Manarat International University

Department of Computer Science & Engineering

Software Development (CSE 312)

Team Information

- Team Name- 1453
- Rahat Hasan Akash
  - 1846CSE00694

Project Name: 1453_Blog (Web Based)

GitHub repository: https://github.com/Rahathasan1453/softDev_1453

Project Video: https://drive.google.com/file/d/1whKiO6PgaNKm4Ng4-as9Kdq6Idek5snj/view?usp=sharing

Objective

Our objective was to build up a personal Blogging Site. Where An admin can read, edit, update and add blog & a user can read the blogs.

Project Introduction

A personal blog is that sort of blog that has grown gradually over the span of years. A blog simply is a sort of website that mainly focuses on written content.

Personal blogs have gained a lot of popularity with time and have become a great source of sharing experiences, feelings, thoughts or even knowledge about anything and everything.

A blog is an online journal or informational website displaying information in reverse chronological order, with the latest posts appearing first, at the top. It is a platform where a writer or a group of writers share their views on an individual subject.

Feature Description

This is a personal blogging site. Here Admin can Add blog, Read Blog, Edit and Update the Blog. For login into Admin panel, admin have to log in into admin page with Email and Password.

A general viewer just can read the blog which are provided by the admin. But if he want he can sing in into the system and can add blog, he can write comment on specific blog, can share it on social media (future plan).

Language – Python3, HTML (Markup language)

Framework – Flask

Platform – Visual Studio Code

Libraries – Flask Libraries

Database – MySql Database

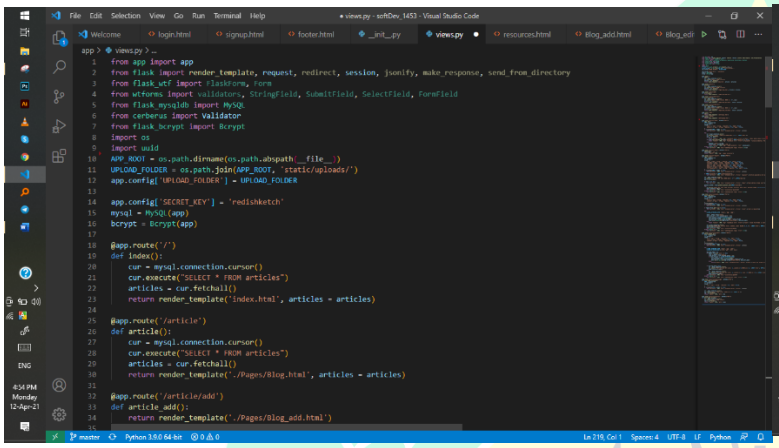
Problem Faced

We have started this project from the very beginning lavel knowledge of Flask framework . So we have faced many problem such as making environment, installing modules, add database with the main file etc. We got some help from The Internet & our friend Ahmed Zobayer helped a lot in this work. Hopefully those problem will help us to enrich our field of knowledge.

Future Plan

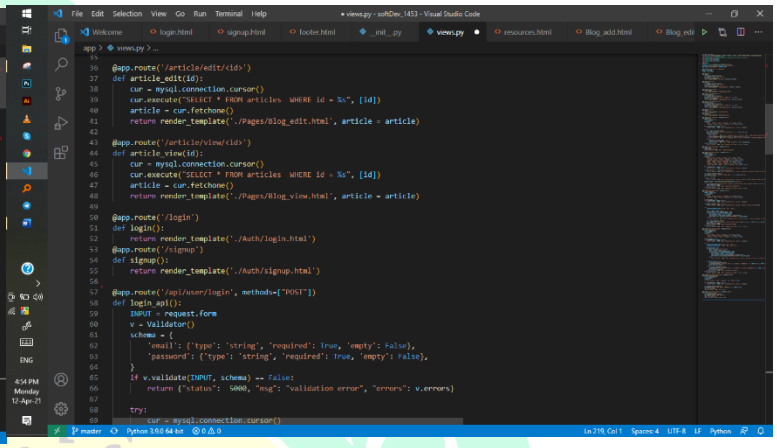
- Add the comment & Share section.
- Make an Android App of this Blog site.

