DBMS LAB MINI PROJECT REPORT

Roll No.: COMP TE B 1223

Name: Darshan Sonawane

PRN: 72018566B

Title: A Movies Recommendation blog site using flask.

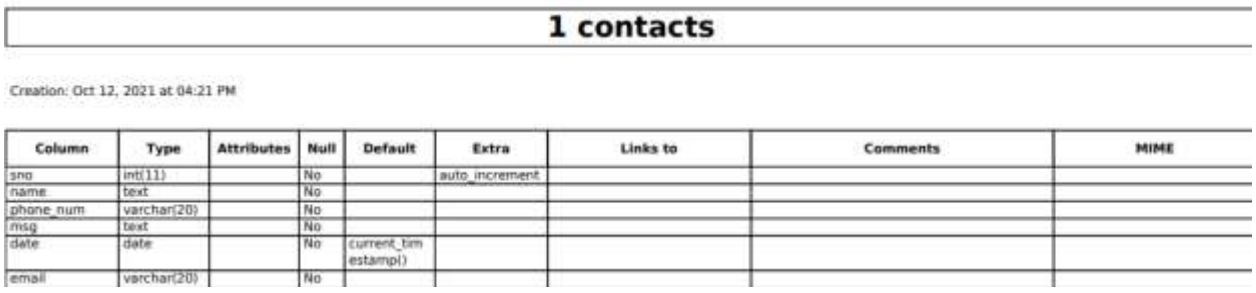
Abstract: In this project we developed a blog site using flask module in python. The site is developed using various tools like PhpMyAdmin, XAMPP, Bootstrap, etc. The templates are provided by Bootstrap as backend of the site and front is developed in python. We utilized the concept of DBMS that we have learned. The main component of site is its database, which is created on PhpMyAdmin and managed on it. Database connectivity is also utilized with python as front-end and Mysql as back-end.

Introduction: The project is about a simple blog site developed using flask module in python for movies recommendation. The mini project requirements are to use data base concepts. We created the blog site which uses data base created on localhost server. The data base connectivity also demonstrated using python as front –end language and Mysql as back-end language***.***

Software requirements:

1. Pycharm IDE (for running python code)
2. Sublime Text (to edit HTML files)
3. XAMPP (to connect computer to the local host server)
4. Myphp (to create and manage the database)
5. Bootstrap (for html templates)

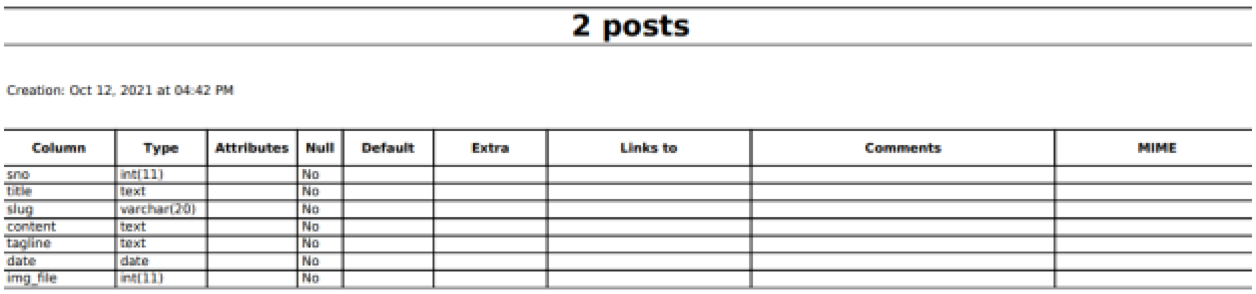
Design of Database:



1 contacts

Creation: Oct 12, 2021 at 04:21 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
sno	int(11)		No		auto_increment			
name	text		No					
phone_num	varchar(20)		No					
msg	text		No					
date	date		No	current_timestamp()				
email	varchar(20)		No					



2 posts

Creation: Oct 12, 2021 at 04:42 PM

Column	Type	Attributes	Null	Default	Extra	Links to	Comments	MIME
sno	int(11)		No					
title	text		No					
slug	varchar(20)		No					
content	text		No					
tagline	text		No					
date	date		No					
img_file	int(11)		No					

posts
sno
title
slug
content
tagline
date
img_file
contacts
sno
name
phone_num
msg
date
email

