**RĪGAS VALSTS TEHNIKUMS**

DATORIKAS NODAĻA

Izglītības programma: Programmēšana

**KVALIFIKĀCIJAS DARBS**

**“Tīmekļa lietotne automašīnu meklēšanai un pārdošanai tiešsaistē”**

Paskaidrojošais raksts 107 lpp.

Audzēknis: Konstantīns Kotovičs

Prakses vadītājs: Igors Litvjakovs

Nodaļas vadītājs: Normunds Barbāns

**Rīga 2024**

**ANNOTĀCIJA**

Kvalifikācijas darba tēma ir “Tīmekļa lietotne automašīnu meklēšanai un pārdošanai tiešsaistē”. Šī projekta pamatmērķis ir radīt lietotājam draudzīgu tīmekļa vietni, kura palīdzēs atrast un iegādāties automašīnas tiešsaistē, nodrošinot efektīvas meklēšanas un filtrēšanas iespējas. Mūsdienās, kad automašīnu izvēle ir sarežģīts process, ir svarīgi, lai lietotāji varētu veikt šo procesu vienkāršā veidā, izmantojot saprotamas un intuitīvas lietojumprogrammas saskarni. Šī vietne nodrošinās plašu informāciju par automašīnu modeļiem un cenām, un tās filtrēšanas funkcijas ļaus lietotājiem atrast precīzi to, ko viņi meklē. Projekta izstrādei tika izmantoti HTML, CSS, JavaScript, Bootstrap, SweetAlert, PHP, Laragon, Visual Studio Code un GIT. Šie rīki nodrošina efektīvu un modernu lietotnes izstrādi, kas spēj nodrošināt ātru un uzticamu datu apstrādi, kā arī estētiski pievilcīgu un lietotājam draudzīgu saskarni.

Kvalifikācijas darba raksts aplūko izstrādātās programmatūras vairākas svarīgas sadaļas, ieskaitot ievadu, uzdevuma nostādni, prasību specifikāciju, uzdevuma risināšanas līdzekļu izvēles pamatojumu, programmatūras produkta modelēšanu un projektēšanu, datu struktūru apskatu, lietotāja ceļvedi, nobeigumu un pielikumus. Ievadā tiek aprakstīta projekta nepieciešamība un nozīme, uzsvērtas vietnes aktuālitāte un priekšrocības. Uzdevuma nostādnē ir noteikts kvalifikācijas darba mērķis un darbības, kuras sistēmai ir jānodrošina. Prasību specifikācijā sastāv no funkcionālajām un nefunkcionālajām prasībām, kā arī no detalizēta ieejas un izejas informācijas apraksta. Uzdevuma risināšanas līdzekļu izvēles pamatojumā tiek pamatota izvēle konkrētām programmēšanas valodām, rīkiem un tehnoloģijām, kā arī to versijām, kas izmantotas projekta realizācijā. Programmatūras modelēšanas un projektēšanas sadaļā ir apskatīti sistēmas strukturālie un funkcionālie modeļi: datu plūsmas un ER diagrammas, kas attēlo sistēmas arhitektūru un darbības principus. Datu struktūru apraksts ietver detalizētu informāciju par visām tabulu struktūrām un to savstarpējām saistībām, ka arī datu bāzes fizisko struktūras shēmu. Lietotāja ceļvedī ir sniegtas nepieciešamās sistēmas prasības un norādījumi par sistēmas uzstādīšanu un lietošanu, iekļaujot programmas funkcionalitātes aprakstu. Testa piemērs demonstrē reālas situācijas un paredzamos rezultātus, pārbaudot sistēmas darbību dažādos apstākļos. Nobeigumā ir apkopota informācija par paveikto darbu.

Kopumā kvalifikācijas darba apjoms ir 107 lappuses, kurā ietilpst 63 attēli, 13 tabulas un 2 pielikumi. Pielikumi satur ER diagrammu un programmas pirmkodu.

**ANNOTATION**

The topic of the qualification work is "Web Application for Searching and Selling Cars Online." The main objective of this project is to create a user-friendly website that will assist in finding and purchasing cars online by providing effective search and filtering options. In today's age, when choosing a car can be a complex process, it is important for users to be able to perform this process easily using a clear and intuitive user interface. This website will provide extensive information about car models and prices, and its filtering functions will allow users to find exactly what they are looking for. The development of the project utilized HTML, CSS, JavaScript, Bootstrap, SweetAlert, PHP, Laragon, Visual Studio Code, and GIT. These tools ensure effective and modern application development, capable of providing fast and reliable data processing, as well as an aesthetically appealing and user-friendly interface.

The qualification paper examines several important sections of the developed software, including introduction, task statement, requirements specification, justification of tool selection for problem solving, software product modeling and design, data structure overview, user guide, conclusion, and appendices. The introduction describes the necessity and significance of the project, emphasizing the relevance and advantages of the website. The task statement defines the objective of the qualification paper and the actions that the system must provide. The requirements specification consists of functional and non-functional requirements, as well as a detailed description of input and output information. The justification of tool selection for problem solving explains the choice of specific programming languages, tools, and technologies, as well as their versions used in the project implementation. The section on software modeling and design examines the structural and functional models of the system: data flow and ER diagrams illustrating the system architecture and operation principles. The data structure overview includes detailed information about all table structures and their relationships, as well as the physical structure schema of the database. The user guide provides necessary system requirements and instructions for system installation and usage, including a description of program functionality. The test case demonstrates real-life situations and expected outcomes by testing the system's operation under various conditions. In conclusion, information about the work completed is summarized.

In total, the qualification paper comprises 107 pages, including 63 images, 13 tables, and 2 attachments. The attachments contain the ER diagram and the program source code.