Source Code



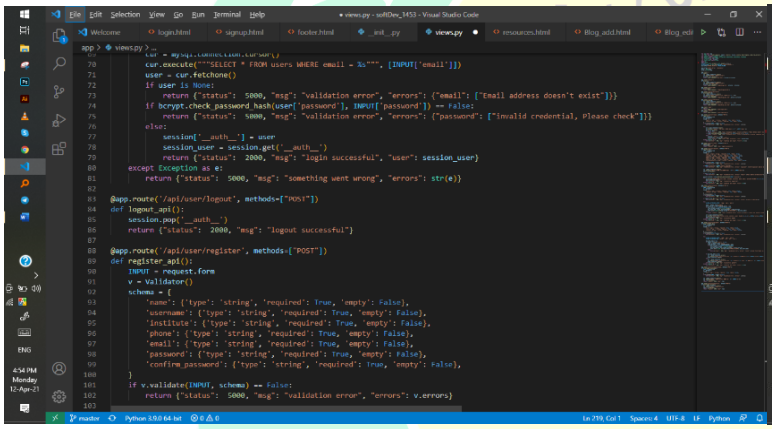
```python
from app import app
from flask import render_template, request, redirect, session, jsonify, make_response, send_from_directory
from flask_wtf import FlaskForm, Form
from wtforms import validators, StringField, SubmitField, SelectField, FormField
from flask_mysqldb import MySQL
from cerberus import Validator
from flask_bcrypt import Bcrypt
import os
import uuid
APP_ROOT = os.path.dirname(os.path.abspath(__file__))
UPLOAD_FOLDER = os.path.join(APP_ROOT, 'static/uploads/')
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

app.config['SECRET_KEY'] = 'redishketch'
mysql = MySQL(app)
bcrypt = Bcrypt(app)

@app.route('/')
def index():
    cur = mysql.connection.cursor()
    cur.execute("SELECT * FROM articles")
    articles = cur.fetchall()
    return render_template('index.html', articles = articles)

@app.route('/article')
def article():
    cur = mysql.connection.cursor()
    cur.execute("SELECT * FROM articles")
    articles = cur.fetchall()
    return render_template('../Pages/Blog.html', articles = articles)

@app.route('/article/add')
def article_add():
    return render_template('../Pages/Blog_add.html')
```

1



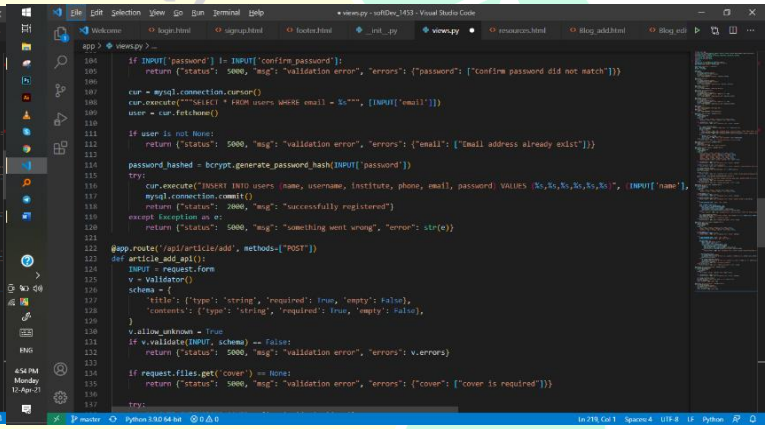
```python
@app.route('/article/edit/<id>')
def article_edit(id):
    cur = mysql.connection.cursor()
    cur.execute("SELECT * FROM articles  WHERE id = %s", [id])
    article = cur.fetchone()
    return render_template('./Pages/Blog_edit.html', article = article)

@app.route('/article/view/<id>')
def article_view(id):
    cur = mysql.connection.cursor()
    cur.execute("SELECT * FROM articles  WHERE id = %s", [id])
    article = cur.fetchone()
    return render_template('./Pages/Blog_view.html', article = article)

@app.route('/login')
def login():
    return render_template('./Auth/login.html')
@app.route('/signup')
def signup():
    return render_template('./Auth/signup.html')

@app.route('/api/user/login', methods=["POST"])
def login_api():
    INPUT = request.form
    v = Validator()
    schema = {
        'email': {'type': 'string', 'required': True, 'empty': False},
        'password': {'type': 'string', 'required': True, 'empty': False},
    }
    if v.validate(INPUT, schema) == False:
        return {"status":  5000, "msg": "validation error", "errors": v.errors}

    try:
        cur = mysql.connection.cursor()
```