Source code:

from flask import Flask, render_template, request, session, redirect

from flask_sqlalchemy import SQLAlchemy

from flask_mail import Mail

```
import json
import os
import math
from werkzeug import secure_filename
from datetime import datetime

with open('config.json','r') as c:
    params = json.load(c) ["params"]

local_server = True
app = Flask(__name__)
app.secret_key = "super-secret-key"
app.config['UPLOAD_FOLDER'] = params['upload_location']
app.config.update(
    MAIL_SERVER = 'smtp.gmail.com',
    MAIL_PORT = '465',
    MAIL_USE_SSL = True,
    MAIL_USERNAME = params['gmail-user'],
    MAIL_PASSWORD = params['gmail-password']
)
mail = Mail(app)


if(local_server):
    app.config['SQLALCHEMY_DATABASE_URI'] = params['local_uri']
else:
    app.config['SQLALCHEMY_DATABASE_URI'] = params['prod_uri']

db = SQLAlchemy(app)

class Contacts(db.Model):
    sno = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(80),  nullable=False)
    phone_num = db.Column(db.String(12),  nullable=False)
    msg = db.Column(db.String(120), nullable=False)
    date = db.Column(db.String(12), nullable=True)
    email = db.Column(db.String(20), nullable=False)


class Posts(db.Model):
    sno = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(80),  nullable=False)
    slug = db.Column(db.String(12),  nullable=False)
    content = db.Column(db.String(120), nullable=False)
    tagline = db.Column(db.String(120), nullable=False)
    date = db.Column(db.String(12), nullable=True)
    img_file = db.Column(db.String(12), nullable=True)
```

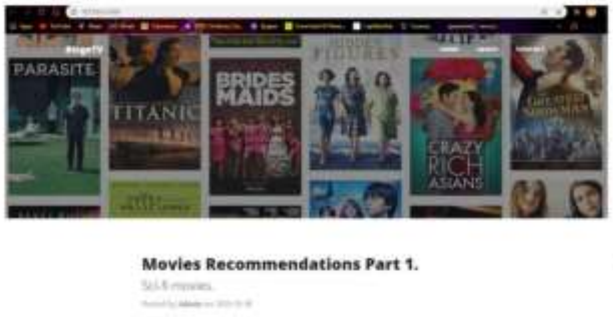
```
@app.route("/")
def home():
    posts = Posts.query.filter_by().all()
    last = math.ceil(len(posts) / int(params['no_of_post']))
    page = request.args.get('page')
    if (not str(page).isnumeric()):
        page = 1
    page = int(page)
    posts = posts[(page - 1) * int(params['no_of_post']):(page - 1) *
int(params['no_of_post']) + int(params['no_of_post'])]
    if page == 1:
        prev = "#"
        next = "/?page=" + str(page + 1)
    elif page == last:
        prev = "/?page=" + str(page - 1)
        next = "#"
    else:
        prev = "/?page=" + str(page - 1)
        next = "/?page=" + str(page + 1)


    return render_template('index.html', params=params, posts=posts, prev=prev,
next=next)


@app.route("/post/<string:post_slug>",methods=['GET'])
```

```
def post_route(post_slug):
    post = Posts.query.filter_by(slug=post_slug).first()



    return render_template('post.html',params=params, post=post)


@app.route("/about")
def about():
    return render_template('about.html',params=params)


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

    posts = Posts.query.all()
    return render_template('dashboard.html',params=params, posts=posts)
    if request.method=='POST':
        username = request.form.get('uname')
        userpass = request.form.get('pass')
        if(username == params['admin_user'] and userpass ==
params['admin_password']):
            session['user'] = username
            posts = Posts.query.all()
            return render_template('dashboard.html',params=params, posts=posts)

    return render_template('login.html', params=params)
```

```
@app.route("/edit/<string:sno>", methods=['GET','POST'])
def edit(sno):
    if request.method == 'POST':
        box_title = request.form.get('title')
        tline = request.form.get('tline')
        slug = request.form.get('slug')
        content = request.form.get('content')
        img_file = request.form.get('img_file')
        date = datetime.now()

        if sno == '0':
            post = Posts(title=box_title, slug=slug, content=content,
img_file=img_file, date=date)
            db.session.add(post)
            db.session.commit()

        else:
            post = Posts.query.filter_by(sno=sno).first()
            post.title = box_title
            post.slug = slug
            post.content = content
            post.img_file = img_file
            post.date = date
            db.session.commit()
            return redirect('/edit/'+ sno)

        post = Posts.query.filter_by(sno=sno).first()
        return render_template('edit.html', params=params, post=post)

@app.route("/uploader" , methods=['GET', 'POST'])
def uploader():
    if "user" in session and session['user'] == params['admin_user']:
        if request.method == "POST":
            f = request.files['file1']
            f.save(os.path.join(app.config['UPLOAD_FOLDER'],
secure_filename(f.filename)))
            return "Uploaded successfully!"



@app.route("/delete/<string:sno>" , methods=['GET', 'POST'])
def delete(sno):
    post = Posts.query.filter_by(sno=sno).first()
    db.session.delete(post)
    db.session.commit()
    return redirect("/dashboard")
```

