Министерство науки и высшего образования РФ

федеральное государственное автономное образовательное учреждение высшего образования

«Омский государственный технический университет»

Лицей Академии Яндекса

Comic-cat

Выполнил:

Мамонова Василиса Максимовна

Омск, 2022

РЕФЕРАТ

Отчет 11 с., 5 рис., 3 источн., 1 прил.

ПРОЕКТ, HTML, ДАННЫЕ, PYTHON, FLASK, САЙТ, КОМИКСЫ, ФОРУМ

Предмет исследования – веб-сайт.

Цель исследования – разработка веб-сайта для просмотра, выкладывания и обсуждения комиксов

В процессе выполнения проекта я изучила библиотеки Flask, SqlAlchemy, а также orm-модели, bootstrap и jquery.

В результате выполнения проекта я получила веб-сайт, который соответствует моей цели.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ](#_2r9v0h46yy99) 3

[1 Описание проблемы и формулирование гипотезы для ее решения](#_ifzanbkzs7df) 5

[2 Проектирование приложения](#_pd7u93hyun9o) 5

[3 Реализация приложения](#_67z7petns5sz) 5

[ЗАКЛЮЧЕНИЕ](#_2et92p0) 6

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ](#_tyjcwt) 7

[ПРИЛОЖЕНИЕ А](#_1t3h5sf) Листинг 8

# ВВЕДЕНИЕ

По данным Википедии, сайт, или веб-сайт, также веб-узел, — одна или несколько логически связанных между собой веб-страниц; также место расположения контента сервера. Виды сайтов: сайт-визитка, интернет-магазин, корпоративный сайт, сайт определённой тематики. В настоящее время мы привыкли получать информацию из тематических сайтов. Среди современной молодёжи много поклонников комиксов. Но покупать бумажные издания не всегда удобно и довольно дорого. Поэтому у меня возникла идея создания платформы, на которой пользователи могут размещать и просматривать комиксы.

Цель исследования – разработка веб-сайта для просмотра, выкладывания и обсуждения комиксов

Для достижения поставленной цели требуется решить следующие задачи:

1. исследовать проблему предметной области ;
2. спроектировать структуру сайта и систему хранения данных ;
3. реализовать спроектированное приложение.

## 1 Описание проблемы и формулирование гипотезы для ее решения

История комиксов в современном понимании этого слова берёт начало в середине XVIII века. Появление комиксов тесно связано с политической и социальной жизнью общества. Одним из первых создателей комиксов считается английский художник и иллюстратор Уильям Хогарт (1697 — 1764). В 1843 году рисунки, появившиеся в газетах и журналах, стали называться мультфильмами. Позже из-за того, что все рисованные истории носили комический характер, их переименуют в знакомый нам «комикс». Началом золотого века принято считать первое появление Супермена в комиксе Action Comics #1, вышедшего в 1938 году и изданного DC Comics. Одна из разновидностей комиков - это манга, графические романы, созданные в Японии. Японские комиксы в том виде, в котором мы привыкли их видеть сегодня, были нарисованы Осаму Тэдзукой в 1947 году.

Комиксы издаются не таким большим тиражом, поэтому заполучить бумажный вариант достаточно сложно, и их стоимость может быть весьма высокой. Куда удобнее читать комиксы онлайн. Изучив аналогичные ресурсы, я пришла к выводу, что стоит создать свой сайт, доработав его для удобства. Например, раздел форум позволит пользователям обсудить понравившиеся издания.

## 2 Проектирование сайта

Я начала работу над сайтом с создания его структуры:

- Формы для регистрации и авторизации

- Сервис для просмотра комиксов

o Главная страница + поиск (по жанрам и т.д.)

o Страница с информацией о комиксе (автор, рецензии)

o Страницы комикса

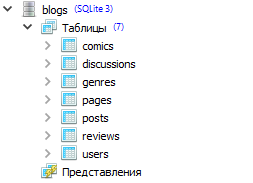
o Формы для выкладывания собственного комикса (сам комикс, страницы комикса)

- Форум для обсуждения комиксов

o Главная страница + поиск пользователей и обсуждений

o Обсуждения

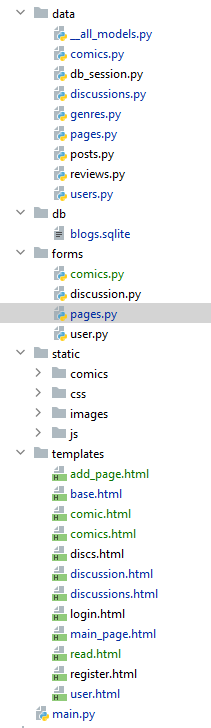
o Страница пользователя (картинка пользователя, описание, список избранных комиксов, роль)

