      {{ form.email(class="form-control") }}
    </div>
    <div class="form-group">
      <label for="verification_code">验证码</label>
      {{ form.verification_code(class="form-control") }}
    </div>
    <button type="button" id="send-code-btn" class="btn btn-secondary">发送验证码</button>
    <button type="submit" name="submit" class="btn btn-primary">修改邮箱</button>
    <p id="countdown-timer" class="mt-2 text-muted" style="display:none;">
          请等待 <span id="timer">60</span> 秒后再发送验证码。
    </p>
  </form>
  <div id="message" class="mt-3"></div>
</div>
<script>
document.getElementById('send-code-btn').addEventListener('click', function() {
  var form = document.getElementById('change-email-form');
  var formData = new FormData(form);
  formData.append('send_code', 'Send Verification Code');
  fetch('{{ url_for("auth.change_email") }}', {
    method: 'POST',
    body: formData,
    headers: {
      'X-CSRFToken': '{{ form.csrf_token._value() }}'
    }
  })
  .then(response => {
    if (!response.ok) {
      throw new Error('Network response was not ok');
    }
    return response.json();
  })
  .then(data => {
    var messageDiv = document.getElementById('message');
```

```
            if (data.success) {
                messageDiv.innerHTML = '<div class="alert alert-info">' + data.message + '</div>';
                startCooldown();
            } else {
                messageDiv.innerHTML = '<div class="alert alert-danger">' + data.message + '</div>';
            }
        })
        .catch(error => {
            console.error('Error:', error);
            var messageDiv = document.getElementById('message');
            messageDiv.innerHTML = '<div class="alert alert-danger">发生错误,请稍后再试!</div>';
        });
        function startCooldown() {
            let timeLeft = 60;
            document.getElementById('countdown-timer').style.display = 'block';
            const timerSpan = document.getElementById('timer');
            timerSpan.innerText = timeLeft;
            let sendCodeBtn = document.getElementById('send-code-btn');
            sendCodeBtn.innerText = `等待 ${timeLeft}s`;
            const countdown = setInterval(() => {
                if (--timeLeft <= 0) {
                    clearInterval(countdown);
                    let sendCodeBtn = document.getElementById('send-code-btn');
                    sendCodeBtn.disabled = false;
                    sendCodeBtn.innerText = '发送验证码';
                    document.getElementById('countdown-timer').style.display = 'none';
                } else {
                    timerSpan.innerText = timeLeft;
                    sendCodeBtn.innerText = `等待 ${timeLeft}s`;
                }
            }, 1000);
        }
    });
    document.getElementById('change-email-form').addEventListener('submit', function(event) {
        event.preventDefault();
        var form = event.target;
        var formData = new FormData(form);
        formData.append('submit', 'Change Email');
        fetch('{{ url_for("auth.change_email") }}', {
            method: 'POST',
            body: formData,
            headers: {
                'X-CSRFToken': '{{ form.csrf_token._value() }}'
            }
        })
        .then(response => {
            if (!response.ok) {
                throw new Error('Network response was not ok');
            }
            return response.json();
        })
        .then(data => {
            var messageDiv = document.getElementById('message');
        if (data.success) {
          messageDiv.innerHTML = '<div class="alert alert-info">' + data.message + '</div>';
          alert('邮箱修改成功!');
          window.location.reload();
          form.reset();
        } else {
          messageDiv.innerHTML = '<div class="alert alert-danger">' + data.message + '</div>';
        }
      })
      .catch(error => {
        console.error('Error:', error);
        var messageDiv = document.getElementById('message');
        messageDiv.innerHTML = '<div class="alert alert-danger">发生错误,请稍后再试!</div>';
      });
    });
</script>
{% endblock %}
change_password.html
{% extends "base.html" %}
{% block content %}
<div class="container mt-5">
  <h2>更改密码</h2>
  <a href="javascript:history.back()" class="btn btn-secondary">返回上一层</a>
  <form method="POST" action="{{ url_for('auth.change_password') }}" id="change-password-form">
    {{ form.hidden_tag() }} <!-- This includes the CSRF token -->
    <div class="mb-3">
      <label for="current_password" class="form-label">当前密码</label>
      {{ form.old_password(class="form-control", id="current_password") }}
    </div>
    <div class="mb-3">
      <label for="new_password" class="form-label">新密码</label>
      {{ form.new_password(class="form-control", id="new_password") }}
    </div>
    <div class="mb-3">
      <label for="confirm_password" class="form-label">确认新密码</label>
      {{ form.confirm_password(class="form-control", id="confirm_password") }}
      <small id="passwordHelp" class="form-text text-muted">请确保新密码和确认密码一致。</small>
    </div>
    <button type="submit" class="btn btn-primary">更改密码</button>
  </form>
</div>
<script>
  const form = document.getElementById('change-password-form');
  form.addEventListener('submit', function(event) {
    const newPassword = document.getElementById('new_password').value;
    const confirmPassword = document.getElementById('confirm_password').value;
    if (newPassword !== confirmPassword) {
      alert('新密码和确认密码不一致。');
      event.preventDefault();
    }
    else {
      alert('密码更改成功。')
```

```
            form.reset();
            window.location.href = "{{ url_for('auth.login') }}";
            return true;
        }
    });
</script>
{% endblock %}
code_login.html
{% extends "base.html" %}
{% block content %}
<div class="container mt-5">
    <div class="row justify-content-center">
        <div class="col-md-6">
            <h2>验证码登录</h2>
            {% with messages = get_flashed_messages(with_categories=true) %}
                {% if messages %}
                    {% for category, message in messages %}
                        <div class="alert alert-{{ category }} alert-dismissible fade show" role="alert">
                            {{ message }}
                            <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="关闭"></button>
                        </div>
                    {% endfor %}
                {% endif %}
            {% endwith %}
            <form method="POST" id="login-form">
                {{ form.hidden_tag() }}
                <div class="mb-3">
                <label for="email" class="form-label">邮箱地址</label>
                    {{ form.email(class="form-control", placeholder="输入您的邮箱", id="email") }}
                    {% for error in form.email.errors %}
                        <div class="text-danger">{{ error }}</div>
                    {% endfor %}
                </div>
                <div class="mb-3" id="verification-code-field" style="display:none;">
                    <label>验证码</label>
                    {{ form.verification_code(class="form-control", placeholder="输入验证码", id="verification-code") }}
                </div>
                <button type="button" class="btn btn-secondary" id="send-code-btn">发送验证码</button>
                <button type="submit" name="submit" class="btn btn-primary" id="login-btn" style="display: none;">登录</button>
                <p id="countdown-timer" class="mt-2 text-muted" style="display:none;">
                    等待 <span id="timer">60</span> 秒
                </p>
            </form>
            <div class="mt-3">
                <a href="{{ url_for('auth.login') }}" class="btn btn-secondary">返回密码登录</a>
            </div>
        </div>
    </div>
</div>
<script>
document.addEventListener('DOMContentLoaded', function() {
    const sendCodeBtn = document.getElementById('send-code-btn');
const loginBtn = document.getElementById('login-btn');
const emailField = document.getElementById('email');
const verificationCodeField = document.getElementById('verification-code-field');
const countdownTimer = document.getElementById('countdown-timer');
const timerSpan = document.getElementById('timer');
const cooldownTime = 60; // 冷却时间，单位秒
let countdown;
function startCooldown() {
  countdownTimer.style.display = 'block';
  let timeLeft = cooldownTime;
  timerSpan.innerText = timeLeft;
  sendCodeBtn.innerText = `等待 ${timeLeft}s`;
  countdown = setInterval(() => {
    timeLeft--;
    timerSpan.innerText = timeLeft;
    sendCodeBtn.innerText = `等待 ${timeLeft}s`;
    if (timeLeft <= 0) {
      clearInterval(countdown);
      sendCodeBtn.disabled = false;
      sendCodeBtn.innerText = '发送验证码';
      countdownTimer.style.display = 'none';
    }
  }, 1000);
}
sendCodeBtn.addEventListener('click', function(e) {
  e.preventDefault();
  const email = emailField.value.trim();
  if (!email) {
    alert("请输入您的邮箱地址。");
    return;
  }
  this.disabled = true;
  fetch('/send_code', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/x-www-form-urlencoded',
      'X-CSRFToken': '{{ csrf_token() }}'
    },
    body: new URLSearchParams({ 'email': email })
  })
  .then(response => response.json())
  .then(data => {
    if (data.success) {
      alert('验证码已成功发送，请检查您的邮箱。');
      startCooldown();
      verificationCodeField.style.display = 'block';
      loginBtn.style.display = 'block';
    } else {
      alert('发送验证码失败，请稍后重试。');
      sendCodeBtn.disabled = false;
    }
  })
```

