آرمان افشارنادری

39910141054183

پروژه پایگاه داده : وب سایت پرسش و پاسخ

مستندات پروژه وب‌سایت پرسش و پاسخ

این پروژه یک وب‌سایت پرسش و پاسخ است که با استفاده از **فریم‌ورک Flask** به زبان **پایتون** توسعه داده شده و از **SQL Server**به عنوان پایگاه داده استفاده می‌کند. همچنین برای مدیریت و استقرار پروژه از **Docker**و برای امکانات هوش مصنوعی از **کتابخانه‌های AI** بهره می‌بریم.

مقدمه

این پروژه یک وب‌سایت پرسش و پاسخ است که با استفاده از فریم‌ورک Flask در پایتون توسعه داده شده است. پایگاه داده پروژه SQL Server بوده و از Docker برای مدیریت و استقرار بهره می‌گیرد. همچنین امکاناتی نظیر تشخیص صدا و هوش مصنوعی در بخش‌هایی از پروژه تعبیه شده است.

تکنولوژی‌های استفاده‌شده

- **زبان برنامه‌نویسی**) Python :نسخه 3.9 یا بالاتر(
- **فریم‌ورک**Flask :
- **پایگاه داده**SQL Server :
- **استقرار**Docker :
- **هوش مصنوعی** :استفاده از کتابخانه‌های پردازش زبان طبیعی (مثل transformers)

معماری پروژه

این پروژه از معماری **MVC**(مدل، ویو، کنترلر) استفاده می‌کند:

- **مدل** :مدیریت داده‌ها و ارتباط با پایگاه دادهSQL Server
- **ویو** :قالب‌های HTML و فایل‌هایJinja
- **کنترلر** :مدیریت لاجیک برنامه و پاسخ به درخواست‌ها

ویژگی‌های هوش مصنوعی

- پردازش سوالات کاربران با استفاده از مدل‌های هوش مصنوعی
- پیشنهاد پاسخ‌های هوشمند بر اساس داده‌های موجود
- تحلیل زبان طبیعی برای درک بهتر سوالات

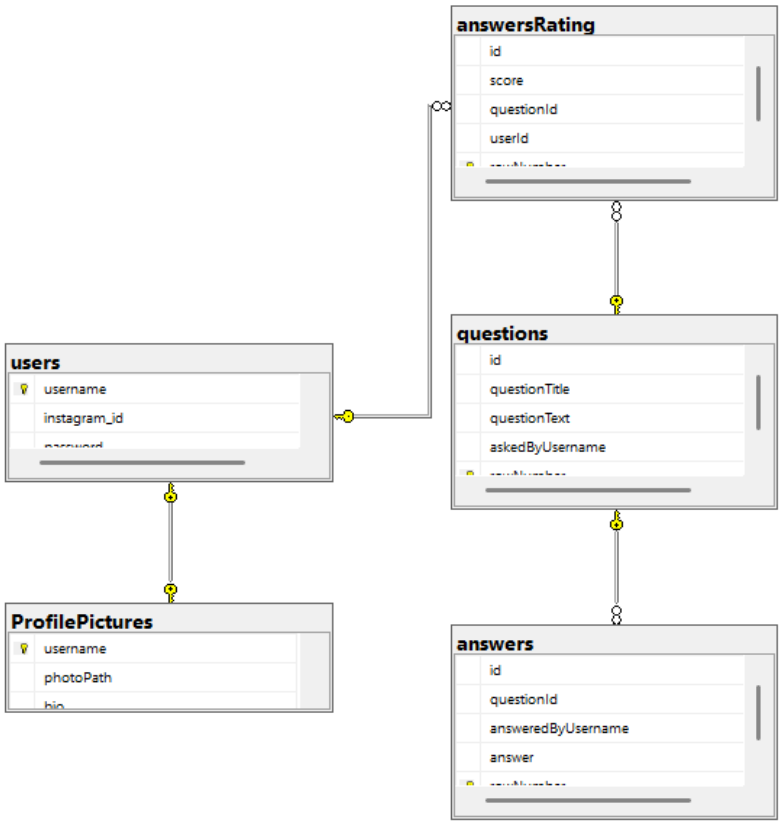
ساختار پروژه

- :**app.py**شامل روت‌ها و منطق اصلی برنامه
- :**models/sql_server_orm.py**مدیریت ارتباط با پایگاه دادهSQL Server
- :**templates/**فایل‌های HTML برای نمایش صفحات
- :**static/**شامل فایل‌های استاتیکCSS) ، JavaScriptو(...
- :**uploads/**پوشه‌ای برای ذخیره فایل‌های آپلود شده

