SteelRoar serviss

Problēmas izpēte un analīze

Problēmas apraksts:

Nesen atvērtais autoserviss "SteelRoar" saskārās ar problēmu. Tā kā autoservisā strādā tikai divi cilvēki un abi nodarbojas ar remontdarbiem, pieaugošās servisa popularitātes dēļ kļuva diezgan neērti saņemt zvanus pa tālruni, lai pierakstītos uz remontu. Man ir jāatrod risinājums, kas atvieglotu mehāniķu darbu un novērstu nepieciešamību visu laiku turēt tālruni pie sevis. Labākais risinājums šai problēmai būtu izveidot tīmekļa lapu, kurā lietotājs var atstāt savu remonta pieprasījumu, norādot konkrētu nedēļas dienu, kā arī uzzināt aptuveno remonta cenu.

# Programmatūras prasību specifikācija

Projekta mērķis:

Izveidot vienotu pieraksta sistēmu autoservisam "SteelRoar", kas ļauj:

* Autoservisa darbiniekiem,izmantojot lietotājvārdu un paroli pārbaudit remonta ierakstus
* Klienti var iesniegt pieteikumu tiešsaistē un iepriekš uzzināt aptuvenās pakalpojuma izmaksas.
* Darbinieki var veikt savu darbu, ko netraucē nepārtraukti zvani.
* sistēmas administratoram ir iespēja pievienot jaunus sistēmas lietotājus.
* sistēmas administratoram ir iespēja pievienot jaunus pakalpojumus.
* Darbiniekiem saņēmt pieteikumu uz darba e-pastu.

Detalizētas prasības:

* Datubaze lai saglabatu visu pieteikuma informaciju
* Datubaze lai saglobatu pieslegšanas datus
* Parole aizsardziba izmantojot algoritmu sha256
* Pieteikuma saraksts kuru var atvert tikai darbinieki
* Instiktivi saprotamais dizains
* Pieteikumu atsuta uz darba e-pastu

Mērķauditorija:

Galvenā tīmekļa vietnes mērķauditorija, visticamāk, būs cilvēki vecumā no 30 līdz 50 gadiem, jo jaunāki cilvēki ir vairāk gatavi veikt nelielu remontu ar savām rokām. Sakarā ar lietotāju vecuma īpašībām ir jārūpējas par to, lai lapa būtu viegli lietojama, ar minimālu pogu skaitu, lai lietotājs netiktu dezorientēts.

