### Инструкция для разработчика по веб-приложению управления заявками на ремонт автомобилей

#### Введение

Это веб-приложение предназначено для управления заявками на ремонт автомобилей в автосервисе «АвтоТранс». В данном руководстве описаны шаги по настройке, запуску и разработке приложения.

### Начало работы

#### Установка и настройка

1. \*\*Клонирование репозитория\*\*

- Склонируйте репозиторий с приложением на локальную машину:

```sh

git clone <URL-репозитория>

```

2. \*\*Создание виртуального окружения\*\*

- Перейдите в директорию проекта:

```sh

cd <имя-проекта>

```

- Создайте виртуальное окружение:

```sh

python -m venv venv

```

- Активируйте виртуальное окружение:

- Для Windows:

```sh

venv\Scripts\activate

```

- Для macOS/Linux:

```sh

source venv/bin/activate

```

3. \*\*Установка зависимостей\*\*

- Установите необходимые зависимости из файла `requirements.txt`:

```sh

pip install -r requirements.txt

```

4. \*\*Настройка базы данных\*\*

- Настройте базу данных SQLite или другую СУБД по вашему выбору.

- Создайте файл базы данных:

```sh

flask db init

flask db migrate -m "Initial migration."

flask db upgrade

```

5. \*\*Настройка переменных окружения\*\*

- Создайте файл `.env` и укажите в нем переменные окружения:

```

FLASK\_APP=app.py

FLASK\_ENV=development

FLASK\_DEBUG=1

```

#### Структура проекта

- `app.py`: Основной файл приложения Flask.

- `templates/`: Директория с HTML-шаблонами.

- `static/`: Директория с файлами CSS, JS и изображениями.

- `models.py`: Модели базы данных.

- `forms.py`: Формы для ввода данных.

- `routes.py`: Маршруты и обработчики запросов.

- `database.db`: Файл базы данных SQLite (если используется SQLite).

### Основные функции

#### Добавление новой заявки

- Маршрут: `/add`

- Метод: GET, POST

- Описание: Обработчик для добавления новой заявки. Использует форму `AddRequestForm`.

#### Редактирование заявки

- Маршрут: `/edit/<int:request\_id>`

- Метод: GET, POST

- Описание: Обработчик для редактирования существующей заявки. Использует форму `EditRequestForm`.

#### Просмотр заявки

- Маршрут: `/view/<int:request\_id>`

- Метод: GET

- Описание: Обработчик для просмотра подробной информации о заявке.

#### Удаление заявки

- Маршрут: `/delete/<int:request\_id>`

- Метод: POST

- Описание: Обработчик для удаления заявки.

#### Загрузка данных из Excel

- Маршрут: `/upload`

- Метод: GET, POST

- Описание: Обработчик для загрузки и импорта заявок из файла Excel. Использует форму `UploadForm`.