Следующим этапом работы было создание базы данных. В ней 7 основных таблиц: 

* comics(комиксы): название, описание, обложка, дата создания, автор, жанр
* discussions(обсуждения): название, описание, дата создания, автор, посты
* genres(жанры): название
* pages(страницы комикса): номер, сама страница(изображение), комикс
* posts(посты): содержание, дата создания, автор, обсуждение
* reviews(рецензии): содержание, дата создания, автор, комикс
* users(пользователи): имя, описание, электронная почта, пароль, любимые комиксы, созданные комиксы.

## 

## 3 Реализация сайта

Вначале я создала 7 классов, соответствующих таблицам в базе данных с использованием библиотеки SQLAlchemy. Затем я начала работу над файлом main.py, где находятся все функции, отвечающие за страницы. Все они созданы с помощью библиотеки flask. 

Функции register и login: отвечают за регистрацию и авторизацию соответственно. Используют одноимённые шаблоны и формы.

Функция index: отвечает за главную страницу с галереей комиксов. Есть кнопки для перехода на страницу форума и для выкладывания своего комикса(функция comics, форма ComicsForm и шаблон comics.html). Загружает шаблон main\_page.html.

Функция discussions: отвечает за форум, страницу с списком обсуждений. Есть кнопки для перехода на главную страницу и для создания своего обсуждения(функция add\_disc, форма DiscussionForm и шаблон discs.html).

Функция find: отвечает за страницу с результатами поиска комиксов засчёт метода filter у объекта query. Использует шаблон main\_page с изменённым набором комиксов в зависимости от ввода. Строка поиска находится в заголовке базового шаблона base.html, как и кнопки Войти и Зарегистрироваться.

Функция discussion: отображает содержание обсуждения и осуществляет добавление в неё постов. Использует одноимённый шаблон.

Функция user: отображает информацию о каком-либо пользователе. Использует одноимённый шаблон.

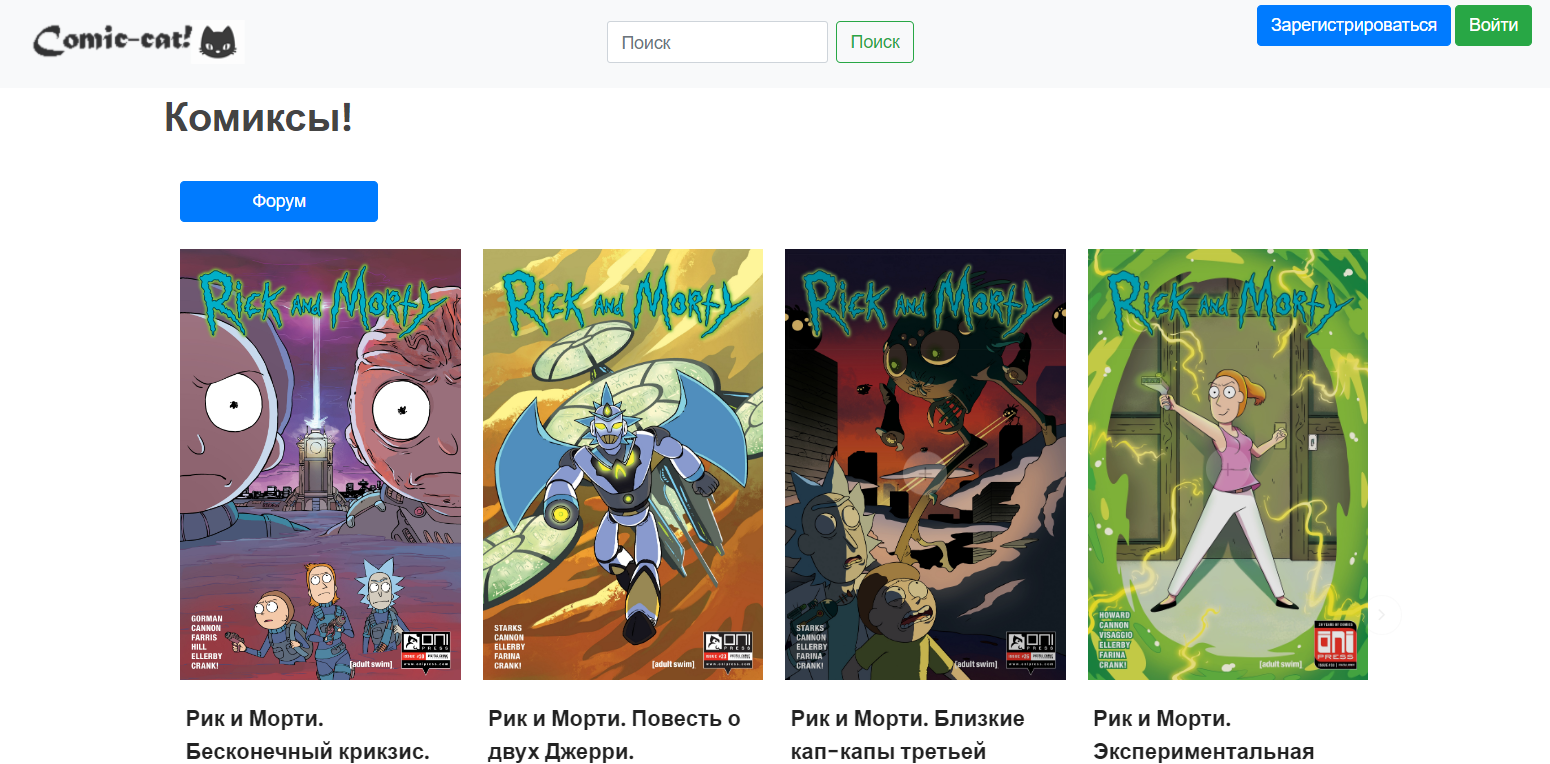
Функция comic: отображает информацию о каком-либо комиксе. Есть кнопки для начала чтения, возвращения на главную страницу, добавления страниц и удаления самого комикса. Использует одноимённый шаблон.