```
        .catch(error => {
            console.error('Error:', error);
            alert('发生错误，请稍后再试。');
            sendCodeBtn.disabled = false;
        });
    });
});
</script>
{% endblock %}
delete_account.html
{% extends "base.html" %}
{% block content %}
<div class="container mt-5">
    <h2>注销账户</h2>
    <a href="javascript:history.back()" class="btn btn-secondary">返回上一层</a>
    <p>请注意：账户一旦被删除，所有数据将被永久删除，无法恢复。</p>
    <p>您确定要删除您的账户吗？</p>
    <div id="message" class="mt-3"></div>
    <button type="button" id="send-code-btn" class="btn btn-secondary">发送验证码</button>
    <form method="POST" action="{{ url_for('auth.delete_account') }}" id="delete-account-form" class="mt-3">
        {{ form.hidden_tag() }} <!-- This includes the CSRF token -->
        <div class="form-group">
            <label for="password">密码</label>
            {{ form.password(class="form-control") }}
        </div>
        <div class="form-group">
            <label for="verification_code">验证码</label>
            {{ form.verification_code(class="form-control") }}
        </div>
        <button type="submit" class="btn btn-danger">删除账户</button>
    </form>
</div>
<script>
document.getElementById('send-code-btn').addEventListener('click', function() {
    var formData = new FormData();
    formData.append('send_code', '1');
    fetch('{{ url_for("auth.send_delete_verification_code") }}', {
        method: 'POST',
        body: formData,
        headers: {
            'X-CSRFToken': '{{ form.csrf_token._value() }}'
        }
    })
    .then(response => response.json())
    .then(data => {
        var messageDiv = document.getElementById('message');
        if (data.success) {
            messageDiv.innerHTML = '<div class="alert alert-success">' + data.message + '</div>';
            startCooldown();
        } else {
            messageDiv.innerHTML = '<div class="alert alert-danger">' + data.message + '</div>';
        }
    })
    .catch(error => {
      console.error('Error:', error);
      var messageDiv = document.getElementById('message');
      messageDiv.innerHTML = '<div class="alert alert-danger">发送验证码时出错，请稍后重试。</div>';
    });
    function startCooldown() {
      let timeLeft = 60;
      document.getElementById('countdown-timer').style.display = 'block';
      const timerSpan = document.getElementById('timer');
      timerSpan.innerText = timeLeft;
        let sendCodeBtn = document.getElementById('send-code-btn');
        sendCodeBtn.innerText = `等待 ${timeLeft}s`;
      const countdown = setInterval(() => {
        if (--timeLeft <= 0) {
          clearInterval(countdown);
          let sendCodeBtn = document.getElementById('send-code-btn');
          sendCodeBtn.disabled = false;
          sendCodeBtn.innerText = '发送验证码';
          document.getElementById('countdown-timer').style.display = 'none';
        } else {
          timerSpan.innerText = timeLeft;
          sendCodeBtn.innerText = `等待 ${timeLeft}s`;
        }
      }, 1000);
    }
  });
  document.getElementById('delete-account-form').addEventListener('submit', function(event) {
    event.preventDefault();  // 阻止默认提交行为
    var form = event.target;
    var formData = new FormData(form);
    fetch(form.action, {
      method: 'POST',  // 确保使用 POST 方法
      body: formData,
      headers: {
        'X-CSRFToken': '{{ form.csrf_token._value() }}'
      }
    })
    .then(response => response.json())
    .then(data => {
      var messageDiv = document.getElementById('message');
      if (data.success) {
        alert(data.message);
        window.location.href = '{{ url_for("main.index") }}';
      } else {
        messageDiv.innerHTML = '<div class="alert alert-danger">' + data.message + '</div>';
      }
    })
    .catch(error => {
      console.error('Error:', error);
      var messageDiv = document.getElementById('message');
      messageDiv.innerHTML = '<div class="alert alert-danger">删除账户时出错，请稍后重试。</div>';
    });
});
</script>
{% endblock %}
```

edit_profile.html

```
{% extends "base.html" %}
{% block content %}
<div class="container mt-5">
    <h2>编辑个人资料</h2>
    <form method="POST" id="edit-profile-form">
        {{ form.hidden_tag() }}
        <div class="mb-3">
            {{ form.username.label(class="form-label") }}
            {{ form.username(class="form-control", placeholder="输入用户名") }}
            {% for error in form.username.errors %}
                <div class="text-danger">{{ error }}</div>
            {% endfor %}
        </div>
        <div class="mb-3">
            {{ form.email.label(class="form-label") }}
            {{ form.email(class="form-control", placeholder="输入邮箱") }}
            {% for error in form.email.errors %}
                <div class="text-danger">{{ error }}</div>
            {% endfor %}
        </div>
        <button type="submit" class="btn btn-primary">更新</button>
    </form>
</div>
<script>
document.querySelector('#edit-profile-form').addEventListener('submit', function(event) {
    event.preventDefault();
    const form = event.target;
    const formData = new FormData(form);
    fetch(form.action, {
        method: form.method,
        body: formData,
        headers: {
            'X-CSRFToken': '{{ csrf_token() }}'
        }
    })
    .then(response => {
        const contentType = response.headers.get('content-type');
        if (!contentType || !contentType.includes('application/json')) {
            throw new Error('Received non-JSON response');
        }
        return response.json();
    })
    .then(data => {
        if (data.success) {
            alert('个人资料更新成功');
            form.reset();
        } else {
```

```
        alert(data.message || '个人资料更新失败');
      }
    })
    .catch(error => {
      alert('更新过程中出现错误，请稍后重试。错误代码为: ' + error.message);
    });
});
</script>
{% endblock %}
file_list.html
{% extends "base.html" %}
{% block content %}
<div class="container mt-5">
  <h2>我的文件</h2>
  <a href="javascript:history.back()" class="btn btn-secondary mb-3">返回上一层</a>
  <div class="d-flex justify-content-between mb-3">
    <div>
      <button type="button" id="batch-download-btn" class="btn btn-primary">批量下载</button>
      <button type="button" id="batch-share-btn" class="btn btn-info">批量分享</button>
      <button type="button" id="batch-delete-btn" class="btn btn-danger">批量删除</button>
    </div>
    <div>
      <label for="page-jump" class="form-label">跳转至页数</label>
      <input type="number" id="page-jump" class="form-control d-inline-block" style="width: 70px;" min="1">
      <button type="button" id="jump-page-btn" class="btn btn-secondary">跳转</button>
    </div>
  </div>
  {% with messages = get_flashed_messages(with_categories=true) %}
  {% if messages %}
    <div id="flash-messages">
      {% for category, message in messages %}
        <div class="alert alert-{{ category }} alert-dismissible fade show" role="alert">
          {{ message }}
          <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
        </div>
      {% endfor %}
    </div>
  {% endif %}
  {% endwith %}
  <form id="file-actions-form" method="post">
    <input type="hidden" name="csrf_token" value="{{ csrf_token() }}">
    <table class="table table-striped table-hover">
      <thead>
        <tr>
          <th><input type="checkbox" id="select-all"></th> <!-- 全选框 -->
          <th>文件名</th>
          <th>大小</th>
          <th>上传时间</th>
          <th>下载次数</th>
          <th>操作</th>
        </tr>
      </thead>
```

```
        <tbody id="file-list">
          {% for file in files %}
          <tr>
            <td><input type="checkbox" class="file-checkbox" value="{{ file.id }}"></td>
            <td>{{ file.filename }}</td>
            <td>{{ file.get_formatted_size() }}</td>
            <td>{{ file.created_at }}</td>
            <td>{{ file.download_count }}</td>
            <td>
              <button type="button" class="btn btn-primary download-btn" data-file-id="{{ file.id }}">下载</button>
              <button type="button" class="btn btn-info share-btn" data-file-id="{{ file.id }}">分享</button>
              <button type="button" class="btn btn-danger delete-btn" data-file-id="{{ file.id }}">删除</button>
            </td>
          </tr>
          {% endfor %}
        </tbody>
      </table>
    </form>
    <div class="pagination mt-4 d-flex justify-content-center">
      <a href="{{ pagination.prev_num }}" class="btn btn-secondary" {% if not pagination.has_prev %}disabled{% endif %}>上一
页</a>
      <span>第 {{ pagination.page }} 页，共 {{ pagination.pages }} 页</span>
      <a href="{{ pagination.next_num }}" class="btn btn-secondary" {% if not pagination.has_next %}disabled{% endif %}>下一
页</a>
    </div>
</div>
<script>
document.getElementById('select-all').addEventListener('change', function () {
    const checkboxes = document.querySelectorAll('.file-checkbox');
    checkboxes.forEach(checkbox => checkbox.checked = this.checked);
});
function getSelectedFiles() {
    const selectedFiles = [];
    document.querySelectorAll('.file-checkbox:checked').forEach(checkbox => {
      selectedFiles.push(checkbox.value);
    });
    return selectedFiles;
}
document.getElementById('batch-share-btn').addEventListener('click', function() {
    const selectedFiles = getSelectedFiles();
    if (selectedFiles.length === 0) {
      alert('请选择要分享的文件');
      return;
    }
    selectedFiles.forEach(fileId => {
      fetch(`/share/${fileId}`, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          'X-CSRFToken': '{{ csrf_token() }}' // CSRF 保护
        }
      })
```

