**Продолжаем писать сайт с курсами**

Обновить app.py, чтоб выводились студенты, преподаватели и курсы

Добавить add\_course.html, где для каждого курса выбирается преподаватель из выпадающего списка.

Курсы также записываются в базу данных

Для тех кто сделал - добавляем страницу авторизации

Чтобы добавить страницу авторизации для пользователей и ограничить доступ к ресурсам (например, к курсам), вы можете использовать библиотеку Flask-Login для управления аутентификацией. Мы создадим следующие компоненты:

1. **Страница регистрации** для создания новых пользователей.
2. **Страница входа** для аутентификации пользователей.
3. Обновим модель пользователя и добавим необходимые маршруты.
4. Ограничим доступ к странице с курсами только для аутентифицированных пользователей.

**Шаг 1: Установите необходимые библиотеки**

Если вы еще не установили Flask-Login, сделайте это:

терминал

pip install Flask-Login

**Шаг 2: Обновите models.py**

Добавьте модель пользователя для хранения данных о пользователе:

from flask\_sqlalchemy import SQLAlchemy

from flask\_login import UserMixin

db = SQLAlchemy()

class Student(db.Model):

id = db.Column(db.Integer, primary\_key=True)

name = db.Column(db.String(100), nullable=False)

class Course(db.Model):

id = db.Column(db.Integer, primary\_key=True)

name = db.Column(db.String(100), nullable=False)

teacher\_id = db.Column(db.Integer, db.ForeignKey('teacher.id'))

teacher = db.relationship('Teacher', backref=db.backref('courses', lazy=True))

class Teacher(db.Model):

id = db.Column(db.Integer, primary\_key=True)

name = db.Column(db.String(100), nullable=False)

class User(UserMixin, db.Model): *# Новая модель User*

id = db.Column(db.Integer, primary\_key=True)

username = db.Column(db.String(100), unique=True, nullable=False)

password = db.Column(db.String(255), nullable=False)

**Шаг 3: Обновите app.py**

Добавьте необходимые маршруты и логику для обработки аутентификации:

from flask import Flask, render\_template, request, redirect, url\_for, flash

from flask\_sqlalchemy import SQLAlchemy

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

from werkzeug.security import generate\_password\_hash, check\_password\_hash

*# Инициализация*

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

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///school.db'

app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False

app.secret\_key = 'your\_secret\_key' *# замените на ваш секретный ключ*

db.init\_app(app)

login\_manager = LoginManager()

login\_manager.init\_app(app)

login\_manager.login\_view = 'login'

@login\_manager.user\_loader

def load\_user(user\_id):

return User.query.get(int(user\_id))

*# Остальная часть кода остается прежней...*

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

def register():

if request.method == 'POST':

username = request.form['username']

password = request.form['password']

hashed\_password = generate\_password\_hash(password)

new\_user = User(username=username, password=hashed\_password)

db.session.add(new\_user)

db.session.commit()

flash('Регистрация успешна! Пожалуйста, войдите.', 'success')

return redirect(url\_for('login'))

return render\_template('register.html')

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

def login():

if request.method == 'POST':

username = request.form['username']

password = request.form['password']

user = User.query.filter\_by(username=username).first()

if user and check\_password\_hash(user.password, password):

login\_user(user)

return redirect(url\_for('index'))

else:

flash('Неправильное имя пользователя или пароль.', 'danger')

return render\_template('login.html')

@app.route('/logout')

@login\_required

def logout():

logout\_user()

return redirect(url\_for('index'))

**Шаг 4: Создайте шаблоны для регистрации и входа**

register.html

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Регистрация</title>

</head>

<body>

<h1>Регистрация</h1>

<form method="post">

<label for="username">Имя пользователя:</label>

<input type="text" id="username" name="username" required>

<label for="password">Пароль:</label>

<input type="password" id="password" name="password" required>

<button type="submit">Зарегистрироваться</button>

</form>

<a href="{{ url\_for('login') }}">Уже есть аккаунт? Вход</a>

</body>

</html>

login.html

html

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Вход</title>

</head>

<body>

<h1>Вход</h1>

<form method="post">

<label for="username">Имя пользователя:</label>

<input type="text" id="username" name="username" required>

<label for="password">Пароль:</label>

<input type="password" id="password" name="password" required>

<button type="submit">Войти</button>

</form>

<a href="{{ url\_for('register') }}">Нет аккаунта? Регистрация</a>

</body>

</html>

**Шаг 5: Ограничьте доступ к страницам**

Теперь нам нужно ограничить доступ к страницам курсов для пользователей, которые не вошли в систему.

Обновите функцию index, чтобы она отображала курсы только для аутентифицированных пользователей:

python

@app.route('/')

def index():

students = Student.query.all()

teachers = Teacher.query.all()

*# Проверяем, вошел ли пользователь в систему*

if current\_user.is\_authenticated:

courses = Course.query.all() *# Получаем курсы, если пользователь аутентифицирован*

else:

courses = [] *# Пустой список курсов для неаутентифицированных пользователей*

return render\_template('index.html', teachers=teachers, students=students, courses=courses)

Настройка доступа к курсам

Если у вас есть отдельная страница для курсов, вы также можете ограничить к ней доступ. Например:

python

@app.route('/courses')

@login\_required

def courses():

courses = Course.query.all()

return render\_template('courses.html', courses=courses)

**Шаг 6: Добавьте ссылки для входа и выхода**

Теперь добавим ссылки для входа и выхода в index.html, чтобы пользователи могли получить доступ к этим функциям.