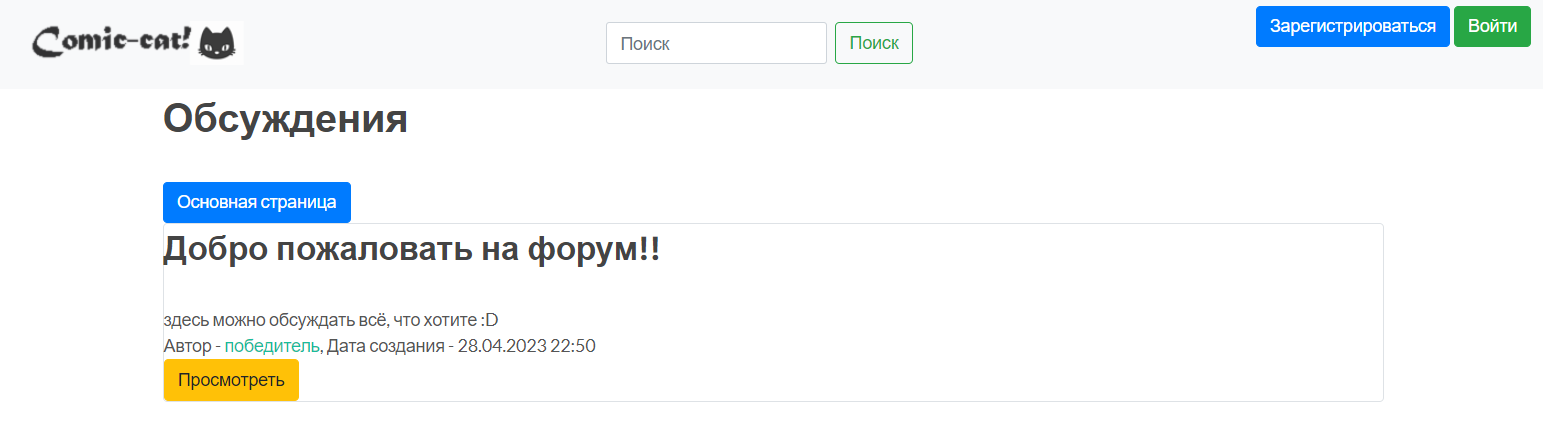
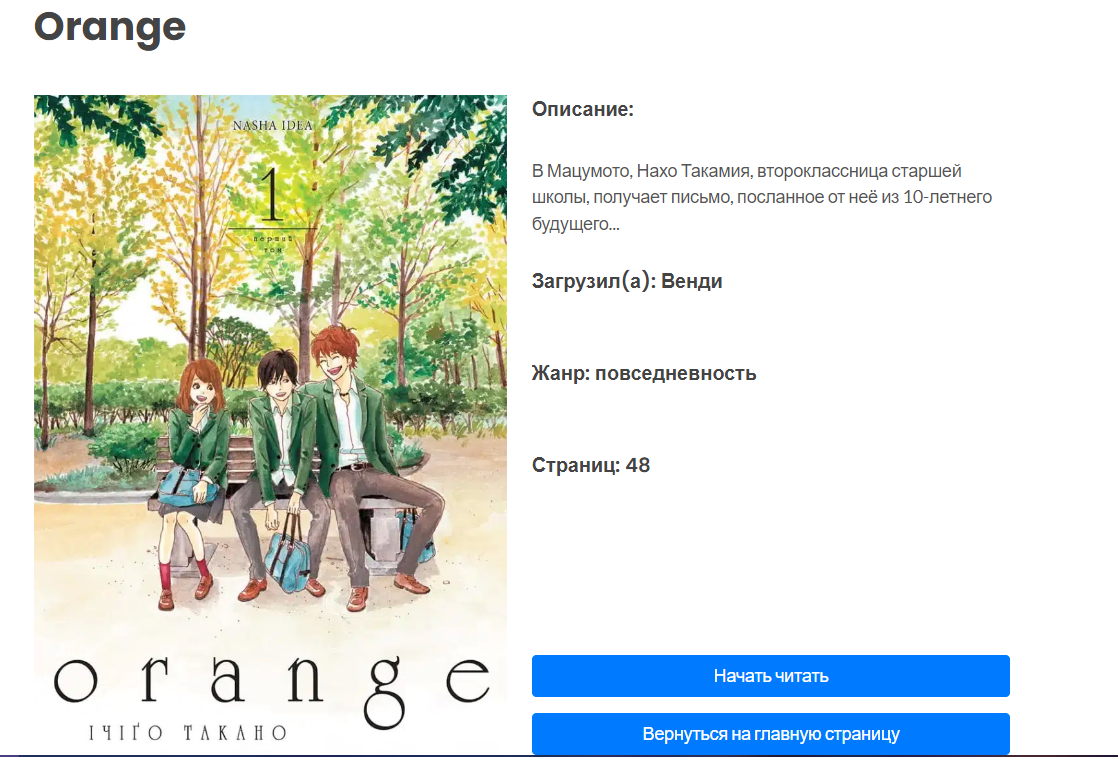
Функция read: отображает все страницы какого-либо комикса. Использует одноимённый шаблон.