Skice: Изображение выглядит как текст, диаграмма, линия, снимок экрана

Автоматически созданное описание

# Programmatūras izstrādes plāns

Es esmu izvēlējies Iteratīvo modeli. Pirmais, kas tika izdarīts šajā darbā, bija sakumlapas izveide, izmantojot html, pievienojot nelielu veidlapu tālruņa numuram, vārdam uzvārdam, izvēloties datumu un remonta veidu. Pēc tam tika pievienota poga, kas aizved uz paldies lapu un arī ieraksta datus teksta dokumentā. Pēc tam es nomainīju manuālo remonta datuma ierakstīšanu ar interaktīvu kalendāru, ko īstenoju, izmantojot JavaScript. Tā kā es jau izmantoju JavaScript, nolēmu, ka mana darba ietvars būs Jquery, un pēc nelielas testešanas un kļūdiem, nomainīju Jquery uz saprotamāku un labāk optimizētu Flask, ar kuru man jau bija pieredze. JavaScript es atstāju, lai īstenotu aptuvenās remonta cenas izvadīšanu, kad lietotājs izvēlējās remonta veidu. Pēc tam tika nolemts papildus pievienot vēstules nosūtīšanu uz e-pastu, jo šādā veidā darbiniekiem nebūs katru reizi jāiet uz vietni, lai pārbaudītu, vai ir jauns ieraksts. Pēc šīs idejas īstenošanas nonācu pie secinājuma, ka datu glabāšana teksta dokumentā pārāk noslogos vietni un, ja datplūsma un vienlaicīgu pieprasījumu apjoms būs liels, dokuments neizturēs un izmetīs kļūdas, tāpēc teksta dokuments tika nomainīts uz SQL datubāzi. Pēc tam sekoja atsevišķas lapas izveide, lai pārbaudītu ierakstus. Sākumā ideja bija atstāt paroli kodā, lai tā tika salīdzina to ar to, ko raksta lietotājs, bet tad, kā izrādījās, ka tas nav pārāk droši, tāpēc es izstrādāju nelielu kodu citā Python failā, lai esošajai lietotāju datubāzei pievienotu atsevišķu tabulu ar lietotāju, kurā būtu viņa lietotājvārds un parole, protams, tas nav drošākais datu glabāšanas veids, bet tas ir labāk nekā atstāt paroli kodā, tāpēc, pievienojot paroli, es to hasheju? ar sha256 algoritmu. Pēc konta pieteikšanās lapas un atsevišķas lapas izveides, kurā var pārbaudīt ierakstus, palika pabeigt projekta dizainu. Šī metodoloģija tika izmantota tāpēc, ka tā man vislabāk piestāv, jo bieži vien es varu kaut ko nepabeigt un pāriet pie nākamā vai pēkšņi izlemt, ka kaut kas man neder, un mainīt to pēc izstrādes puses no darba. Tāpat kā tas notika ar Jquery, ko es ilgu laiku izmantoju un pēc tam nomainīju uz flask. Protams, šai metodoloģijai ir viens trūkums, proti, viss ir jātestē 20 reižu, jo es bieži salauzu lietas, kas darbojās labi.

# Atkļūdošanas un akcepttestēšanas pārskats

Testpiemeri:

SRM-D1(K)-T1

Kur SRM- SteelRoar majaslapa

D1-Darbinieks 1

K-Klients

T1- Tests numurs 1

SRM-K-T1

Scenārijs: Klients apmeklē lapu un pieraksta vizīti uz 27. aprīli, lai veiktu dzinēja remontu.

Sasniedzamais rezultāts: Klients tiek novirzīts uz pateicības lapu, un uz darba e-pastu tiek nosūtīta vēstule, un dati tiek reģistrēti datubāzē. 

SRM-D1-T2

Scenārijs: Darbinieks atver majaslapu, lai pārbaudītu ierakstus.

Sasniedzamais rezultāts: Darbinieks redz visus ierakstus, kuri ir sakārtoti pēc datuma.

SRM-K-T3

Scenārijs: Klients vēlas uzzināt, cik aptuveni maksā automašīnas riepu maiņa.

Sasniedzamais rezultāts: Pēc remonta veida izvēles tiek parādīta aptuvenā cena. Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Sarežģītākā darba daļa kļūdu ziņā bija e-pasta vēstuļu sūtīšanas iestatīšana, jo to uzstādīšana un lietošana bija ļoti sarežģīta. Personīgi es šim nolūkam izmantoju google pakalpojumu, un tā bija viena no lielākajām kļūdām. Pēc tam, kad es sapratu, kā tas viss darbojas, es sapratu, ka to nav grūti pārkonfigurēt, bet tas vairs nav nepieciešams, jo es izmantoju Google, man bija jāizveido lietojumprogrammas parole, īpaša parole, ar kuru kods iegūst piekļuvi e-pastam, ar parastu paroli būs tikai kļūda. Sakarā ar to, ka izmanto Flask, bija arī problēma ar maršrutizāciju, jo pieprasījuma nosūtīšanas poga pēkšņi sāka sūtīt tieši uz veiktajiem ierakstiem, neprasot paroli, kas reālajā dzīvē varētu pārkāpt pāris konfidencialitātes likumus.

# Lietotāja ceļvedis

Mērķauditorija: Cilvēki vecumā no 30 gadiem, kuri vēlas pierakstīties automašīnas remontam un padarīt to pēc iespējas vienkāršāk.