```
        .then(response => response.json())
        .then(data => {
            if (data.success) {
                alert('文件分享成功,已复制到剪贴板'+data.share_link) ;
                var oInput = document.createElement('input');
                oInput.value = data.share_link;
                document.body.appendChild(oInput);
                oInput.select(); // 选择对象
                document.execCommand("Copy"); // 执行浏览器复制命令
                oInput.className = 'oInput';
                oInput.style.display = 'none';
            } else {
                alert(data.message || '文件分享失败');
            }
        })
        .catch(error => console.error('分享文件过程中出现错误:', error));
    });
});
document.getElementById('batch-download-btn').addEventListener('click', function() {
    const selectedFiles = getSelectedFiles();
    if (selectedFiles.length === 0) {
        alert('请选择要下载的文件');
        return;
    }
    selectedFiles.forEach(fileId => {
        window.location.href = `/download/${fileId}`;  // 重定向到下载链接
    });
});
document.getElementById('batch-delete-btn').addEventListener('click', function() {
    const selectedFiles = getSelectedFiles();
    if (selectedFiles.length === 0) {
        alert('请选择要删除的文件');
        return;
    }
    if (!confirm('确定要删除选中的文件吗？')) return;
    fetch('/batch_delete', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            'X-CSRFToken': '{{ csrf_token() }}'
        },
        body: JSON.stringify({ file_ids: selectedFiles })
    })
    .then(response => response.json())
    .then(data => {
        if (data.success) {
            alert('文件删除成功');
            location.reload();
        } else {
            alert(data.message || '文件删除失败');
        }
    })
    .catch(error => console.error('删除文件过程中出现错误:', error));
});
document.querySelectorAll('.delete-btn').forEach(button => {
    button.addEventListener('click', function() {
        const fileId = this.getAttribute('data-file-id');
        if (confirm("确定要删除这个文件吗？")) {
            fetch(`/delete/${fileId}`, {
                method: 'POST',
                headers: {
                    'X-CSRFToken': '{{ csrf_token() }}'
                }
            })
            .then(response => response.json())
            .then(data => {
                if (data.success) {
                    alert('文件删除成功');
                    location.reload();
                } else {
                    alert(data.message || '文件删除失败');
                }
            })
            .catch(error => console.error('删除文件过程中出现错误:', error));
        }
    });
});
document.querySelectorAll('.share-btn').forEach(button => {
    button.addEventListener('click', function() {
        const fileId = this.getAttribute('data-file-id');
        fetch(`/share/${fileId}`, {
            method: 'POST',
            headers: {
                'X-CSRFToken': '{{ csrf_token() }}'
            }
        })
        .then(response => response.json())
        .then(data => {
            if (data.success) {
                alert('文件分享成功,已复制到剪贴板'+data.share_link) ;
                var oInput = document.createElement('input');
                oInput.value = data.share_link;
                document.body.appendChild(oInput);
                oInput.select(); // 选择对象
                document.execCommand("Copy"); // 执行浏览器复制命令
                oInput.className = 'oInput';
                oInput.style.display = 'none';
            } else {
                alert(data.message || '文件分享失败');
            }
        })
        .catch(error => console.error('分享文件过程中出现错误:', error));
    });
});
```

```
document.querySelectorAll('.download-btn').forEach(button => {
  button.addEventListener('click', function() {
    const fileId = this.getAttribute('data-file-id');
    window.location.href = `/download/${fileId}`;
  });
});
</script>
{% endblock %}
index.html
{% extends "base.html" %}
{% block content %}
<style>
  .fade-out {
    opacity: 0;
    transition: opacity 1s ease-out;
  }
  .alert {
    opacity: 1;
    transition: opacity 1s ease-in;
  }
</style>
  <div id="flash-messages-container">
  {% with messages = get_flashed_messages(with_categories=true) %}
    {% if messages %}
      <div id="flash-messages">
        {% for category, message in messages %}
          <div class="alert alert-{{ category }} alert-dismissible fade show" role="alert">
            {{ message }}
            <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
          </div>
        {% endfor %}
      </div>
    {% endif %}
  {% endwith %}
  </div>
<div class="container mt-5 text-center">
  <div class="row">
    <div class="col-12">
      <h1 class="display-4">欢迎使用 Lanshi 云盘</h1>
      <p class="lead">您的文件，您的数据，随时随地轻松管理。</p>
      {% if current_user.is_authenticated %}
        <p class="text-muted">欢迎回来, {{ current_user.email }}</p>
      {% else %}
        <p class="text-muted">请登录或注册以开始使用。</p>
      {% endif %}
    </div>
  </div>
  {% if current_user.is_authenticated %}
    <div class="row justify-content-center mt-5">
      <div class="col-md-4 col-lg-3 mb-4 d-flex justify-content-center">
        <div class="card shadow-sm h-100">
          <div class="card-body d-flex flex-column">
            <h5 class="card-title">我的文件</h5>
```

```
                <p class="card-text">查看并管理您上传的文件。</p>
                <a href="{{ url_for('main.file_list') }}" class="btn btn-primary mt-auto">查看</a>
              </div>
            </div>
          </div>
          <div class="col-md-4 col-lg-3 mb-4 d-flex justify-content-center">
            <div class="card shadow-sm h-100">
              <div class="card-body d-flex flex-column">
                <h5 class="card-title">上传文件</h5>
                <p class="card-text">安全上传您的文件并随时访问它们。</p>
                <a href="{{ url_for('main.upload_file') }}" class="btn btn-success mt-auto">上传</a>
              </div>
            </div>
          </div>
          <div class="col-md-4 col-lg-3 mb-4 d-flex justify-content-center">
            <div class="card shadow-sm h-100">
              <div class="card-body d-flex flex-column">
                <h5 class="card-title">转存文件</h5>
                <p class="card-text">转存他人分享的文件到您的云盘。</p>
                <a href="{{ url_for('main.save_shared_file') }}" class="btn btn-warning mt-auto">转存</a>
              </div>
            </div>
          </div>
          <div class="col-md-4 col-lg-3 mb-4 d-flex justify-content-center">
            <div class="card shadow-sm h-100">
              <div class="card-body d-flex flex-column">
                <h5 class="card-title">数据统计</h5>
                <p class="card-text">查看您上传和下载的统计数据。</p>
                <a href="{{ url_for('main.view_stats') }}" class="btn btn-info mt-auto">查看统计</a>
              </div>
            </div>
          </div>
        </div>
      {% else %}
        <div class="row justify-content-center mt-5">
          <div class="col-md-4 col-lg-3 mb-4 d-flex justify-content-center">
            <div class="card shadow-sm h-100">
              <div class="card-body d-flex flex-column">
                <h5 class="card-title">登录</h5>
                <p class="card-text">已有账户？ 立即登录。</p>
                <a href="{{ url_for('auth.login') }}" class="btn btn-primary mt-auto">登录</a>
              </div>
            </div>
          </div>
          <div class="col-md-4 col-lg-3 mb-4 d-flex justify-content-center">
            <div class="card shadow-sm h-100">
              <div class="card-body d-flex flex-column">
                <h5 class="card-title">注册</h5>
                <p class="card-text">还没有账户？ 点击这里注册。</p>
                <a href="{{ url_for('auth.register') }}" class="btn btn-success mt-auto">注册</a>
              </div>
            </div>
            </div>
        </div>
    {% endif %}
</div>
<script>
document.addEventListener('DOMContentLoaded', function() {
    setTimeout(function() {
        var flashMessages = document.getElementById('flash-messages-container');
        if (flashMessages) {
            var alerts = flashMessages.getElementsByClassName('alert');
            for (var i = 0; i < alerts.length; i++) {
                alerts[i].classList.add('.fade-out');
            }
            setTimeout(function() {
                flashMessages.innerHTML = '';
            }, 1000);  // 1 秒的淡化时间
        }
    }, 2000);  // 5 秒后开始淡化
});
</script>
<style>
    .card {
        width: 100%;  /* 卡片占满父元素的宽度 */
        max-width: 280px; /* 设置卡片最大宽度 */
        transition: transform 0.2s ease-in-out;
    }
    .card:hover {
        transform: scale(1.05);  /* 悬停时放大 */
    }
    .btn {
        font-size: 1.1rem;
    }
    .row {
        margin-top: 20px;  /* 调整行间距 */
    }
</style>
{% endblock %}
```

login.html

```
{% extends "base.html" %}
{% block content %}
<div class="container mt-5">
    <div class="row justify-content-center">
        <div class="col-md-6">
            <h2>密码登录</h2>
            {% with messages = get_flashed_messages(with_categories=true) %}
                {% if messages %}
                    {% for category, message in messages %}
                        <div class="alert alert-{{ category }} alert-dismissible fade show" role="alert">
                            {{ message }}
                            <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="关闭"></button>
                        </div>
                    {% endfor %}
            {% endif %}
          {% endwith %}
          <form method="POST" action="{{ url_for('auth.login') }}">
          {{ form.hidden_tag() }}
          <div class="mb-3">
            {{ form.email.label(class="form-label") }}
            {{ form.email(class="form-control", placeholder="输入您的邮箱") }}
            {% for error in form.email.errors %}
              <div class="text-danger">{{ error }}</div>
            {% endfor %}
          </div>
          <div class="mb-3">
            {{ form.password.label(class="form-label") }}
            {{ form.password(class="form-control", placeholder="输入您的密码") }}
            {% for error in form.password.errors %}
              <div class="text-danger">{{ error }}</div>
            {% endfor %}
          </div>
          <button type="submit" class="btn btn-primary">登录</button>
          </form>
          <div class="mt-3">
            <a href="{{ url_for('auth.email_login') }}" class="btn btn-secondary">使用验证码登录</a>
            <a href="{{ url_for('auth.reset_password_request') }}" class="btn btn-link float-end">忘记密码？ </a>
          </div>
        </div>
    </div>
</div>
{% endblock %}
register.html
{% extends "base.html" %}
{% block content %}
<div class="container mt-5">
    <div class="row justify-content-center">
        <div class="col-md-6">
          <h2>注册</h2>
          {% with messages = get_flashed_messages(with_categories=true) %}
            {% if messages %}
              {% for category, message in messages %}
                <div class="alert alert-{{ 'success' if category == 'message' else category }} alert-dismissible fade show"
role="alert">
                  {{ message }}
                  <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="关闭"></button>
                </div>
              {% endfor %}
            {% endif %}
          {% endwith %}
          <form method="POST" id="register-form">
            {{ form.hidden_tag() }}
            <div class="mb-3">
              <label for="email" class="form-label">邮箱地址</label>
              {{ form.email(class="form-control", placeholder="输入您的邮箱", id="email") }}
              {% for error in form.email.errors %}
```