Функция add\_page: отвечает за добавление страниц к какому-либо комиксу. Использует форму PageForm и одноимённый шаблон.

В созданных мною шаблонах используются bootstrap, jquery и другие файлы css и скрипты на языке JavaScript. Картинки, загруженные пользователями на сайт, хранятся в подпапке comics папки static.

Также c помощью сайта reg.ru я создала домен comic-cat.ru, на который загружен мой сайт.

Главная страница

Форум

# 

# 

Страница с информацией о комиксе

# ЗАКЛЮЧЕНИЕ

В ходе исследования я узнала, что идея сайта для просмотра комиксов является актуальной, потому что бумажные издания дорогостоящие и иногда их трудно найти, а также что форум для обсуждения комиксов будет пользоваться популярностью среди поклонников.

Я создала структуру сайта и базу данных для его использования. Комиксы взяты из открытых интернет-источников.

При реализации проекта я создала главную страницу и форум и организовала их совместную работу с базой данных. В итоге я получила работающий сайт.

В дальнейшем возможно расширить функционал портала, увеличить возможности управления, просмотра, администрирования, взаимодействия пользователей с комиксом.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 - <https://lyceum.yandex.ru/courses/766/groups/6082/lessons/3568/materials/9849> - информация о библиотеке Flask

2 - <https://lyceum.yandex.ru/courses/766/groups/6082/lessons/3569/materials/9781> - информация об использовании шаблонов в Flask

3 - <https://www.sqlalchemy.org/> - документация SqlAlchemy

# ПРИЛОЖЕНИЕ А

(обязательное)