Mērķis: Galvenais mērķis ir atvieglot cilvēkiem reģistrēšanos remontdarbiem un atvieglot autoservisā darbiniekiem, lai viņiem nebūtu vienmēr jāstaigā ar telefonu.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Sākuma lapā redzami 4 ievades laukumi. Pirmkārt, tālruņa numura lauks - šeit jāievada tālruņa numuru, lai galvenais mehāniķis varētu ar jums sazināties ārkārtas gadījumā.

Otrais lauks, šeit jāievada datumu, kad vēlaties ierasties uz remontu.

Trešais, vārds un uzvārds, šis lauks ir nepieciešams, lai darbinieki zinātu, kā jūs adresēt.

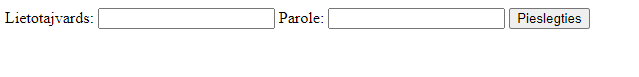
Ceturtkārt, remonta veids, šeit norādiet, ko tieši jums ir nepieciešams remontēt.

Tālāk ir divas pogas, pirmā ir atbildīga par datu nosūtīšanu (neaizmirstiet, noklikšķinot uz šīs pogas, jūs piekrītat, ka mūsu darbinieki saņems jūsu personīgo informāciju), pēc tās nospiešanas informācija par jūsu remontu tiks nosūtīta mūsu darbiniekiem.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Nospiežot uz pogas iesniegt, tiek atvērta šāda lapa, kurā jūs varat nospiest pogu un atgriezties galvenajā lapā.



Nospiežot otro pogu, tiks atvērta pieteikšanās lapa, kurā varat pieslēgties savam kontam, ja esat darbinieks, un pārbaudīt ierakstus.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Ievadot konta informāciju, ko jums ir piešķīris administrators, un nospiežot uz pogas pieteikties, redzēsiet visus ierakstus par remontu, kā arī pogu, kas jūs aizvedīs atpakaļ uz sākumlapu.

# Piemērotās licences pamatojums

Manai programmatūrai būs komerciāla licence.

Standarta komerciālā licence šāda veida programmatūrai. Lietošanas noteikumi būs šādi: pircējam ir tiesības veikt izmaiņas, papildināt programmu un arī izmantot to, lai pelnītu naudu, bet programmas cena nebūs fiksēta, bet gan procentuālā daļa no šīs programmas peļņas. Teiksim, ka vietne 4 mēnešu laikā ir piesaistījusi 173 klientus un vidējais čeks bija 250 eiro, no katra klienta viņiem būs jāmaksā man 5 procenti. Tātad tas ir 2162 eiro. Bet tas nenozīmē, ka viņi maksās visu mūžu un neko nesaņems, tiks noslēgts līgums ar visiem nosacījumiem un no manas puses man būs pienākums avāriju, servera sabrukumu gadījumā atrisināt problēmu un arī veikt labojumus, ja nepieciešams.

# Programmatūras kods

Saksim ar HTML templates

# templates/index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<link rel="stylesheet" type="text/css" href="static/style.css">

<title>Remonts</title>

</head>

<body>

<form action="/submit-form" method="post">

<label for="phone">Telefona numurs:</label>

<input type="text" id="phone" name="phone">

<br>

<label for="repairDate">Pieteikuma datums:</label>

<input type="date" id="repairDate" name="repairDate">

<br>

<label for="fullName">Vards Uzvards:</label>

<input type="text" id="fullName" name="fullName">

<br>

<label for="repairType">Remonta veids:</label>

<select id="repairType" name="repairType">

<option value="nothing">-------</option>

<option value="Dzinējs">Dzinējs</option>

<option value="Transmisija">Transmisija</option>

<option value="Balstiekārta">Balstiekārta</option>

<option value="Riepuapkope">Riepu apkope</option>

</select>

<br>

<div id="priceEstimate">Aptuvenā remonta cena: <span id="price"></span></div>

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

<button type="button" onclick="window.location.href='/login'">Pieslegties</button>

</form>

<script src="/static/script.js"></script>

</body>

</html>

# templates/login.html

<form method="post" action="{{ url\_for('login') }}">

<label for="username">Lietotajvards:</label>

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

<label for="password">Parole:</label>

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

<input type="submit" value="Pieslegties">

</form>

# templates/paldies.html

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<!--ierakstiju css kodu šeit jo tur jau tik daudz uzrakstis ko es tur jau gandriz neko nesaprotu-->