2



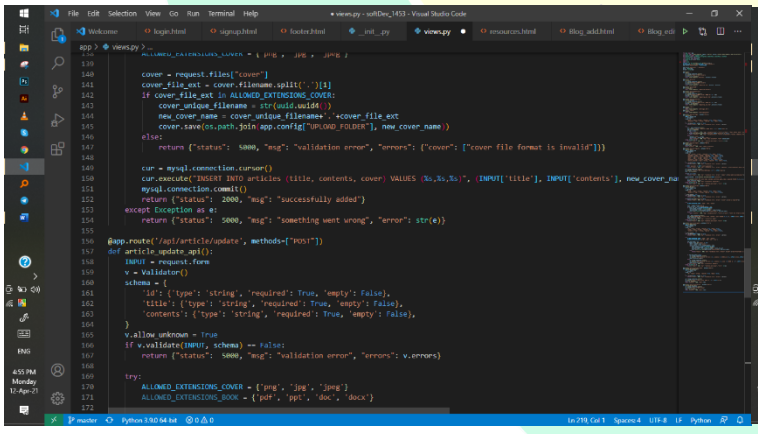
```python
        cur.execute("""SELECT * FROM users WHERE email = %s""", [INPUT['email']])
        user = cur.fetchone()
        if user is None:
            return {"status":  5000, "msg": "validation error", "errors": {"email": ["Email address doesn't exist"]}}
        if bcrypt.check_password_hash(user['password'], INPUT['password']) == False:
            return {"status":  5000, "msg": "validation error", "errors": {"password": ["Invalid credential, Please check"]}}
        else:
            session['__auth__'] = user
            session_user = session.get('__auth__')
            return {"status":  2000, "msg": "login successful", "user": session_user}
    except Exception as e:
        return {"status":  5000, "msg": "something went wrong", "errors": str(e)}

@app.route('/api/user/logout', methods=["POST"])
def logout_api():
    session.pop('__auth__')
    return {"status":  2000, "msg": "logout successful"}

@app.route('/api/user/register', methods=["POST"])
def register_api():
    INPUT = request.form
    v = Validator()
    schema = {
        'name': {'type': 'string', 'required': True, 'empty': False},
        'username': {'type': 'string', 'required': True, 'empty': False},
        'institute': {'type': 'string', 'required': True, 'empty': False},
        'phone': {'type': 'string', 'required': True, 'empty': False},
        'email': {'type': 'string', 'required': True, 'empty': False},
        'password': {'type': 'string', 'required': True, 'empty': False},
        'confirm_password': {'type': 'string', 'required': True, 'empty': False},
    }
    if v.validate(INPUT, schema) == False:
        return {"status":  5000, "msg": "validation error", "errors": v.errors}
```

3



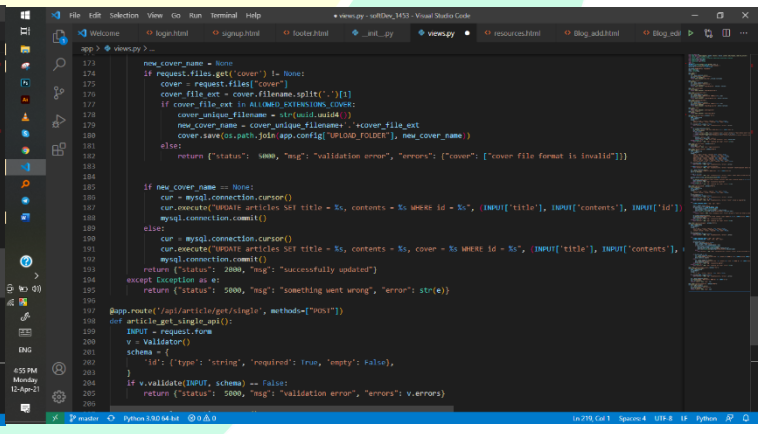
```python
    if INPUT['password'] != INPUT['confirm_password']:
        return {"status":  5000, "msg": "validation error", "errors": {"password": ["confirm password did not match"]}}

    cur = mysql.connection.cursor()
    cur.execute("""SELECT * FROM users WHERE email = %s""", [INPUT['email']])
    user = cur.fetchone()

    if user is not None:
        return {"status":  5000, "msg": "validation error", "errors": {"email": ["Email address already exist"]}}

    password_hashed = bcrypt.generate_password_hash(INPUT['password'])
    try:
        cur.execute("INSERT INTO users (name, username, institute, phone, email, password) VALUES (%s,%s,%s,%s,%s,%s)", (INPUT['name'],
        mysql.connection.commit()
        return {"status":  2000, "msg": "successfully registered"}
    except Exception as e:
        return {"status":  5000, "msg": "something went wrong", "error": str(e)}

@app.route('/api/article/add', methods=["POST"])
def article_add_api():
    INPUT = request.form
    v = Validator()
    schema = {
        'title': {'type': 'string', 'required': True, 'empty': False},
        'contents': {'type': 'string', 'required': True, 'empty': False},
    }
    v.allow_unknown = True
    if v.validate(INPUT, schema) == False:
        return {"status":  5000, "msg": "validation error", "errors": v.errors}

    if request.files.get('cover') == None:
        return {"status":  5000, "msg": "validation error", "errors": {"cover": ["cover is required"]}}

    try:
```