```
                <div class="text-danger">{{ error }}</div>
              {% endfor %}
            </div>
            <div class="mb-3" id="verification-code-field" style="display:none;">
              {{ form.verification_code.label(class="form-label") }}
              {{ form.verification_code(class="form-control", placeholder="输入验证码", id="verification-code") }}
            </div>
            <div class="mb-3" id="password-field" style="display:none;">
              {{ form.password.label(class="form-label") }}
              {{ form.password(class="form-control", placeholder="设置密码", id="password") }}
            </div>
            <button type="button" name="button" value="True" class="btn btn-secondary" id="send-code-btn">发送验证码
</button>
            <button type="submit" name="submit" value="True" class="btn btn-primary" id="register-btn" style="display:
none;">注册</button>
            <p id="countdown-timer" class="mt-2 text-muted" style="display:none;">请等待 <span id="timer">60</span> 秒后
再发送验证码。</p>
          </form>
        </div>
      </div>
    </div>
<script>
document.addEventListener('DOMContentLoaded', function() {
  const sendCodeBtn = document.getElementById('send-code-btn');
  const registerBtn = document.getElementById('register-btn');
  const emailField = document.getElementById('email');
  const verificationCodeField = document.getElementById('verification-code-field');
  const passwordField = document.getElementById('password-field');
  const countdownTimer = document.getElementById('countdown-timer');
  const timerSpan = document.getElementById('timer');
  const cooldownTime = 60; // 冷却时间，单位秒
  let countdown;
  function startCooldown() {
    countdownTimer.style.display = 'block';
    let timeLeft = cooldownTime;
    timerSpan.innerText = timeLeft;
    sendCodeBtn.innerText = `等待 ${timeLeft}s`;
    countdown = setInterval(() => {
      timeLeft--;
      timerSpan.innerText = timeLeft;
      sendCodeBtn.innerText = `等待 ${timeLeft}s`;
      if (timeLeft <= 0) {
        clearInterval(countdown);
        sendCodeBtn.disabled = false;
        sendCodeBtn.innerText = '发送验证码';
        countdownTimer.style.display = 'none';
      }
    }, 1000);
  }
  sendCodeBtn.addEventListener('click', function(e) {
    e.preventDefault();
    const email = emailField.value.trim();
```

```
        if (!email) {
          alert("请输入您的邮箱地址。");
          return;
        }
        this.disabled = true;
        fetch('/send_code', {
          method: 'POST',
          headers: {
            'Content-Type': 'application/x-www-form-urlencoded',
            'X-CSRFToken': '{{ csrf_token() }}'
          },
          body: new URLSearchParams({ 'email': email })
        })
        .then(response => response.json())
        .then(data => {
          if (data.success) {
            alert('验证码已成功发送，请检查您的邮箱。');
            startCooldown();
            verificationCodeField.style.display = 'block';
            passwordField.style.display = 'block';
            registerBtn.style.display = 'block';
          } else {
            alert('发送验证码失败，请稍后重试。');
            sendCodeBtn.disabled = false;
          }
        })
        .catch(error => {
          console.error('Error:', error);
          alert('发生错误，请稍后再试。');
          sendCodeBtn.disabled = false;
        });
    });
});
</script>
{% endblock %}
```

reset_password.html

```
{% extends "base.html" %}
{% block content %}<link rel="stylesheet" href="{{ url_for('static', filename='css/custom.css') }}">
<div class="container mt-5">
    <div class="row justify-content-center">
        <div class="col-md-6">
            <h2>重置密码</h2>
            {% with messages = get_flashed_messages(with_categories=true) %}
                {% if messages %}
                    {% for category, message in messages %}
                        <div class="alert alert-{{ category }} alert-dismissible fade show" role="alert">
                            {{ message }}
                            <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="关闭"></button>
                        </div>
                    {% endfor %}
                {% endif %}
            {% endwith %}
```

```
      <form method="POST" action="{{ url_for('auth.reset_password', token=token) }}">
        {{ form.hidden_tag() }}
        <div class="mb-3">
          {{ form.password.label(class="form-label") }}
          {{ form.password(class="form-control", placeholder="输入新密码") }}
          {% for error in form.password.errors %}
            <div class="text-danger">{{ error }}</div>
          {% endfor %}
        </div>
        <div class="mb-3">
          {{ form.confirm_password.label(class="form-label") }}
          {{ form.confirm_password(class="form-control", placeholder="确认新密码") }}
          {% for error in form.confirm_password.errors %}
            <div class="text-danger">{{ error }}</div>
          {% endfor %}
        </div>
        <button type="submit" class="btn btn-primary">重置密码</button>
      </form>
      <div class="mt-3">
        <a href="{{ url_for('auth.login') }}" class="btn btn-secondary">返回登录</a>
      </div>
    </div>
  </div>
</div>
{% endblock %}
```

reset_password_request.html

```
{% extends "base.html" %}
{% block content %}
<div class="container mt-5">
  <div class="row justify-content-center">
    <div class="col-md-6">
      <h2>重置密码</h2>
      {% with messages = get_flashed_messages(with_categories=true) %}
        {% if messages %}
          {% for category, message in messages %}
            <div class="alert alert-{{ category }} alert-dismissible fade show" role="alert">
              {{ message }}
              <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="关闭"></button>
            </div>
          {% endfor %}
        {% endif %}
      {% endwith %}
      <form method="POST" action="{{ url_for('auth.reset_password_request') }}">
        {{ form.hidden_tag() }}
        <div class="mb-3">
          {{ form.email.label(class="form-label") }}
          {{ form.email(class="form-control", placeholder="输入您的邮箱") }}
          {% for error in form.email.errors %}
            <div class="text-danger">{{ error }}</div>
          {% endfor %}
        </div>
        <button type="submit" class="btn btn-primary">发送重置密码链接</button>
```

```
            </form>
            <div class="mt-3">
                <a href="{{ url_for('auth.login') }}" class="btn btn-secondary">返回登录</a>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

save_shared_file.html

```
{% extends "base.html" %}
{% block content %}
<div class="container mt-5">
    <h2>转存分享的文件</h2>
    <p>请输入分享链接或代码，将文件保存到您的云盘中。</p>
    <a href="javascript:history.back()" class="btn btn-secondary">返回上一层</a>
    <form method="POST" action="{{ url_for('main.save_shared_file') }}">
        {{ form.hidden_tag() }} <!-- This ensures the CSRF token is included -->
       <div class="mb-3">
        {{ form.share_token.label(class="form-label") }}
        <input type="text" name="share_token" class="form-control" placeholder="输入分享令牌" value="{{ share_token }}"
required>
    </div>
        <button type="submit" class="btn btn-primary">保存文件</button>
    </form>
</div>
{% endblock %}
```

share_file.html

```
{% extends "base.html" %}
{% block content %}
    <div class="container mt-5">
        <h2>分享文件: {{ file.filename }}</h2>
        <a href="javascript:history.back()" class="btn btn-secondary">返回上一层</a>
        <p>您可以通过以下链接分享您的文件：</p>
        {% if share_link %}
            <div class="alert alert-info">
                <strong>分享链接：</strong>
                <a href="{{ share_link }}" target="_blank">{{ share_link }}</a>
            </div>
        {% else %}
            <form method="POST" action="{{ url_for('main.share_file', file_id=file.id) }}">
                {{ form.hidden_tag() }}
                <div class="form-group">
                    <label for="filename">文件名</label>
                    <input type="text" class="form-control" value="{{ file.filename }}" readonly>
                </div>
                <button type="submit" class="btn btn-primary">生成分享链接</button>
            </form>
        {% endif %}
    </div>
{% endblock %}
```