<title>Paldies par pieteikumu!</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 20px;

text-align: center;

}

.thank-you-note {

margin-top: 50px;

}

.back-button {

margin-top: 30px;

padding: 10px 20px;

font-size: 16px;

color: #fff;

background-color: #5c6bc0;

border: none;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s ease;

}

.back-button:hover {

background-color: #3f51b5;

}

</style>

</head>

<body>

<div class="thank-you-note">

<h1>Paldies, ka atstājāt pieprasījumu!</h1>

<p>Jūsu pieteikums ir veiksmīgi apstrādāts. Mūsu galvenais mehāniķis drīz ar jums sazināsies, lai precizētu informāciju.</p>

<button onclick="goBack()" class="back-button">Atgriezties uz sākumlapu</button>

</div>

<script>

function goBack() {

window.location.href = '/';

}

</script>

</body>

</html>

# templates/appointments.html

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<link rel="stylesheet" type="text/css" href="static/style.css">

<title>Pieteikumu saraksts</title>

</head>

<body>

<div class="container">

<h1>Pieteikumu saraksts</h1>

<!--jinja2 sintaksis ļoti interesanta lieta , ka es saprotu var izmantot tikai ar flask kopa, priekš dinamiskam mainigajiem ļoti laba lieta-->

{% for request in appointments %}

<div class="request">

<p>{{ request.repair\_date }} - {{ request.full\_name }} - {{ request.repair\_type }} - {{ request.phone }}</p>

</div>

{% endfor %}

<button onclick="window.location.href='/'">Atgriezties uz sakumlapu</button>

</div>

</body>

</html>

# Python kods main.py

Viss lielakais kods mana darba

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

from flask\_sqlalchemy import SQLAlchemy

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

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

import smtplib

from email.mime.text import MIMEText

from email.mime.multipart import MIMEMultipart

app = Flask(\_\_name\_\_, static\_folder='static')

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

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

app.config['SECRET\_KEY'] = 'q2we4t6'

db = SQLAlchemy(app)

#nezinu kapec kļuda

login\_manager = LoginManager()

login\_manager.init\_app(app)

login\_manager.login\_view = 'login'

class User(UserMixin, db.Model):

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

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

password = db.Column(db.String(200))

class Request(db.Model):

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

phone = db.Column(db.String(50))

repair\_date = db.Column(db.String(50))

full\_name = db.Column(db.String(100))

repair\_type = db.Column(db.String(100))

@login\_manager.user\_loader

def load\_user(user\_id):

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

#tabulas veidošana

with app.app\_context():

db.create\_all()

@app.route('/appointments')

@login\_required

def appointments():

all\_requests = Request.query.order\_by(Request.repair\_date).all()

return render\_template('appointments.html', appointments=all\_requests)

@app.route('/')

def home():

return render\_template('index.html')

@app.route('/submit-form', methods=['POST'])

def submit\_form():

phone = request.form['phone']

repair\_date = request.form['repairDate']

full\_name = request.form['fullName']

repair\_type = request.form['repairType']

new\_request = Request()

new\_request.phone = phone

new\_request.repair\_date = repair\_date

new\_request.full\_name = full\_name

new\_request.repair\_type = repair\_type

db.session.add(new\_request)

db.session.commit()

all\_requests = Request.query.order\_by(Request.repair\_date).all()