امکانات پروژه

1. **ثبت‌نام و ورود کاربران**
   - مدیریت حساب کاربری با استفاده از ایمیل و رمز عبور
2. **پرسش و پاسخ**
   - ارسال سوالات و دریافت پاسخ از کاربران دیگر
3. **تشخیص صدا**
   - امکان آپلود فایل صوتی و تبدیل آن به متن
4. **مدیریت پروفایل کاربری**
   - افزودن اطلاعات شخصی و آپلود تصویر پروفایل

کد ها :


answersRating
id
score
questionId
userId
questions
id
questionTitle
questionText
askedByUsername
users
username
instagram_id
ProfilePictures
username
photoPath
answers
id
questionId
answeredByUsername
answer

database_queries.sql

models > database_queries.sql

```
GO

CREATE TABLE [dbo].[questions] (
    [id] NVARCHAR(50) NOT NULL,
    [questionTitle] NVARCHAR(255) NOT NULL,
    [questionText] NVARCHAR(MAX) NOT NULL,
    [askedByUsername] NVARCHAR(50) NOT NULL,
    [rowNumber] INT IDENTITY(1,1) NOT NULL,
    PRIMARY KEY ([rowNumber]),
    CONSTRAINT UQ_questions_id UNIQUE ([id])
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY];

CREATE TABLE [dbo].[answers] (
    [id] NVARCHAR(50) NOT NULL,
    [questionId] NVARCHAR(50) NOT NULL,
    [answeredByUsername] NVARCHAR(50) NULL,
    [answer] NVARCHAR(MAX) NOT NULL,
    [rowNumber] INT IDENTITY(1,1) NOT NULL,
    PRIMARY KEY ([rowNumber]),
    CONSTRAINT FK_answers_questions FOREIGN KEY ([questionId]) REFERENCES [dbo].[questions]([id]),
    CONSTRAINT UQ_answers_id UNIQUE ([id])
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY];

CREATE TABLE [dbo].[answersRating] (
    [id] NVARCHAR(50) NOT NULL,
    [score] INT NOT NULL,
    [questionId] NVARCHAR(50) NOT NULL,
    [userId] NVARCHAR(50) NOT NULL,
    [rowNumber] INT IDENTITY(1,1) NOT NULL,
    PRIMARY KEY ([rowNumber]),
    CONSTRAINT FK_answersRating_questions FOREIGN KEY ([questionId]) REFERENCES [dbo].[questions]([id]),
    CONSTRAINT FK_answersRating_users FOREIGN KEY ([userId]) REFERENCES [dbo].[users]([username]),
    CONSTRAINT UQ_answersRating_id UNIQUE ([id])
) ON [PRIMARY];

CREATE TABLE [dbo].[users] (
    [username] NVARCHAR(50) NOT NULL,
```

```sql
) ON [PRIMARY];

CREATE TABLE [dbo].[users] (
    [username] NVARCHAR(50) NOT NULL,
    [instagram_id] NVARCHAR(50) NULL,
    [password] NVARCHAR(100) NOT NULL,
    PRIMARY KEY ([username])
) ON [PRIMARY];

CREATE TABLE [dbo].[ProfilePictures] (
    [username] NVARCHAR(50) NOT NULL,
    [photoPath] NVARCHAR(MAX) NULL,
    PRIMARY KEY CLUSTERED ([username] ASC)
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY];
GO

ALTER TABLE [dbo].[ProfilePictures]
WITH CHECK ADD CONSTRAINT [FK_photo_user]
FOREIGN KEY ([username]) REFERENCES [dbo].[users] ([username]);
GO


ALTER TABLE [dbo].[ProfilePictures]
CHECK CONSTRAINT [FK_photo_user];
```