### Пример кода для маршрутов

```python

# app.py

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

from flask\_sqlalchemy import SQLAlchemy

from forms import AddRequestForm, EditRequestForm, UploadForm

from models import db, Request, Mechanic

import pandas as pd

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

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

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

db.init\_app(app)

@app.route('/')

def index():

requests = Request.query.all()

return render\_template('index.html', requests=requests)

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

def add\_request():

form = AddRequestForm()

if form.validate\_on\_submit():

new\_request = Request(

request\_number=form.request\_number.data,

date\_added=form.date\_added.data,

car\_type=form.car\_type.data,

car\_model=form.car\_model.data,

problem\_description=form.problem\_description.data,

client\_name=form.client\_name.data,

phone\_number=form.phone\_number.data,

status=form.status.data,

assigned\_mechanic=form.assigned\_mechanic.data

)

db.session.add(new\_request)

db.session.commit()

flash('Request added successfully!', 'success')

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

return render\_template('add\_request.html', form=form)

@app.route('/edit/<int:request\_id>', methods=['GET', 'POST'])

def edit\_request(request\_id):

request\_obj = Request.query.get\_or\_404(request\_id)

form = EditRequestForm(obj=request\_obj)

if form.validate\_on\_submit():

form.populate\_obj(request\_obj)

db.session.commit()

flash('Request updated successfully!', 'success')

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

return render\_template('edit\_request.html', form=form)

@app.route('/view/<int:request\_id>')

def view\_request(request\_id):

request\_obj = Request.query.get\_or\_404(request\_id)

return render\_template('view\_request.html', request=request\_obj)

@app.route('/delete/<int:request\_id>', methods=['POST'])

def delete\_request(request\_id):

request\_obj = Request.query.get\_or\_404(request\_id)

db.session.delete(request\_obj)

db.session.commit()

flash('Request deleted successfully!', 'success')

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

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

def upload():

form = UploadForm()

if form.validate\_on\_submit():

file = form.file.data

df = pd.read\_excel(file)

for \_, row in df.iterrows():

new\_request = Request(

request\_number=row['request\_number'],

date\_added=row['date\_added'],

car\_type=row['car\_type'],

car\_model=row['car\_model'],

problem\_description=row['problem\_description'],

client\_name=row['client\_name'],

phone\_number=row['phone\_number'],

status=row['status'],

assigned\_mechanic=row['assigned\_mechanic']

)

db.session.add(new\_request)

db.session.commit()

flash('File uploaded and requests added successfully!', 'success')

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

return render\_template('upload.html', form=form)

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

app.run(debug=True)

```

### Модели базы данных

```python

# models.py

from flask\_sqlalchemy import SQLAlchemy

db = SQLAlchemy()

class Request(db.Model):

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

request\_number = db.Column(db.String(100), nullable=False)

date\_added = db.Column(db.Date, nullable=False)

car\_type = db.Column(db.String(100), nullable=False)

car\_model = db.Column(db.String(100), nullable=False)

problem\_description = db.Column(db.Text, nullable=False)

client\_name = db.Column(db.String(100), nullable=False)

phone\_number = db.Column(db.String(15), nullable=False)

status = db.Column(db.String(50), nullable=False)

assigned\_mechanic = db.Column(db.Integer, db.ForeignKey('mechanic.id'))

class Mechanic(db.Model):

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

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

phone = db.Column(db.String(15), nullable=False)

requests = db.relationship('Request', backref='mechanic', lazy=True)

```

### Формы

```python

# forms.py

from flask\_wtf import FlaskForm

from wtforms import StringField, SubmitField, DateField, SelectField, TextAreaField, FileField

from wtforms.validators import DataRequired

from models import Mechanic

class AddRequestForm(FlaskForm):

request\_number = StringField('Номер заявки', validators=[DataRequired()])

date\_added = DateField('Дата добавления', validators=[DataRequired()])

car\_type = StringField('Тип авто', validators=[DataRequired()])

car\_model = StringField('Модель авто', validators=[DataRequired()])

problem\_description = TextAreaField('Описание проблемы', validators=[DataRequired()])

client\_name = StringField('ФИО клиента', validators=[DataRequired()])

phone\_number = StringField('Номер телефона', validators=[DataRequired()])

status = SelectField('Статус заявки', choices=[('new', 'Новая'), ('in\_progress', 'В процессе ремонта'), ('completed', 'Завершена')], validators=[DataRequired()])

assigned\_mechanic = SelectField('Ответственный механик', coerce=int, choices=[(m.id, m.name) for m in Mechanic.query.all()])

submit = SubmitField('Добавить заявку')

class EditRequestForm(AddRequestForm):

submit = SubmitField('Сохранить изменения')

class UploadForm(FlaskForm):

file = FileField('Файл Excel', validators=[DataRequired()])

submit = SubmitField('Загрузить')

```

### Заключение

Данное руководство поможет вам развернуть и настроить веб-приложение для управления заявками на ремонт автомобилей, а также внести необходимые изменения и улучшения. Следуйте инструкциям и используйте предоставленный код для разработки и отладки приложения.