stats.html

```
{% extends "base.html" %}
{% block content %}
<div class="container align-center mt-5">
  <div id="main" style="width: 100%; height: 400px;"></div>
</div>
<div class="container align-center mt-5">
  <div id="wordcloud" style="width: 100%; height: 400px;"></div>
</div>
<script src="https://cdn.jsdelivr.net/npm/echarts/dist/echarts.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/echarts-wordcloud/dist/echarts-wordcloud.min.js"></script>
<script>
var myChart = echarts.init(document.getElementById('main'), 'light', {renderer: 'convas'});
var wordcloudChart = echarts.init(document.getElementById('wordcloud'), 'light', {renderer: 'convas'});
fetch("/upload_stats")
  .then(response => response.json())
  .then(data => {
    console.log("词云数据:", data.wordcloud_data);
    var uploadOption = {
      title: {
        text: '每月上传文件统计',
        left: 'center',
        textStyle: {
          fontSize: 18,
          fontWeight: 'bold'
        }
      },
      tooltip: {},
      legend: {
        data: ['上传量'],
        top: 20
      },
      xAxis: {
        type: 'category',
        data: Object.keys(data.monthly_uploads),
        axisLabel: {
          rotate: 45,
          interval: 0
        }
      },
      yAxis: {
        type: 'value'
      },
      series: [{
        name: '上传量',
        type: 'bar',
        data: Object.values(data.monthly_uploads),
        itemStyle: {
          color: '#73C9E6'  // 设置柱子的颜色
        }
      }]
    };
var wordcloudOption = {
  title: {
      text: '文件名词云',
      left: 'center'
    },
    tooltip: {
      show: true
    },
    series: [{
      type: 'wordCloud',
      shape: 'circle',
      gridSize: 10,
      sizeRange: [12, 60],
      rotationRange: [-90, 90],
      textStyle: {
        normal: {
          color: function () {
            return 'rgb(' + [
              Math.round(Math.random() * 255),
              Math.round(Math.random() * 255),
              Math.round(Math.random() * 255)
            ].join(',') + ')';
          },
          textBorderColor: '#000',  // 边框颜色（黑色）
          textBorderWidth: 2,  // 边框宽度
          textShadowBlur: 10,  // 添加阴影模糊效果
          textShadowColor: 'rgba(0, 0, 0, 0.5)'  // 阴影颜色
        },
        emphasis: {
          shadowBlur: 10,
          shadowColor: '#383'
        }
      },
      data: data.wordcloud_data.map(function (item) {
        return {
          name: item.name,
          value: item.value,
        };
      })
    }]
};
      myChart.setOption(uploadOption);
      wordcloudChart.setOption(wordcloudOption);
    })
    .catch(error => console.error('获取数据时出错:', error));
</script>
{% endblock %}
upload.html
{% extends "base.html" %}
{% block content %}
<div class="container mt-5">
    <h2>上传文件</h2>
    <p>安全地上传您的文件并随时随地访问它们。</p>
    <a href="javascript:history.back()" class="btn btn-secondary">返回上一层</a>
    {% with messages = get_flashed_messages(with_categories=true) %}
      {% if messages %}
        {% for category, message in messages %}
          <div class="alert alert-{{ category }} alert-dismissible fade show mt-3" role="alert">
            {{ message }}
            <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
          </div>
        {% endfor %}
      {% endif %}
    {% endwith %}
    <form method="POST" action="{{ url_for('main.upload_file') }}" enctype="multipart/form-data" id="upload-form">
      {{ form.hidden_tag() }} <!-- CSRF token -->
      <div class="mb-3">
        <label for="files" class="form-label">选择文件</label>
        <input type="file" class="form-control" id="files" name="files" multiple required>
        <small class="form-text text-muted">支持多文件上传</small>
        <ul id="file-list" class="list-unstyled mt-2"></ul>
      </div>
      <div class="progress mt-3" style="height: 25px; display: none;" id="progress-bar-container">
        <div class="progress-bar" role="progressbar" style="width: 0%;" id="progress-bar"></div>
      </div>
      <button type="submit" class="btn btn-primary">上传</button>
    </form>
  </div>
  <script>
    document.getElementById('files').addEventListener('change', function() {
      var fileList = document.getElementById('file-list');
      fileList.innerHTML = '';  // 清空之前的列表
      for (var i = 0; i < this.files.length; i++) {
        var li = document.createElement('li');
        li.textContent = this.files[i].name;
        fileList.appendChild(li);
      }
    });
  document.querySelector('#upload-form').addEventListener('submit', function(event) {
    event.preventDefault();
    const form = event.target;
    const formData = new FormData(form);
    fetch(form.action, {
      method: form.method,
      body: formData,
      headers: {
        'X-CSRFToken': '{{ csrf_token() }}'  // Include the CSRF token in the headers
      }
    })
    .then(response => {
      const contentType = response.headers.get('content-type');
      if (!contentType || !contentType.includes('application/json')) {
        throw new Error('Received non-JSON response');
      }
      return response.json();
    })
    .then(data => {
```

```
    console.log("服务器响应数据:", data);
    if (data.success) {
      alert('文件上传成功');
      form.reset();
      data.upload_results.forEach(result => {
        if (result.success) {
          console.log(`文件 ${result.file} 上传成功`);
        } else {
          console.log(`文件 ${result.file} 上传失败: ${result.message}`);
        }
      });
    } else {
      alert(data.message || '文件上传失败');
    }
  })
  .catch(error => {
    alert('上传过程中出现错误，请稍后重试。错误代码为: ' + error.message);
  });
});
</script>
{% endblock %}
```

user_center.html

```
{% extends "base.html" %}
{% block content %}
<div class="container mt-5">
  <h2>用户中心</h2>
  <a href="javascript:history.back()" class="btn btn-secondary">返回上一层</a>
  <div class="row mt-4">
    <div class="col-md-4">
      <div class="card">
        <div class="card-body">
          <h5 class="card-title">更改邮箱</h5>
          <p class="card-text">更改您的登录邮箱。</p>
          <a href="{{ url_for('auth.change_email') }}" class="btn btn-primary">修改</a>
        </div>
      </div>
    </div>
    <div class="col-md-4">
      <div class="card">
        <div class="card-body">
          <h5 class="card-title
          ">修改密码</h5>
          <p class="card-text">更改您的登录密码。</p>
          <a href="{{ url_for('auth.change_password') }}" class="btn btn-primary">修改</a>
        </div>
      </div>
    </div>
    <div class="col-md-4">
      <div class="card">
        <div class="card-body">
          <h5 class="card-title
          ">注销账户</h5>
          <p class="card-text">删除您的账户。</p>
          <a href="{{ url_for('auth.delete_account') }}" class="btn btn-danger">注销</a>
        </div>
      </div>
    </div>
  </div>
</div>
{% endblock %}
test_email.py
from flask import Flask
from flask_mail import Mail, Message
from app.config import Config  # Adjusted import
app = Flask(__name__)
app.config.from_object(Config)
mail = Mail(app)
with app.app_context():
  msg = Message('Test Email', recipients=['1020037769@qq.com'])
  msg.body = 'This is a test email.'
  try:
    mail.send(msg)
    print('Email sent successfully!')
  except Exception as e:
    print(f'Failed to send email: {e}')
utils.py
def format_file_size(size_in_bytes):
  """
  Convert file size in bytes to a human-readable format.
  :param size_in_bytes: File size in bytes
  :return: Formatted string with appropriate size unit
  """
  if size_in_bytes == 0:
    return "0B"
  size_units = ["B", "KB", "MB", "GB", "TB"]
  index = 0
  while size_in_bytes >= 1024 and index < len(size_units) - 1:
    size_in_bytes /= 1024.0
    index += 1
  return f"{size_in_bytes:.2f} {size_units[index]}"
__init__.py

__init__.py
from flask import Flask
from app.config import Config
from .extensions import init_extensions
from .main import main_bp
from .auth import auth_bp
from .models.errors import init_app as init_error_handlers
from flask_jwt_extended import JWTManager
def create_app(config_class=Config):
  app = Flask(__name__, static_folder='static', template_folder='templates')
  JWTManager(app)
  app.config.from_object(config_class)
  init_extensions(app)  # Initialize extensions like db and login_manager
```

```
    init_error_handlers(app)  # Initialize error handlers
    app.register_blueprint(main_bp)
    app.register_blueprint(auth_bp)
    return app
log.py
import logging
from flask import has_request_context, request
from run import app
class RequestFormatter(logging.Formatter):
    def format(self, record):
        if has_request_context():
            record.url = request.url
            record.method = request.method
        else:
            record.url = None
            record.method = None
        return super().format(record)
handler = logging.StreamHandler()
handler.setFormatter(RequestFormatter(
    '[%(asctime)s] %(method)s %(url)s %(levelname)s: %(message)s'
))
app.logger.addHandler(handler)
app.logger.setLevel(logging.DEBUG)
import logging
from flask import has_request_context, request
logging.basicConfig(level=logging.DEBUG)
@app.before_request
def log_request_info():
    if has_request_context():
        app.logger.debug('Headers: %s', request.headers)
        app.logger.debug('Body: %s', request.get_data())
import logging
from logging.handlers import RotatingFileHandler
import os
if not app.debug:
    if not os.path.exists('logs'):
        os.mkdir('logs')
    file_handler = RotatingFileHandler('logs/yourapp.log', maxBytes=10240, backupCount=10)
    file_handler.setFormatter(logging.Formatter(
        '%(asctime)s %(levelname)s: %(message)s [in %(pathname)s:%(lineno)d]'))
    file_handler.setLevel(logging.INFO)
    app.logger.addHandler(file_handler)
    app.logger.setLevel(logging.INFO)
    app.logger.info('YourApp startup')
env.py
from __future__ import with_statement
import logging
from logging.config import fileConfig
from flask import current_app
from alembic import context
config = context.config
fileConfig(config.config_file_name)
logger = logging.getLogger('alembic.env')
```

```
config.set_main_option(
    'sqlalchemy.url',
    str(current_app.extensions['migrate'].db.get_engine().url).replace(
        '%', '%%'))
target_metadata = current_app.extensions['migrate'].db.metadata
def run_migrations_offline():
    """Run migrations in 'offline' mode.
    This configures the context with just a URL
    and not an Engine, though an Engine is acceptable
    here as well.  By skipping the Engine creation
    we don't even need a DBAPI to be available.
    Calls to context.execute() here emit the given string to the
    script output.
    """
    url = config.get_main_option("sqlalchemy.url")
    context.configure(
        url=url, target_metadata=target_metadata, literal_binds=True
    )
    with context.begin_transaction():
        context.run_migrations()
def run_migrations_online():
    """Run migrations in 'online' mode.
    In this scenario we need to create an Engine
    and associate a connection with the context.
    """
    def process_revision_directives(context, revision, directives):
        if getattr(config.cmd_opts, 'autogenerate', False):
            script = directives[0]
            if script.upgrade_ops.is_empty():
                directives[:] = []
                logger.info('No changes in schema detected.')
    connectable = current_app.extensions['migrate'].db.get_engine()
    with connectable.connect() as connection:
        context.configure(
            connection=connection,
            target_metadata=target_metadata,
            process_revision_directives=process_revision_directives,
        )
        with context.begin_transaction():
            context.run_migrations()
if context.is_offline_mode():
    run_migrations_offline()
else:
    run_migrations_online()
```