:ORM

```python
import pyodbc
import bcrypt
import base64
import asyncio

def connect_to_db():
    DRIVER = '{ODBC Driver 17 for SQL Server}'
    SERVER = '.'
    DATABASE = 'Flask'
    conn = pyodbc.connect(f'DRIVER={DRIVER};' +
                          f'SERVER={SERVER};' +
                          f'DATABASE={DATABASE};' +
                          'Trusted_Connection=yes;')
    cursor = conn.cursor()
    return conn,cursor

def hash_password(password):
    salt = bcrypt.gensalt()
    hashed_password = bcrypt.hashpw(password.encode('utf-8'), salt)
    return base64.b64encode(hashed_password).decode('utf-8')

def verify_password(stored_password, entered_password):
    stored_password_bytes = base64.b64decode(stored_password)
    return bcrypt.checkpw(entered_password.encode('utf-8'), stored_password_bytes)

def search_user(username):
    conn, cursor = connect_to_db()
    cursor.execute("SELECT * FROM users WHERE username = ?", (username,))
    try :
        username, instagram_id, password = cursor.fetchone()
        conn.close()
        return {'username': username, 'instagram_id': instagram_id, 'password': password}
    except TypeError:
        conn.close()
        return dict()

```

```python

def create_user(username, instagram_id,password):
    conn,cursor = connect_to_db()
    hashed_password = hash_password(password)
    cursor.execute("""
    INSERT INTO users (username, instagram_id, password) VALUES (?, ?, ?)
    """,(username, instagram_id, hashed_password,))
    conn.commit()
    conn.close()


def add_question(question_title, question_body, username):
    conn, cursor = connect_to_db()
    query = """
    INSERT INTO questions (id, questionTitle, questionText, askedByUsername)
    VALUES (newid(), ?, ?, ?)
    """
    cursor.execute(query, (question_title, question_body, username))
    conn.commit()
    conn.close()

def get_questions():
    conn, cursor = connect_to_db()
    cursor.execute("SELECT * FROM Flask..questions")
    return cursor.fetchall()

def get_answers(question_id):
    conn, cursor = connect_to_db()
    cursor.execute("SELECT * FROM Flask..answers WHERE questionId = ?", (question_id,))
    return cursor.fetchall()


def add_answer(question_id, answered_username, answer):
    conn, cursor = connect_to_db()
    cursor.execute("""
```

:FRONT

```html
<!DOCTYPE html>
<html lang="fa">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Forgot Password - Code Hub</title>
  <link rel="stylesheet" href="../static/forget.css">

  <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap" rel="stylesheet">
</head>
<body>

  <header class="showcase">


    <section class="forgot-password-section">
      <div class="container center">
        <h3>بازیابی رمز عبور</h3>
        <p>لطفا ایمیل خود را جهت بازیابی رمز عبور وارد نمایید</p>
        <form action="/forget" method="post">
          <div class="input-group">
                <label for="email">{{ TEXT }}</label>
                <input type="text" id="email" name="email" required>
          </div>
          <button type="submit" class="btn">ارسال</button>
        </form>
        <p>وارد شوید>"/login"=href  "l2"=id a<رمز عبور خود را به یاد دارید؟</a></p>
      </div>
    </section>

  </header>

</body>
</html>
```

```html
<!DOCTYPE html>
<html lang="fa">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>جامعه تک - Code Hub</title>
  <link rel="stylesheet" href="../static/home.css">
  <link href="https://cdn.jsdelivr.net/gh/rastikerdar/vazir-font@v30.1.0/dist/font-face.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <header class="showcase">
    <div class="content">
      <img src="https://image.ibb.co/ims4Ep/logo.png" class="logo" alt="Traversy Media">
      <div class="title">جامعه تک</div>

      <div class="text"> <h1>ارتقاء نوآوری‌ها و رامحل‌های نوین</h1></div>
    </div>
  </header>

  <section class="services">
    <div class="container grid-1 center">
      <div>
        <i class="fas fa-chalkboard-teacher fa-3x"></i>
        <p>اینجا در کنار یکدیگر رشد کنید و مهارت های خود را ارتقابدهید</p>
          <form action="/home" method="post">
{#        <button id="btn-learnMore">بیشتر بدانید</button>#}
        <button class="btn">شروع کنید</button>
         </form>
      </div>
    </div>
  </section>
</body>
</html>
```

```html
<!DOCTYPE html>
<html lang="fa">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>ورود به حساب - Code Hub</title>
  <link rel="stylesheet" href="../static/login.css">

  <link href="https://cdn.jsdelivr.net/gh/rastikerdar/vazir-font@v30.1.0/dist/font-face.css" rel="stylesheet" type="text/css" />
    <style>
        .alert-for-wrong-user {
            font-size: 18px;
            color: red !important;
        }
    </style>
</head>
<body>
  <header class="showcase">


    <section class="login">
      <div class="container center">
        <h3>ورود به حساب کاربری شما</h3>
          {% if error %}
                <div class="alert-for-wrong-user" role="alert">
                {{ error }}
                </div>
            {% endif %}
        <form action="/login" method="post">
          <div class="input-group">
            <label for="username">نام کاربری</label>
            <input type="text" id="username" name="username" required>
          </div>
          <div class="input-group">
```

```
<!DOCTYPE html>
<html lang="fa">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>پروفایل کاربری - Code Hub</title>
    <link rel="stylesheet" href="../static/profile.css"> <!-- Link to profile CSS -->
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">
</head>
<body>
    <!-- Profile Header -->
    <header class="profile-header">
        <div class="container">
```