**Обновите index.html:**

html

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Главная страница</title>

</head>

<body>

<h1>Главная страница</h1>

{% if current\_user.is\_authenticated %}

<p>Добро пожаловать, {{ current\_user.username }}!</p>

<a href="{{ url\_for('logout') }}">Выход</a>

{% else %}

<a href="{{ url\_for('login') }}">Вход</a>

<a href="{{ url\_for('register') }}">Регистрация</a>

{% endif %}

<h2>Студенты</h2>

<ul>

{% for student in students %}

<li>{{ student.name }}</li>

{% endfor %}

</ul>

<h2>Преподаватели</h2>

<ul>

{% for teacher in teachers %}

<li>{{ teacher.name }}</li>

{% endfor %}

</ul>

<h2>Курсы</h2>

<ul>

{% for course in courses %}

<li>{{ course.name }} - Преподаватель: {{ course.teacher.name if course.teacher else 'Не назначен' }}</li>

{% endfor %}

{% if courses | length == 0 %}

<li>Нет доступных курсов.</li>

{% endif %}

</ul>

<h2>Добавить</h2>

<ul>

<li><a href="{{ url\_for('add\_student') }}">Добавить Студента</a></li>

<li><a href="{{ url\_for('add\_teacher') }}">Добавить Преподавателя</a></li>

<li><a href="{{ url\_for('add\_course') }}">Добавить Курс</a></li>

</ul>

</body>

</html>

**Шаг 7: Инициализация базы данных**

Не забудьте инициализировать базу данных, если вы ее еще не инициализировали.

python

*# В основном файле (например, app.py)*

with app.app\_context():

db.create\_all()

Шаг 8: Запустите приложение

Когда вы добавили всю необходимую логику для регистрации и входа пользователей, пора запустить ваше приложение Flask. Вот как это сделать:

1. **Убедитесь, что у вас установлены все необходимые зависимости**:

Убедитесь, что у вас установлен Flask и другие библиотеки, которые используются в проекте. Например:

терминал

pip install Flask Flask-SQLAlchemy Flask-Login

1. **Запустите сервер**:

В вашем терминале перейдите в каталог проекта и запустите приложение командой:

bash

flask run

Убедитесь, что у вас правильно настроена переменная окружения FLASK\_APP. Вы можете установить ее следующим образом (в зависимости от вашей операционной системы):

**Windows**:

bash

set FLASK\_APP=app.py

**Linux и macOS**:

bash

export FLASK\_APP=app.py

1. **Перейдите в браузер**:

Откройте веб-браузер и перейдите по адресу:

text

http://127.0.0.1:5000/

**Шаг 9: Реализация функции выхода**

Чтобы пользователи могли выходить из системы, нужно определить функцию выхода. Добавьте следующий код в ваш app.py для обработки выхода:

python

from flask import redirect, url\_for, flash

from flask\_login import logout\_user

@app.route('/logout')

def logout():

logout\_user() *# Очищает данные о текущем пользователе*

flash('Вы успешно вышли из системы.') *# Сообщение об успешном выходе*

return redirect(url\_for('index')) *# Перенаправляем на главную страницу*

Не забудьте добавить соответствующий импорт для flash:

python

from flask import flash

**Шаг 10: Добавление сообщений об успешных действиях**

Чтобы пользователи видели сообщения об успешных действиях (таких как успешный вход, регистрация и выход), давайте добавим отображение этих сообщений в index.html.

Добавьте следующий код в начало секции <body> вашего

**index.html:**

{% with messages = get\_flashed\_messages() %}

{% if messages %}

<ul class="flashes">

{% for message in messages %}

<li>{{ message }}</li>

{% endfor %}

</ul>

{% endif %}

{% endwith %}

Это позволит отображать сообщения, которые вы передаете с помощью flash().

**Шаг 11: Обновление функций регистрации и входа**

Вот как могут выглядеть функции для регистрации и входа, если вы их еще не реализовали:

Регистрация

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

def register():

if request.method == 'POST':

username = request.form['username']

password = request.form['password']

*# Создаем нового пользователя*

new\_user = User(username=username)

new\_user.set\_password(password) *# Предполагаем, что у вас есть метод для установки пароля*

db.session.add(new\_user)

db.session.commit()

flash('Регистрация прошла успешно! Пожалуйста, войдите в систему.')

return redirect(url\_for('login'))

return render\_template('register.html')

**Вход**

python

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

def login():

if request.method == 'POST':

username = request.form['username']

password = request.form['password']

user = User.query.filter\_by(username=username).first()

if user and user.check\_password(password): *# Предполагаем, что у вас есть метод проверки пароля*

login\_user(user) *# Вход пользователя*

flash('Вы вошли в систему.')

return redirect(url\_for('index'))

else:

flash('Неверное имя пользователя или пароль.')

return render\_template('login.html')