1a0a6bacc100_initial_migration.py

```
"""Initial migration.
Revision ID: 1a0a6bacc100
Revises: 24f450b0a739
Create Date: 2024-09-01 14:43:41.927358
"""
from alembic import op
import sqlalchemy as sa
from sqlalchemy.dialects import mysql
```

```
revision = '1a0a6bacc100'
down_revision = '24f450b0a739'
branch_labels = None
depends_on = None
def upgrade():
    op.drop_index('name', table_name='role')
    op.drop_table('role')
    op.add_column('user', sa.Column('email', sa.String(length=120), nullable=True))
    op.alter_column('user', 'username',
              existing_type=mysql.VARCHAR(length=80),
              type_=sa.String(length=64),
              nullable=True)
    op.drop_index('phone', table_name='user')
    op.drop_index('username', table_name='user')
    op.create_index(op.f('ix_user_email'), 'user', ['email'], unique=True)
    op.create_index(op.f('ix_user_username'), 'user', ['username'], unique=True)
    op.drop_column('user', 'phone')
def downgrade():
    op.add_column('user', sa.Column('phone', mysql.VARCHAR(length=20), nullable=False))
    op.drop_index(op.f('ix_user_username'), table_name='user')
    op.drop_index(op.f('ix_user_email'), table_name='user')
    op.create_index('username', 'user', ['username'], unique=True)
    op.create_index('phone', 'user', ['phone'], unique=True)
    op.alter_column('user', 'username',
              existing_type=sa.String(length=64),
              type_=mysql.VARCHAR(length=80),
              nullable=False)
    op.drop_column('user', 'email')
    op.create_table('role',
    sa.Column('id', mysql.INTEGER(), autoincrement=True, nullable=False),
    sa.Column('name', mysql.VARCHAR(length=50), nullable=True),
    sa.PrimaryKeyConstraint('id'),
    mysql_default_charset='utf8',
    mysql_engine='InnoDB'
    )
    op.create_index('name', 'role', ['name'], unique=True)
24f450b0a739_initial_migration.py
"""Initial migration.
Revision ID: 24f450b0a739
Revises:
Create Date: 2024-08-31 20:53:42.027075
"""
from alembic import op
import sqlalchemy as sa
revision = '24f450b0a739'
down_revision = None
branch_labels = None
depends_on = None
def upgrade():
    op.create_table('file',
    sa.Column('id', sa.Integer(), nullable=False),
    sa.Column('filename', sa.String(length=255), nullable=False),
    sa.Column('path', sa.String(length=255), nullable=False),
    sa.Column('token', sa.String(length=255), nullable=True),
    sa.PrimaryKeyConstraint('id'),
    sa.UniqueConstraint('token')
    )
    op.create_table('role',
    sa.Column('id', sa.Integer(), nullable=False),
    sa.Column('name', sa.String(length=50), nullable=True),
    sa.PrimaryKeyConstraint('id'),
    sa.UniqueConstraint('name')
    )
    op.create_table('user',
    sa.Column('id', sa.Integer(), nullable=False),
    sa.Column('username', sa.String(length=80), nullable=False),
    sa.Column('phone', sa.String(length=20), nullable=False),
    sa.Column('password_hash', sa.String(length=128), nullable=True),
    sa.PrimaryKeyConstraint('id'),
    sa.UniqueConstraint('phone'),
    sa.UniqueConstraint('username')
    )
def downgrade():
    op.drop_table('user')
    op.drop_table('role')
    op.drop_table('file')
```