ith-Flask / templates / **profile.html**

Blame 53 lines (51 loc) · 2.32 KB

```
        </div>
    </header>

    <!-- Main Content -->
    <main class="profile-content">
        <div class="container">
            <!-- Recent Activity Section -->
            <section class="recent-activity">
                <h2>فعالیت‌های اخیر</h2>
                <ul>
                    <li><a href="#">پرسش جدیدی که کاربر ایجاد کرده است</a></li>
                    <li><a href="#">پاسخی که کاربر داده است</a></li>
                    <li><a href="#">لایک‌ها و نظرات اخیر</a></li>
                </ul>
            </section>

            <!-- Account Settings Section -->
            <section class="account-settings">
                <h2>تنظیمات حساب</h2>
                <form action="/update-profile" method="post">
                    <div class="input-group">
                        <label for="email">ایمیل</label>
                        <input type="email" id="email" name="email" value="user@example.com" required>
                    </div>
                    <div class="input-group">
                        <label for="bio">توضیحات</label>
                        <textarea id="bio" name="bio" rows="4">توضیحات کوتاه درباره کاربر</textarea>
                    </div>
                    <button type="submit" class="btn">به‌روزرسانی</button>
                </form>
            </section>
        </div>
    </main>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="fa">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>جامعه تکنولوژی - پاسخ‌ها</title>
  <link rel="stylesheet" href="../static/viewAnswer.css">
  <link href="https://cdn.jsdelivr.net/gh/rastikerdar/vazir-font@v30.1.0/dist/font-face.css" rel="stylesheet" type="text/css" />
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">
  <style>
    /* Modal styles */
    .modal {
      display: none;
      position: fixed;
      z-index: 1;
      left: 0;
      top: 0;
      width: 100%;
      height: 100%;
      overflow: auto;
      background-color: rgb(0,0,0);
      background-color: rgba(0,0,0,0.4);
    }
    .modal-content {
      background-color: #fefefe;
      margin: 15% auto;
      padding: 20px;
      border: 1px solid #888;
      width: 80%;
      max-width: 500px;
    }
    .close {
      color: #aaa;
      float: right;
      font-size: 28px;
      font-weight: bold;
    }
    .close:hover,
    .close:focus {
      color: black;
      text-decoration: none;
      cursor: pointer;
    }
  </style>
</head>
<body>

  <!-- Background Showcase -->
  <header class="showcase">
    <div class="content">
      <img src="https://image.ibb.co/ims4Ep/logo.png" class="logo" alt="Tech Community">
    </div>
  </header>

  <!-- Navigation Bar -->
  <nav class="navbar">
    <div class="container">
      <a href="/questions" class="logo">جامعه تک</a>
      <ul class="nav-links">
        <li><a href="/home">خانه</a></li>
        <li><a href="/questions">سوالات</a></li>
        <li><a href="/topUsers">نفرات برتر</a></li>
        <li><a href="/logout">خروج</a></li>
      </ul>
    </div>
  </nav>
```

```
<script>
  var modal = document.getElementById("starModal");
  var span = document.getElementsByClassName("close")[0];
  var rateLinks = document.getElementsByClassName("rate-link");

  for (var i = 0; i < rateLinks.length; i++) {
    rateLinks[i].onclick = function(event) {
      event.preventDefault();
      var username = this.getAttribute("data-username");
      document.getElementById("usernameValue").value = username;
      modal.style.display = "block";
    }
  }

  span.onclick = function() {
    modal.style.display = "none";
  }

  window.onclick = function(event) {
    if (event.target == modal) {
      modal.style.display = "none";
    }
  }

  // Add event listener to update the hidden ratingValue input when a star is clicked
  var rateInputs = document.getElementsByName('rate');
  for (var i = 0; i < rateInputs.length; i++) {
    rateInputs[i].onclick = function() {
      document.getElementById("ratingValue").value = this.value;
    }
  }
</script>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="fa">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>ایجاد حساب کاربری - Code Hub</title>
  <link rel="stylesheet" href="../static/signUp.css">
  <link href="https://cdn.jsdelivr.net/gh/rastikerdar/vazir-font@v30.1.0/dist/font-face.css" rel="stylesheet" type="text/css" />
    <style>
        .alert-for-taken-user {
            font-size: 18px;
            color: red !important;
        }
    </style>
</head>
<body>
  <header class="showcase">
    <section class="signup">
      <div class="container center">
        <h3>ایجاد حساب کاربری</h3>
          {% if error %}
            <div class="alert-for-taken-user" role="alert">
                {{ error }}
            </div>
          {% endif %}
        <form action="/signup" method="post">
          <div class="input-group">
            <label for="email">ایمیل</label>
            <input type="email" id="email" name="email" required>
          </div>
          <div class="input-group">
            <label for="instagram-id">آیدی اینستاگرام</label>
            <input type="text" id="instagram-id" name="instagram-id" required>
          </div>
          <div class="input-group">
            <label for="password">رمز عبور</label>
            <input type="password" id="password" name="password" required>
          </div>
          <div class="input-group">
            <label for="repeat-password">تکرار رمز عبور</label>
            <input type="password" id="repeat-password" name="repeat-password" required>
          </div>
          <button type="submit" class="btn">ثبت نام</button>

           <script>
                function validatePassword() {
                    const passwordInput = document.getElementById('password');
                    const errorMessage = document.getElementById('error-message');

                    if (passwordInput.value.length < 8) {