**Листинг**

**main.py:**

import os

import hashlib

from flask import Flask, make\_response, jsonify, render\_template, redirect, request, abort

from flask\_login import LoginManager, current\_user, login\_user, logout\_user, login\_required

from sqlalchemy import desc

from data import db\_session

from data.discussions import Discussions

from data.genres import Genres

from data.posts import Posts

from data.users import User

from data.comics import Comics

from data.pages import Pages

from forms.discussion import DiscussionForm

from forms.user import RegisterForm, LoginForm

from forms.comics import ComicsForm

from forms.pages import PageForm

from werkzeug.utils import secure\_filename

from pathlib import Path

app = Flask(\_\_name\_\_)

app.config['SECRET\_KEY'] = 'yandexlyceum\_secret\_key'

login\_manager = LoginManager()

login\_manager.init\_app(app)

def main():

db\_session.global\_init("db/blogs.sqlite")

app.run()

@app.errorhandler(404)

def not\_found(error):

return make\_response(jsonify({'error': 'Not found'}), 404)

@app.errorhandler(400)

def bad\_request(\_):

return make\_response(jsonify({'error': 'Bad Request'}), 400)

@app.route("/")

def index():

db\_sess = db\_session.create\_session()

comics = db\_sess.query(Comics)

return render\_template("main\_page.html", comics=comics)

@app.route("/discussions/")

def discussions():

db\_sess = db\_session.create\_session()

discs = db\_sess.query(Discussions)

return render\_template("discussions.html", discs=discs)

@app.route("/comics/", methods=['GET', 'POST'])

@login\_required

def comics():

db\_sess = db\_session.create\_session()

genres = db\_sess.query(Genres).all()

groups\_list = [(i.id, i.name) for i in genres]

form = ComicsForm()

form.genres.choices = groups\_list

if form.validate\_on\_submit():

hash\_dir = hashlib.md5(form.name.data.encode('utf-8')).hexdigest()

f = form.cover.data

filename = secure\_filename(f.filename)

Path('static/comics/' + hash\_dir + '/').mkdir(parents=True, exist\_ok=True)

f.save(os.path.join(

'static/comics/' + hash\_dir + '/', filename

))

db\_sess = db\_session.create\_session()

comic = Comics(

name=form.name.data,

namefind=form.name.data.lower(),

about=form.about.data,

genres\_id=form.genres.data,

cover='comics/' + hash\_dir + '/' + filename

)

current\_user.comics.append(comic)

db\_sess.merge(comic)

db\_sess.commit()

return redirect('/')

return render\_template("comics.html", form=form)

@app.route('/register', methods=['GET', 'POST'])

def reqister():

form = RegisterForm()

if form.validate\_on\_submit():

if form.password.data != form.password\_again.data:

return render\_template('register.html', title='Регистрация',

form=form,

message="Пароли не совпадают")

db\_sess = db\_session.create\_session()

if db\_sess.query(User).filter(User.email == form.email.data).first():

return render\_template('register.html', title='Регистрация',

form=form,

message="Такой пользователь уже есть")

user = User(

name=form.name.data,

email=form.email.data,

about=form.about.data

)

user.set\_password(form.password.data)

db\_sess.add(user)

db\_sess.commit()

return redirect('/login')

return render\_template('register.html', title='Регистрация', form=form)

@login\_manager.user\_loader

def load\_user(user\_id):

db\_sess = db\_session.create\_session()

return db\_sess.query(User).get(user\_id)

@app.route('/login', methods=['GET', 'POST'])

def login():

form = LoginForm()

if form.validate\_on\_submit():

db\_sess = db\_session.create\_session()

user = db\_sess.query(User).filter(User.email == form.email.data).first()

if user and user.check\_password(form.password.data):

login\_user(user, remember=form.remember\_me.data)

return redirect("/")

return render\_template('login.html',

message="Неправильный логин или пароль",

form=form)

return render\_template('login.html', title='Авторизация', form=form)

@app.route('/logout')

@login\_required

def logout():

logout\_user()

return redirect("/")

@app.route('/discussions/discs', methods=['GET', 'POST'])

@login\_required

def add\_disc():

form = DiscussionForm()

if form.validate\_on\_submit():

db\_sess = db\_session.create\_session()

discs = Discussions()

discs.title = form.title.data

discs.content = form.content.data

current\_user.discussions.append(discs)

db\_sess.merge(current\_user)

db\_sess.commit()

return redirect('/discussions')

return render\_template('discs.html', title='Добавление новости',

form=form)