3596f074b2e4_add_user_id_to_file_model.py

```
"""Add user_id to File model
Revision ID: 3596f074b2e4
Revises: d1ea81cc37c9
Create Date: 2024-09-01 17:26:24.692856
"""
from alembic import op
import sqlalchemy as sa
from sqlalchemy.dialects import mysql
revision = '3596f074b2e4'
down_revision = 'd1ea81cc37c9'
branch_labels = None
depends_on = None
def upgrade():
    op.add_column('user', sa.Column('user_id', sa.Integer(), nullable=False))
    op.add_column('user', sa.Column('created_at', sa.DateTime(), nullable=True))
    op.alter_column('user', 'username',
               existing_type=mysql.VARCHAR(length=64),
               type_=sa.String(length=150),
               existing_nullable=True)
    op.alter_column('user', 'email',
               existing_type=mysql.VARCHAR(length=120),
               type_=sa.String(length=150),
               nullable=False)
    op.alter_column('user', 'password_hash',
               existing_type=mysql.VARCHAR(length=128),
               nullable=False)
    op.drop_index('ix_user_username', table_name='user')
    op.create_foreign_key(None, 'user', 'user', ['user_id'], ['id'])
def downgrade():
    op.drop_constraint(None, 'user', type_='foreignkey')
    op.create_index('ix_user_username', 'user', ['username'], unique=True)
    op.alter_column('user', 'password_hash',
               existing_type=mysql.VARCHAR(length=128),
               nullable=True)
    op.alter_column('user', 'email',
               existing_type=sa.String(length=150),
               type_=mysql.VARCHAR(length=120),
               nullable=True)
    op.alter_column('user', 'username',
               existing_type=sa.String(length=150),
               type_=mysql.VARCHAR(length=64),
               existing_nullable=True)
    op.drop_column('user', 'created_at')
    op.drop_column('user', 'user_id')
44cf2b238717_fix_relationships_between_user_and_file_.py
"""Fix relationships between User and File models
Revision ID: 44cf2b238717
Revises: d5e4f818151e
Create Date: 2024-09-01 19:04:51.455938
"""
from alembic import op
import sqlalchemy as sa
from sqlalchemy.dialects import mysql
revision = '44cf2b238717'
down_revision = 'd5e4f818151e'
branch_labels = None
depends_on = None
def upgrade():
    op.alter_column('file', 'size',
               existing_type=mysql.FLOAT(),
               type_=sa.Integer(),
               nullable=False)
    op.drop_column('file', 'path')
    op.alter_column('user', 'email',
               existing_type=mysql.VARCHAR(length=150),
               type_=sa.String(length=120),
               existing_nullable=False)
    op.drop_column('user', 'path')
def downgrade():
    op.add_column('user', sa.Column('path', mysql.VARCHAR(length=256), server_default=sa.text("''"), nullable=True))
    op.alter_column('user', 'email',
               existing_type=sa.String(length=120),
               type_=mysql.VARCHAR(length=150),
               existing_nullable=False)
    op.add_column('file', sa.Column('path', mysql.VARCHAR(charset='utf8', collation='utf8_general_ci', length=256),
nullable=True))
    op.alter_column('file', 'size',
               existing_type=sa.Integer(),
               type_=mysql.FLOAT(),
               nullable=True)
72c1197ee31c_add_verification_code_column_to_user_.py
"""Add verification_code column to User model
```

```
Revision ID: 72c1197ee31c
Revises: cb5a8e38345a
Create Date: 2024-09-02 23:25:56.866638
"""
from alembic import op
import sqlalchemy as sa
from sqlalchemy.dialects import mysql
revision = '72c1197ee31c'
down_revision = 'cb5a8e38345a'
branch_labels = None
depends_on = None
def upgrade():
    op.alter_column('file', 'share_token',
               existing_type=mysql.VARCHAR(length=64),
               type_=sa.String(length=32),
               existing_nullable=True,
               existing_server_default=sa.text("''"))
    op.create_unique_constraint(None, 'file', ['share_token'])
    op.add_column('user', sa.Column('verification_code', sa.String(length=6), nullable=True))
def downgrade():
    op.drop_column('user', 'verification_code')
    op.drop_constraint(None, 'file', type_='unique')
    op.alter_column('file', 'share_token',
               existing_type=sa.String(length=32),
               type_=mysql.VARCHAR(length=64),
               existing_nullable=True,
               existing_server_default=sa.text("''"))
828d662c9ec6_add_size_column_to_file_model.py
"""Add size column to File model
Revision ID: 828d662c9ec6
Revises: bc879d9b9e63
Create Date: 2024-09-01 18:06:12.433273
"""
from alembic import op
import sqlalchemy as sa
revision = '828d662c9ec6'
down_revision = 'bc879d9b9e63'
branch_labels = None
depends_on = None
def upgrade():
    op.create_foreign_key(None, 'user', 'user', ['user_id'], ['id'])
def downgrade():
    op.drop_constraint(None, 'user', type_='foreignkey')
88789f2b01a9_add_size_column_to_file_model.py
"""Add size column to File model
Revision ID: 88789f2b01a9
Revises: 828d662c9ec6
Create Date: 2024-09-01 18:12:43.707997
"""
from alembic import op
import sqlalchemy as sa
from sqlalchemy.dialects import mysql
revision = '88789f2b01a9'
```

```
down_revision = '828d662c9ec6'
branch_labels = None
depends_on = None
def upgrade():
   op.alter_column('file', 'name',
            existing_type=mysql.VARCHAR(length=255),
            type_=sa.String(length=256),
            existing_nullable=False)
   op.create_foreign_key(None, 'user', 'user', ['user_id'], ['id'])
def downgrade():
   op.drop_constraint(None, 'user', type_='foreignkey')
   op.alter_column('file', 'name',
            existing_type=sa.String(length=256),
            type_=mysql.VARCHAR(length=255),
            existing_nullable=False)
b9afdffd95b3_add_verification_sent_at_column_to_user_.py
"""Add verification_sent_at column to user table
Revision ID: b9afdffd95b3
Revises: 72c1197ee31c
Create Date: 2024-09-07 23:02:05.441751
"""
from alembic import op
import sqlalchemy as sa
revision = 'b9afdffd95b3'
down_revision = '72c1197ee31c'
branch_labels = None
depends_on = None
def upgrade():
   op.add_column('user', sa.Column('verification_sent_at', sa.DateTime(), nullable=True))
def downgrade():
   op.drop_column('user', 'verification_sent_at')
bc879d9b9e63_add_size_column_to_file_model.py
"""Add size column to File model
Revision ID: bc879d9b9e63
Revises: f96ccb3dad13
Create Date: 2024-09-01 18:05:25.624705
"""
from alembic import op
import sqlalchemy as sa
revision = 'bc879d9b9e63'
down_revision = 'f96ccb3dad13'
branch_labels = None
depends_on = None
def upgrade():
   op.create_foreign_key(None, 'user', 'user', ['user_id'], ['id'])
def downgrade():
   op.drop_constraint(None, 'user', type_='foreignkey')
cb5a8e38345a_add_share_token_to_file_model.py
"""Add share_token to File model
Revision ID: cb5a8e38345a
Revises: 44cf2b238717
Create Date: 2024-09-01 22:12:18.311618
"""
```

```
from alembic import op
import sqlalchemy as sa
from sqlalchemy.dialects import mysql
revision = 'cb5a8e38345a'
down_revision = '44cf2b238717'
branch_labels = None
depends_on = None
def upgrade():
    op.alter_column('file', 'size',
               existing_type=mysql.FLOAT(),
               type_=sa.Integer(),
               comment=None,
               existing_comment='mb',
               existing_nullable=False)
    op.drop_column('file', 'path')
def downgrade():
    op.add_column('file', sa.Column('path', mysql.VARCHAR(charset='utf8', collation='utf8_general_ci', length=256),
server_default=sa.text("''"), nullable=True))
    op.alter_column('file', 'size',
               existing_type=sa.Integer(),
               type_=mysql.FLOAT(),
               comment='mb',
               existing_nullable=False)
d1ea81cc37c9_add_is_active_field_to_user_model.py
"""Add is_active field to User model
Revision ID: d1ea81cc37c9
Revises: 1a0a6bacc100
Create Date: 2024-09-01 16:23:13.593843
"""
from alembic import op
import sqlalchemy as sa
revision = 'd1ea81cc37c9'
down_revision = '1a0a6bacc100'
branch_labels = None
depends_on = None
def upgrade():
    op.add_column('user', sa.Column('is_active', sa.Boolean(), nullable=True))
def downgrade():
    op.drop_column('user', 'is_active')
d5e4f818151e_remove_username_and_use_email_as_unique_.py
"""Remove username and use email as unique identifier
Revision ID: d5e4f818151e
Revises: 88789f2b01a9
Create Date: 2024-09-01 18:36:34.565552
"""
from alembic import op
import sqlalchemy as sa
from sqlalchemy.dialects import mysql
revision = 'd5e4f818151e'
down_revision = '88789f2b01a9'
branch_labels = None
depends_on = None
def upgrade():
    op.drop_constraint('user_ibfk_1', 'user', type_='foreignkey')
    op.drop_column('user', 'username')
    op.drop_column('user', 'user_id')
def downgrade():
    op.add_column('user', sa.Column('user_id', mysql.INTEGER(), autoincrement=False, nullable=False))
    op.add_column('user', sa.Column('username', mysql.VARCHAR(length=150), nullable=True))
    op.create_foreign_key('user_ibfk_1', 'user', 'user', ['user_id'], ['id'])
```

f96ccb3dad13_add_user_id_to_file_model.py

```
"""Add user_id to File model
Revision ID: f96ccb3dad13
Revises: 3596f074b2e4
Create Date: 2024-09-01 17:45:04.157342
"""
from alembic import op
import sqlalchemy as sa
from sqlalchemy.dialects import mysql
revision = 'f96ccb3dad13'
down_revision = '3596f074b2e4'
branch_labels = None
depends_on = None
def upgrade():
    op.add_column('file', sa.Column('name', sa.String(length=255), nullable=False))
    op.add_column('file', sa.Column('user_id', sa.Integer(), nullable=False))
    op.add_column('file', sa.Column('created_at', sa.DateTime(), nullable=True))
    op.drop_index('token', table_name='file')
    op.create_foreign_key(None, 'file', 'user', ['user_id'], ['id'])
    op.drop_column('file', 'token')
    op.drop_column('file', 'path')
    op.drop_column('file', 'filename')
    op.create_foreign_key(None, 'user', 'user', ['user_id'], ['id'])
def downgrade():
    op.drop_constraint(None, 'user', type_='foreignkey')
    op.add_column('file', sa.Column('filename', mysql.VARCHAR(length=255), nullable=False))
    op.add_column('file', sa.Column('path', mysql.VARCHAR(length=255), nullable=False))
    op.add_column('file', sa.Column('token', mysql.VARCHAR(length=255), nullable=True))
    op.drop_constraint(None, 'file', type_='foreignkey')
    op.create_index('token', 'file', ['token'], unique=True)
    op.drop_column('file', 'created_at')
    op.drop_column('file', 'user_id')
    op.drop_column('file', 'name')
```

package.README.md

See more details about in the "exports" field of `package.json` and why it is written like that in https://github.com/apache/echarts/pull/19513 .

Only these entries are officially exported to users:

+ `'echarts'`
+ `'echarts/index.js'`
+ `'echarts/index.blank.js'`
+ `'echarts/index.common.js'`
+ `'echarts/index.simple.js'`
+ `'echarts/core.js'`
+ `'echarts/charts.js'`
+ `'echarts/components.js'`
+ `'echarts/features.js'`

+ `'echarts/renderers.js'`
+ `'echarts/i18n/*`
+ `'echarts/theme/*`
+ `'echarts/types/*`
+ `'echarts/extension/*`
+ `'echarts/dist/*`
+ `'echarts/ssr/client/index.js'`
The other entries listed in the `"exports"` field of `package.json` make the internal files visible, but they are legacy usages, not recommended but not forbidden (for the sake of keeping backward compatible). These entries are made from the search in github about which internal files are imported.
Since `v5.5.0`, `"type": "module"` and `"exports: {...}"` are added to `package.json`. When upgrading to `v5.5.0+`, if you meet some problems about "can not find/resolve xxx" when importing `echarts/i18n/xxx` or `echarts/theme/xxx` or some internal files, it probably because of the issue "file extension not fully specified". Please try to make the file extension fully specified (that is, `import 'xxx/xxx/xxx.js'` rather than `import 'xxx/xxx/xxx'`), or change the config of you bundler tools to support auto adding file extensions.
About `"./types/dist/shared": "./types/dist/shared.d.ts",` in "exports", see https://github.com/apache/echarts/pull/19663 .
Although using `"exports"` of `package.json` we can make alias (or say, route) to physical file (for example: `{ "exports": { "./xxx": "./yyy/zzz.js" } }` enables `import 'echarts/xxx'` to route to `'echarts/yyy/zzz.js'`), at present we can not make sure all the versions of bundle tools and runtimes in use do it consistently. So we do not use the alias setting, but keep providing physical file for each public entry. For example, for an official public entry `'echarts/core'`, we provide a file `echarts/core.js` (and `echarts/core.d.ts`).
README.md
<a href="https://echarts.apache.org/">
  <img style="vertical-align: top;" src="./asset/logo.png?raw=true" alt="logo" height="50px">
</a>
Apache ECharts is a free, powerful charting and visualization library offering easy ways to add intuitive, interactive, and highly customizable charts to your commercial products. It is written in pure JavaScript and based on <a href="https://github.com/ecomfe/zrender">zrender</a>, which is a whole new lightweight canvas library.
[![License](https://img.shields.io/npm/l/echarts?color=5470c6)](https://github.com/apache/echarts/blob/master/LICENSE)
[![Latest npm release](https://img.shields.io/npm/v/echarts?color=91cc75)](https://www.npmjs.com/package/echarts)
[![NPM downloads](https://img.shields.io/npm/dm/echarts.svg?label=npm%20downloads&style=flat&color=fac858)](https://www.npmjs.com/package/echarts)
[![Contributors](https://img.shields.io/github/contributors/apache/echarts?color=3ba272)](https://github.com/apache/echarts/graphs/contributors)
[![Build Status](https://github.com/apache/echarts/actions/workflows/ci.yml/badge.svg)](https://github.com/apache/echarts/actions/workflows/ci.yml)
You may choose one of the following methods:
+ Download from the [official website](https://echarts.apache.org/download.html)
+ `npm install echarts --save`
+ CDN: [jsDelivr CDN](https://www.jsdelivr.com/package/npm/echarts?path=dist)
+ [Get Started](https://echarts.apache.org/handbook)
+ [API](https://echarts.apache.org/api.html)
+ [Option Manual](https://echarts.apache.org/option.html)
+ [Examples](https://echarts.apache.org/examples)
+ [GitHub Issues](https://github.com/apache/echarts/issues) for bug report and feature requests
+ Email dev@echarts.apache.org for general questions
+ Subscribe to the [mailing list](https://echarts.apache.org/maillist.html) to get updated with the project
Build echarts source code:
Execute the instructions in the root directory of the echarts:
([Node.js](https://nodejs.org) is required)
```shell
```