                        errorMessage.textContent = "Password must be at least 8 characters long.";
```

```python
from flask import (
    Flask, render_template,
    url_for, redirect, request,
    session,abort,jsonify
)
from uuid import uuid4
from instagram_bot import *
from models.sql_server_orm import *
from question_shortener import questionShortener
from werkzeug.exceptions import BadRequest
import os
import soundfile as sf
import av
import speech_recognition as sr
from werkzeug.utils import secure_filename


app = Flask(__name__)
app.secret_key = uuid4().hex
user_section = dict()
user_is_login = dict()

UPLOAD_FOLDER = 'uploads/'
ALLOWED_EXTENSIONS = {'wav','mp3'}
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
```

```python
def recognize_speech_from_wav(wav_path: str) -> str:
    recognizer = sr.Recognizer()
    try:
        with sf.SoundFile(wav_path) as audio_file:
            logging.info(f"Processing file - Sample rate: {audio_file.samplerate}, Channels: {audio_file.channels}")

        with sr.AudioFile(wav_path) as source:
            audio = recognizer.record(source)
        text = recognizer.recognize_google(audio, language='fa-IR')

        logging.info(f"Recognized text: {text}")
        return text
    except sr.UnknownValueError:
        return "Google Speech Recognition could not understand audio"
    except sr.RequestError as e:
        return f"Could not request results from Google Speech Recognition service; {e}"
    except Exception as e:
        return f"An error occurred: {str(e)}"

def convert_webm_to_wav(input_file_path, output_file_path):
    input_container = av.open(input_file_path)
    output_container = av.open(output_file_path, 'w')

    stream = output_container.add_stream('pcm_s16le')

    for frame in input_container.decode(audio=0):
        for packet in stream.encode(frame):
            output_container.mux(packet)

    for packet in stream.encode(None):
        output_container.mux(packet)

    input_container.close()
```

```python
                return redirect('/login')
            else:
                context['error'] = 'رمز باید بیشتر از 8 حرف باشد'
        else:
            context['error'] = 'این ایمیل موجود میباشد'
    return render_template('signup.html',**context)


@app.route('/forget',methods=['GET','POST'])
def forget():
    print(0)
    code = str(uuid4())[:5]
    if request.method == "POST":
        if 'answer-code' not in session:
            username = request.form['email']
            print(1)
            username = search_user(username)['instagram_id']
            cl = start(INSTAGRAM_USERNAME,INSTAGRAM_PASSWORD)
            user_id = str(find_user_id(cl,username))
            if user_id:
                send_direct_from_specific_comment(cl, list_of_users=[user_id], ANSWER=code)
                user_section[request.remote_addr] = code
                session['answer-code'] = True
                return render_template('forget.html', TEXT='کد را وارد کنید')
        else:
            print(2)
            input_code = request.form['email']
            if input_code == user_section[request.remote_addr]:
                session['Email'] = request.form['email']
                return redirect('/questions')
            else:
                return redirect('/login')

    return render_template('forget.html',TEXT='ایمیل')

@app.route('/search', methods=['GET'])
```

```python

@app.route('/search', methods=['GET'])
def searchQuestion():
    query = request.args.get('query')
    print("this is query: ", query.encode('utf-8'))
    if query:
        filtered_question = search_question(query)
    else:
        filtered_question = get_questions()
    return render_template('questions.html', questions=filtered_question)


@app.route('/questions' ,methods=['GET', 'POST'])
def questions():
    if request.method == "POST":
        if question := request.form['question']:
            title = questionShortener(question)
            add_question(title,question,session['Email'])
    Questions = get_questions()
    return render_template('questions.html',questions=Questions)


@app.route('/questions/<int:question_id>', methods=['GET', 'POST'])
def question_detail(question_id):
    if request.method == 'POST':
        if 'ratingValue' in request.form and request.form['ratingValue']:
            rate = request.form.get('rate')
            user_id = request.form.get('usernameValue')
            add_rating(rate, question_id, user_id)
            return redirect(request.referrer)
        if 'answer' in request.form:
            answer = request.form['answer']
            try:
                add_answer(question_id, session.get('Email', 'unknown'), answer)
                return redirect(f'/questions/{question_id}')
            except BadRequest:
                return redirect(f'/questions/{question_id}')
```

```
179                     return redirect(f'/questions/{question_id}')
180
181         Questions = get_questions()
182         question_keys = [item[-1] for item in Questions]
183         question_dict = dict(zip(question_keys, Questions))
184         isAdmin = question_dict[question_id][-2] == session.get('Email', '')
185         answers = get_answers(question_id)
186         return render_template('question_detail.html',
187                                question=question_dict[question_id],
188                                answers=answers,
189                                question_id=question_id,
190                                isAdmin=True)
191
192     @app.route('/topUsers',methods=['GET'])
193     def topUsers():
194         users = get_top_users()
195         return render_template('topUsers.html',users=users)
196
197     @app.route('/myProfile',methods=['GET','POST'])
198     def myProfile():
199         return render_template('myProfile.html')
200
201     @app.route("/logout" ,methods=['GET'])
202     def logout():
203         session.pop("Email", None)
204         user_is_login.pop(request.remote_addr, None)
205         return render_template('home.html')
206
```