@app.route('/find', methods=['GET', 'POST'])

def find():

if request.method == 'POST':

content = request.form.get("findtext")

if content:

content = content.lower()

db\_sess = db\_session.create\_session()

findcomics = db\_sess.query(Comics).filter(Comics.namefind.contains(f"{content}"))

return render\_template("main\_page.html", comics=findcomics)

return redirect('/')

@app.route('/discussions/<int:id>', methods=['GET', 'POST'])

def discussion(id):

db\_sess = db\_session.create\_session()

disc = db\_sess.query(Discussions).filter(Discussions.id == id)[0]

if request.method == 'POST':

content = request.form.get("content")

if content:

post = Posts()

post.content = content

current\_user.posts.append(post)

post1 = db\_sess.merge(post)

disc.posts.append(post1)

db\_sess.merge(disc)

db\_sess.commit()

return redirect(f'/discussions/{id}')

posts = db\_sess.query(Posts).filter(Posts.discussion\_id == id)

leng = len([i for i in posts])

return render\_template("discussion.html", posts=posts, disc=disc, leng=leng)

@app.route('/users/<int:id>')

def user(id):

db\_sess = db\_session.create\_session()

user = db\_sess.query(User).filter(User.id == id)[0]

comics = list(db\_sess.query(Comics).filter(Comics.user\_id == id))

return render\_template("user.html", user=user, comics=comics)

@app.route("/comics/<int:id>")

def comic(id):

db\_sess = db\_session.create\_session()

comic = db\_sess.query(Comics).filter(Comics.id == id)[0]

try:

pages = db\_sess.query(Pages).filter(Pages.comics\_id == id).order\_by(desc(Pages.number)).limit(1)[0]

except IndexError:

pages = 0

return render\_template("comic.html", comic=comic, pages=pages)

@app.route("/comics/<int:id>/read")

def read(id):

db\_sess = db\_session.create\_session()

comic = db\_sess.query(Comics).filter(Comics.id == id)[0]

pages = list(db\_sess.query(Pages).filter(Pages.comics\_id == id).order\_by(Pages.number))

link = f"/comics/{id}"

return render\_template("read.html", pages=pages, link=link, comic=comic)

@app.route("/add\_page/<int:id>", methods=['GET', 'POST'])

@login\_required

def add\_page(id):

form = PageForm()

db\_sess = db\_session.create\_session()

comic = db\_sess.query(Comics).filter(Comics.id == id)[0]

try:

pages = db\_sess.query(Pages).filter(Pages.comics\_id == id).order\_by(desc(Pages.number)).limit(1)[0]

except IndexError:

pages = 0

if form.validate\_on\_submit():

hash\_dir = hashlib.md5(comic.name.encode('utf-8')).hexdigest()

f = form.page.data

filename = secure\_filename(f.filename)

Path('static/comics/' + hash\_dir + '/').mkdir(parents=True, exist\_ok=True)

f.save(os.path.join(

'static/comics/' + hash\_dir + '/', filename

))

db\_sess = db\_session.create\_session()

page = Pages(

number=int(form.number.data),

comics\_id=id,

page='comics/' + hash\_dir + '/' + filename

)

db\_sess.merge(page)

db\_sess.commit()

return redirect(f'/add\_page/{comic.id}')

return render\_template("add\_page.html", comic=comic, form=form, pages=pages)

@app.route('/comic\_delete/<int:id>', methods=['GET', 'POST'])

@login\_required

def comic\_delete(id):

db\_sess = db\_session.create\_session()

comic = db\_sess.query(Comics).filter(Comics.id == id,

Comics.user == current\_user

).first()

if comic:

db\_sess.delete(comic)

db\_sess.commit()

else:

abort(404)

return redirect('/')

@app.route('/disc\_delete/<int:id>', methods=['GET', 'POST'])

@login\_required

def disc\_delete(id):

db\_sess = db\_session.create\_session()

discs = db\_sess.query(Discussions).filter(Discussions.id == id,

Discussions.user == current\_user

).first()

if discs:

db\_sess.delete(discs)

db\_sess.commit()

else:

abort(404)

return redirect('/discussions')

@app.route('/post\_delete/<int:id>', methods=['GET', 'POST'])

@login\_required

def post\_delete(id):

db\_sess = db\_session.create\_session()

post = db\_sess.query(Posts).filter(Posts.id == id,

Posts.user == current\_user

).first()

if post:

db\_sess.delete(post)

db\_sess.commit()

else:

abort(404)

return redirect(f'/discussions/{post.discussion\_id}')

if \_\_name\_\_ == '\_\_main\_\_':

main()