npm install
npm run dev
npm run checktype
npm run release
~~~
Then the "production" files are generated in the `dist` directory.
Please refer to the [contributing](https://github.com/apache/echarts/blob/master/CONTRIBUTING.md) document if you wish to debug locally or make pull requests.
[https://github.com/ecomfe/awesome-echarts](https://github.com/ecomfe/awesome-echarts)
+ [ECharts GL](https://github.com/ecomfe/echarts-gl) An extension pack of ECharts, which provides 3D plots, globe visualization, and WebGL acceleration.
+ [Liquidfill 水球图](https://github.com/ecomfe/echarts-liquidfill)
+ [Wordcloud 字符云](https://github.com/ecomfe/echarts-wordcloud)
+ [Extension for Baidu Map 百度地图扩展](https://github.com/apache/echarts/tree/master/extension-src/bmap) An extension provides a wrapper of Baidu Map Service SDK.
+ [vue-echarts](https://github.com/ecomfe/vue-echarts) ECharts component for Vue.js
+ [echarts-stat](https://github.com/ecomfe/echarts-stat) Statistics tool for ECharts
ECharts is available under the Apache License V2.
Please refer to [Apache Code of Conduct](https://www.apache.org/foundation/policies/conduct.html).
Deqing Li, Honghui Mei, Yi Shen, Shuang Su, Wenli Zhang, Junting Wang, Ming Zu, Wei Chen.
[ECharts: A Declarative Framework for Rapid Construction of Web-based Visualization](https://www.sciencedirect.com/science/article/pii/S2468502X18300068).
Visual Informatics, 2018.
CopyrightNotice.txt

LICENSE.txt

README.md
This is a runtime library for [TypeScript](http://www.typescriptlang.org/) that contains all of the TypeScript helper functions.
This library is primarily used by the `--importHelpers` flag in TypeScript.
When using `--importHelpers`, a module that uses helper functions like `__extends` and `__assign` in the following emitted file:
```ts
var __assign = (this && this.__assign) || Object.assign || function(t) {
    for (var s, i = 1, n = arguments.length; i < n; i++) {
        s = arguments[i];
~~~

    for (var p in s) if (Object.prototype.hasOwnProperty.call(s, p))
      t[p] = s[p];
  }
  return t;
};
exports.x = {};
exports.y = __assign({}, exports.x);
~~~
will instead be emitted as something like the following:
```ts
var tslib_1 = require("tslib");
exports.x = {};
exports.y = tslib_1.__assign({}, exports.x);
~~~
Because this can avoid duplicate declarations of things like `__extends`, `__assign`, etc., this means delivering users smaller files on average, as well as less runtime overhead.
For optimized bundles with TypeScript, you should absolutely consider using `tslib` and `--importHelpers`.
For the latest stable version, run:
```sh
npm install tslib
npm install tslib@^1
npm install tslib@1.6.1
```
```sh
yarn add tslib
yarn add tslib@^1
yarn add tslib@1.6.1
```
```sh
bower install tslib
bower install tslib@^1
bower install tslib@1.6.1
```
```sh
jspm install tslib
jspm install tslib@^1
jspm install tslib@1.6.1
```
Set the `importHelpers` compiler option on the command line:
```
tsc --importHelpers file.ts
```
or in your tsconfig.json:
```json
{
  "compilerOptions": {
    "importHelpers": true
  }
}
```
You will need to add a `paths` mapping for `tslib`, e.g. For Bower users:
```json
{
```

```
  "compilerOptions": {
    "module": "amd",
    "importHelpers": true,
    "baseUrl": "./",
    "paths": {
      "tslib" : ["bower_components/tslib/tslib.d.ts"]
    }
  }
}
```
For JSPM users:
```json
{
  "compilerOptions": {
    "module": "system",
    "importHelpers": true,
    "baseUrl": "./",
    "paths": {
      "tslib" : ["jspm_packages/npm/tslib@2.x.y/tslib.d.ts"]
    }
  }
}
```
- Choose your new version number
- Set it in `package.json` and `bower.json`
- Create a tag: `git tag [version]`
- Push the tag: `git push --tags`
- Create a [release in GitHub](https://github.com/microsoft/tslib/releases)
- Run the [publish to npm](https://github.com/microsoft/tslib/actions?query=workflow%3A%22Publish+to+NPM%22) workflow

Done.
There are many ways to [contribute](https://github.com/Microsoft/TypeScript/blob/master/CONTRIBUTING.md) to TypeScript.
tslib.es6.html
<script src="tslib.es6.js"></script>
tslib.html
<script src="tslib.js"></script>
package.README.md
Only `zrender.js` are officially exported to users.
The other entries listed in the `"exports"` field of `package.json` make the internal files visible, but they are legacy usages, not recommended but not forbidden (for the sake of keeping backward compatible). These entries are made from the search in github about which internal files are imported.
Since `v5.5.0`, `"type": "module"` and `"exports: {...}"` are added to `package.json`. When upgrading to `v5.5.0+`, if you meet some problems about "can not find/resolve xxx" when importing some internal files, it probably because of the issue "file extension not fully specified". Please try to make the file extension fully specified (that is, `import 'xxx/xxx/xxx.js'` rather than `import 'xxx/xxx/xxx'`), or change the config of you bundler tools to support auto adding file extensions.
See more details about the `"exports"` field of `package.json` and why it is written like that in https://github.com/apache/echarts/pull/19513 .
README.md

ZRender
=======
A lightweight graphic library which provides 2d draw for [Apache ECharts](https://github.com/apache/echarts).
[![](https://img.shields.io/github/actions/workflow/status/ecomfe/zrender/ci.yml)]()

[![](https://img.shields.io/npm/dw/zrender.svg?label=npm%20downloads&style=flat)](https://www.npmjs.com/package/zrender) ![Commits Since 4.0.0](https://img.shields.io/github/commits-since/ecomfe/zrender/4.0.0.svg?colorB=%234c1&style=flat)
https://ecomfe.github.io/zrender-doc/public/

README.md

requirements.txt

```
Flask==2.1.2
WTForms==3.0.1
Werkzeug==2.1.1
click~=8.1.7
pytest~=8.3.2
```

run.py

```python
import os
from app import create_app
app = create_app()
upload_folder = app.config['UPLOAD_FOLDER']
if not os.path.exists(upload_folder):
    os.makedirs(upload_folder)
print(app.config['SQLALCHEMY_DATABASE_URI'])
if __name__ == '__main__':
    app.run(debug=True)
```

conftest.py

```python
import pytest
from app import create_app, db
@pytest.fixture
def app():
    app = create_app()
    with app.app_context():
        db.create_all()
        yield app
        db.session.remove()
        db.drop_all()
@pytest.fixture
def client(app):
    return app.test_client()
test_integration.py
def test_user_registration_and_login(client):
    client.post('/register', data={
        'username': 'newuser',
        'email': 'new@example.com',
        'password': 'testpass',
        'confirm_password': 'testpass'
    })
    response = client.post('/login', data={
        'username': 'newuser',
        'password': 'testpass'
    }, follow_redirects=True)
    assert 'Welcome to the Dashboard' in response.get_data(as_text=True)
test_models.py
from app.models.models import User
def test_new_user():
    user = User(username='testuser', email='test@example.com')
    user.set_password('123456')
    assert user.username == 'testuser'
    assert user.check_password('123456') == True
    assert user.check_password('wrongpassword') == False
test_views.py
def test_login_page(client):
    response = client.get('/login')
    assert response.status_code == 200
    assert 'Login' in response.get_data(as_text=True)
def test_register(client):
    response = client.post('/register', data={
        'username': 'testuser',
        'email': 'test@example.com',
        'password': '123456',
        'confirm_password': '123456'
    }, follow_redirects=True)
    assert 'Congratulations, registration successful!' in response.get_data(as_text=True)
```