4



```python
        cover = request.files["cover"]
        cover_file_ext = cover.filename.split('.')[1]
        if cover_file_ext in ALLOWED_EXTENSIONS_COVER:
            cover_unique_filename = str(uuid.uuid4())
            new_cover_name = cover_unique_filename+'.'+cover_file_ext
            cover.save(os.path.join app.config["UPLOAD_FOLDER"], new_cover_name))
        else:
            return {"status":  5000, "msg": "validation error", "errors": {"cover": ["cover file format is invalid"]}}

        cur = mysql.connection.cursor()
        cur.execute("INSERT INTO articles (title, contents, cover) VALUES (%s,%s,%s)", (INPUT['title'], INPUT['contents'], new_cover_na
        mysql.connection.commit()
        return {"status":  2000, "msg": "successfully added"}
    except Exception as e:
        return {"status":  5000, "msg": "something went wrong", "error": str(e)}

@app.route('/api/article/update', methods=["POST"])
def article_update_api():
    INPUT = request.form
    v = Validator()
    schema = {
        'id': {'type': 'string', 'required': True, 'empty': False},
        'title': {'type': 'string', 'required': True, 'empty': False},
        'contents': {'type': 'string', 'required': True, 'empty': False},
    }
    v.allow_unknown = True
    if v.validate(INPUT, schema) == False:
        return {"status":  5000, "msg": "validation error", "errors": v.errors}

    try:
        ALLOWED_EXTENSIONS_COVER = {'png', 'jpg', 'jpeg'}
        ALLOWED_EXTENSIONS_BOOK = {'pdf', 'ppt', 'doc', 'docx'}
```

5



```python
        new_cover_name = None
        if request.files.get('cover') != None:
            cover = request.files["cover"]
            cover_file_ext = cover.filename.split('.')[1]
            if cover_file_ext in ALLOWED_EXTENSIONS_COVER:
                cover_unique_filename = str(uuid.uuid4())
                new_cover_name = cover_unique_filename+'.'+cover_file_ext
                cover.save(os.path.join app.config["UPLOAD_FOLDER"], new_cover_name))
            else:
                return {"status":  5000, "msg": "validation error", "errors": {"cover": ["cover file format is invalid"]}}


        if new_cover_name == None:
            cur = mysql.connection.cursor()
            cur.execute("UPDATE articles SET title = %s, contents = %s WHERE id = %s", (INPUT['title'], INPUT['contents'], INPUT['id'])
            mysql.connection.commit()
        else:
            cur = mysql.connection.cursor()
            cur.execute("UPDATE articles SET title = %s, contents = %s, cover = %s WHERE id = %s", (INPUT['title'], INPUT['contents'],
            mysql.connection.commit()
        return {"status":  2000, "msg": "successfully updated"}
    except Exception as e:
        return {"status":  5000, "msg": "something went wrong", "error": str(e)}

@app.route('/api/article/get/single', methods=["POST"])
def article_get_single_api():
    INPUT = request.form
    v = Validator()
    schema = {
        'id': {'type': 'string', 'required': True, 'empty': False},
    }
    if v.validate(INPUT, schema) == False:
        return {"status":  5000, "msg": "validation error", "errors": v.errors}
```

6

```python
    cur = mysql.connection.cursor()
    cur.execute("SELECT * FROM library  WHERE id = %s", (INPUT['id']))
    row = cur.fetchone()
    return {"status":  2000, "data": row}

@app.route('/api/article/get/all', methods=["POST"])
def article_get_all_api():
    cur = mysql.connection.cursor()
    cur.execute("SELECT * FROM library")
    rows = cur.fetchall()
    return {"status":  2000, "data": rows}
```

7

Conclusion

A personal Blog is one of the platforms where a person can share their views and knowledge and openion. It can also help him to gather his knowledge, thoughts.

To build up this site, we have used Python language for the operations and functions, Flask framework and MySql Database. To develop this site, we have faced many difficulties which ‘ll help us in near future.