# atsuta emailu uz darba epastu

msg = MIMEMultipart()

msg['From'] = 'steelroar12@gmail.com'

msg['To'] = 'zaharij148@gmail.com'

msg['Subject'] = 'Jauns remonts'

body = f"Telefona numurs: {phone}, Datums: {repair\_date}, VardsUzvards: {full\_name}, Remonta veids: {repair\_type}"

msg.attach(MIMEText(body, 'plain'))

smtp\_server = 'smtp.gmail.com'

smtp\_port = 587

smtp\_username = 'steelroar12@gmail.com'

smtp\_password = 'šeit vajadzetu but parolei'

server = smtplib.SMTP(smtp\_server, smtp\_port)

server.starttls()

try:

server.login(smtp\_username, smtp\_password)

server.sendmail(msg['From'], msg['To'], msg.as\_string())

except smtplib.SMTPAuthenticationError as e:

print(f"Autorizešanas kļuda: {e.smtp\_code}, {e.smtp\_error}")

except smtplib.SMTPException as e:

print(f"SMTP Kļuda: {e}")

finally:

server.quit()

return render\_template('paldies.html')

#pieslegšana lai apskatītu pieteikumus

@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, remember=True)

next\_page = request.args.get('next')

return redirect(next\_page or url\_for('appointments'))

return render\_template('login.html')

@app.route('/logout')

@login\_required

def logout():

logout\_user()

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

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

app.run(host='0.0.0.0', port=8080)

Es speciāli izdzēsu paroli, jo to var izmantot, lai piekļūtu manam e-pastam.

# add\_user\_app.py

import sys

from main import app, db, User

from werkzeug.security import generate\_password\_hash

#šeit ari nesaprotu kapec ir kļuda

def add\_user(username, password):

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

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

db.session.add(new\_user)

db.session.commit()

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

if len(sys.argv) == 3:

username = sys.argv[1]

password = sys.argv[2]

with app.app\_context():

add\_user(username, password)

else:

print("Usage: python add\_user\_app.py <username> <password>")

# static/style.css

form div {

margin-bottom: 10px;

}

label {

margin-right: 10px;

}

input[type=text], select {

padding: 5px;

margin-right: 10px;

}

button {

padding: 5px 15px;

cursor: pointer;

}

body, html {

height: 100%;

margin: 0;

padding: 0;

font-family: Arial, sans-serif;

display: flex;

justify-content: center;

align-items: center;

background-color: #f7f7f7;

background-image: url('/static/123.webp');

background-size: cover;

background-position: center;

background-repeat: no-repeat;

}

form {

background-color: #fff;

padding: 20px;

border-radius: 5px;

box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);

width: 100%;

max-width: 400px;

}

input[type="text"],

select,

button {

width: 100%;

padding: 10px;

margin: 10px 0;

border: 1px solid #ddd;

border-radius: 5px;

box-sizing: border-box;

}

button {

background-color: #5c6bc0;

color: white;

border: none;

cursor: pointer;

transition: background-color 0.3s ease;

}

button:hover {

background-color: #3f51b5;

}

h1 {

text-align: center;

color: #333;

}

#priceEstimate {

text-align: center;

font-size: 1.1em;

margin-top: 20px;

}

.request {

background: #e3e3e3;

padding: 10px;

margin-bottom: 10px;

border-radius: 3px;

}

.container {

max-width: 600px;

margin: auto;

background: #fff;

padding: 20px;

border-radius: 5px;

box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

}

.form-group {

margin-bottom: 15px;

}

# static/script.js

document.addEventListener('DOMContentLoaded', function () {

console.log('DOM fully loaded and parsed');

var repairTypeSelect = document.getElementById('repairType');

var priceDisplay = document.getElementById('price');

console.log(repairTypeSelect, priceDisplay);

var repairPrices = {

Dzinējs: '300 Euro',

Transmisija: '200 Euro',

Balstiekārta: '100 euro',

Riepuapkope: "70 euro",

};

repairTypeSelect.addEventListener('change', function () {

console.log('Change event triggered');

var selectedOption = this.value;

console.log('Selected option:', selectedOption);

var price = repairPrices[selectedOption];

console.log('Price:', price);

priceDisplay.textContent = price || 'Izveleties remonta veidu';

});

repairTypeSelect.dispatchEvent(new Event('change'));

});

Interesants fakts, lai izmantotu kādu citu kodu kopā ar Flask, ir nepieciešams, lai šis kods obligāti būtu static mapē citādi viņš mēģinās to atrast šajā mapē un tā arī nevarēs atskaņot.

Paldies par uzmanību.