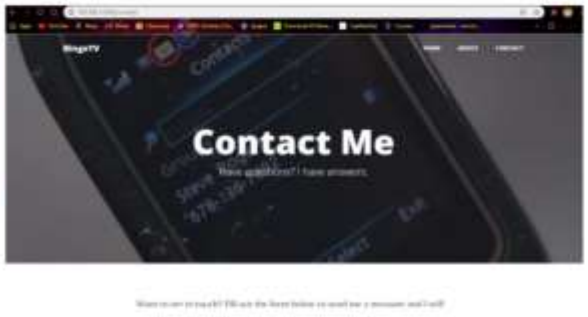
```
@app.route('/logout')
def logout():
    session.pop('user')
    return redirect('/dashboard')

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

    if(request.method=='POST'):
        name = request.form.get('name')
        email = request.form.get('email')
        phone = request.form.get('phone')
        message = request.form.get('message')
        entry = Contacts(name=name, phone_num=phone, msg=message,
date=datetime.now(), email=email)
        db.session.add(entry)
        db.session.commit()
        mail.send_message('New message from'+ name, sender=email,
recipients=[params['gmail-user']], body=message+"\n"+ phone)
    return render_template('contact.html',params=params)

app.run(debug=True)
```

Graphical Interface:



Movies Recommendations Part 1.

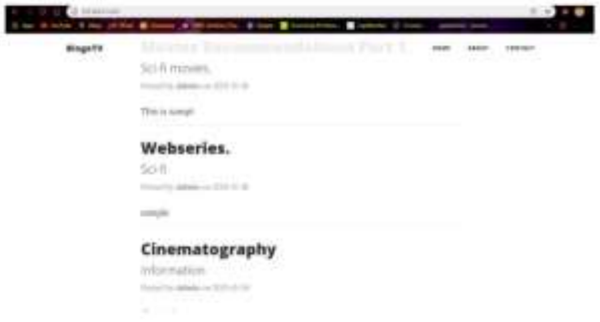
Sci-fi movies.



Contact Me

Have questions? I have answers.

Want to get in touch? Fill out the form below to send me a message and I will



Movies Recommendations Part 1.

Sci-fi movies.

This is sample

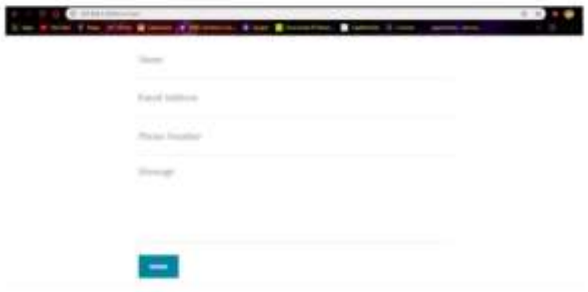
Webseries.

Sci-fi

sample

Cinematography

Information



Conclusion:

We have successfully implemented the project using concepts of database management.