Q-A-with-Flask / **app.py**

Code | Blame | 246 lines (203 loc) · 8.75 KB

```
208     def voice():
213                 Questions = get_questions()
214                 return render_template('questions.html',questions=Questions,
215                                        question_data = text)
216
217             else:
218                 if 'audio' not in request.files:
219                     return jsonify({"error": "No audio file in request"}), 400
220
221                 audio = request.files['audio']
222                 if audio.filename == '':
223                     return jsonify({"error": "No selected file"}), 400
224
225                 secure_path = str(uuid4())
226
227                 filename = secure_filename(f'{secure_path}.webm')
228                 webm_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
229                 audio.save(webm_path)
230
231                 wav_filename = secure_filename(f'{secure_path}.wav')
232                 wav_path = os.path.join(app.config['UPLOAD_FOLDER'], wav_filename)
233
234                 convert_webm_to_wav(webm_path, wav_path)
235
236                 session['path'] = wav_path
237                 os.remove(webm_path)
238
239                 return jsonify({"message": "Audio uploaded successfully"}), 200
240
241         return render_template('voice.html')
242
```

```python
from instagrapi import types
from typing import List, AnyStr
from instagrapi import Client
from pathlib import Path
import logging

INSTAGRAM_USERNAME = [redacted]
INSTAGRAM_PASSWORD = [redacted]


def start(user, password):
    client = Client()
    session_file = Path(f"{user}.json")

    try:
        client.load_settings(Path("./user_settings/" + str(session_file)))
    except Exception as e:
        logging.error("Could not load session settings: %s", e)

    try:
        client.login(username=user, password=password)
        logging.info('Logged in successfully')
        client.dump_settings(Path("./user_settings/" + str(session_file)))
    except Exception as e:
        logging.error("Login failed: %s", e)
    return client


def find_user_id(client: Client,username):
    return client.user_id_from_username(username)

def send_direct_from_specific_comment(client: Client, list_of_users: List[AnyStr], ANSWER: AnyStr):
    logging.info('Sending direct comment')
    for user in list_of_users:
        client.direct_send(
            ANSWER,
            user_ids=[int(user)]
        )
        logging.info(f'Answer to user {user} sent...')
```