**SATURS**

[**IEVADS** 5](#_Toc168959775)

[**1.** **UZDEVUMA NOSTĀDNE** 6](#_Toc168959776)

[**2.** **PRASĪBU SPECIFIKĀCIJA** 8](#_Toc168959777)

[**2.1.** **Ieejas un izejas informācijas apraksts** 8](#_Toc168959778)

[*2.1.1.* *Ieejas informācijas apraksts* 8](#_Toc168959779)

[*2.1.2.* *Izejas informācijas apraksts* 11](#_Toc168959780)

[**2.2.** **Funkcionālās prasības** 11](#_Toc168959781)

[**2.3.** **Nefunkcionālās prasības** 14](#_Toc168959782)

[**3.** **UZDEVUMA RISINĀŠANAS LĪDZEKĻU IZVĒLES PAMATOJUMS** 17](#_Toc168959783)

[**4.** **PROGRAMMATŪRAS PRODUKTA MODELĒŠANA UN PROJEKTĒŠANA** 19](#_Toc168959784)

[**4.1.** **Sistēmas struktūras modelis** 19](#_Toc168959785)

[*4.1.1.* *Sistēmas arhitektūra* 19](#_Toc168959786)

[*4.1.2.* *Sistēmas ER modelis* 20](#_Toc168959787)

[**4.2.** **Funkcionālais sistēmas modelis** 22](#_Toc168959788)

[*4.2.1.* *Datu plūsmu modelis* 22](#_Toc168959789)

[**5.** **DATU STRUKTŪRU APRAKSTS** 29](#_Toc168959790)

[**6.** **LIETOTĀJA CEĻVEDIS** 35](#_Toc168959791)

[**6.1.** **Sistēmas prasības** 35](#_Toc168959792)

[**6.2.** **Sistēmas instalācija un palaišana** 36](#_Toc168959793)

[**6.3.** **Programmas apraksts** 38](#_Toc168959794)

[**6.4.** **Testa piemērs** 59](#_Toc168959795)

[**NOBEIGUMS** 62](#_Toc168959796)

[**INFORMĀCIJAS AVOTI** 63](#_Toc168959797)

[**PIELIKUMI** 64](#_Toc168959798)

[1. pielikums. Entity relationship modelis 65](#_Toc168959799)

[2. pielikums. Programmas pirmkods 66](#_Toc168959801)

# **IEVADS**

Mūsdienās ir ļoti svarīgi izmantot tīmekļa vietotnes, kuras var vienkāršot un paātrināt atlases procesu. Šajā projektā tiek izstrādāta vietne, kas palīdzēs lietotājiem izvēlēties atbilstošāko automašīnu, izmantojot efektīvas meklēšanas un filtrēšanas iespējas, jo automašīnas izvēle daudziem cilvēkiem var būt ļoti grūts un mulsinošs process. Izpētot daudzus automašīnu tirgotājus un mājaslapas, var secināt, ka nevienā no tām nav ērtu un saprotamu navigācijas paneļu un filtrēšanas iespēju. Lietotājus bieži mulsina piedāvājumu un iespēju lielums.

Šī vietne nebūs tikai datu avots par automašīnām, bet arī būs rīks, kas palīdzēs lietotājiem veikt informētus lēmumus par savu nākamo transportlīdzekli. Tā piedāvās detalizētu informāciju par katru transportlīdzekļi un cenām. Pateicoties filtrēšanas iespējām, lietotāji varēs viegli un ātri izvēlēties savām prasībām atbilstošu transportlīdzekli.

Šī projekta aktualitāte būs neapšaubāma, jo cilvēku interese par automašīnu iegādi turpina strauji pieaugt. Arvien vairāk cilvēku sāk izprot, cik svarīgi ir atrast automašīnu, kas pilnībā atbilst viņu vajadzībām un vēlmēm. Šajā dinamiskajā automašīnu tirgū, kur katru dienu parādās jauni modeļi un tehnoloģijas, ir nepieciešams efektīvs instruments, kas palīdzēs lietotājiem veikt gudrus un informētus lēmumus.

Tirgus izpētes rezultāti liecina, ka ir pieprasījums pēc vietnes, kas palīdzētu lietotājiem izvēlēties atbilstošāko automašīnu. Lietotāji vēlas vietni, kas būtu viegli lietojama, sniegtu visaptverošu informāciju un būtu objektīva. Konkurentu analīze liecina, ka pastāvošās vietnes piedāvā dažādas funkcijas un priekšrocības. Dažas vietnes koncentrējas uz cenu salīdzinājumu, citas - uz specifikācijām un funkcijām, bet vēl citas - uz atsauksmēm.

Šī projekta mērķa auditorija ir ļoti plaša un daudzveidīga. To veido gan pieredzējuši automašīnu entuziasti, kuri meklē jaunākos un inovatīvākos modeļus, gan arī tie, kuri tikai sāk iepazīties ar automašīnu pasaules piedāvājumu. Lietotājiem būs pieejama visaptveroša informācija par dažādām automašīnām, nevis tikai modeļiem ar augstām cenām vai jaunākajām tehnoloģijām. Projekts ir paredzēts arī tiem, kuri vēlas atrast ekonomiski izdevīgus risinājumus un kvalitatīvus transportlīdzekļus, kas atbilst viņu budžetam un prasībām. Šī vietne ir domāta ikvienam, kurš vēlas saņemt pilnīgu informāciju un veikt izglītotu izvēli, iegādājoties automašīnu.

Šī vietne būs ceļvedis caur automašīnu izvēles dzīlēm, nodrošinot lietotājiem skaidru un saprotamu pieredzi, kas vienlaikus būs informatīva un emocijām bagāta.

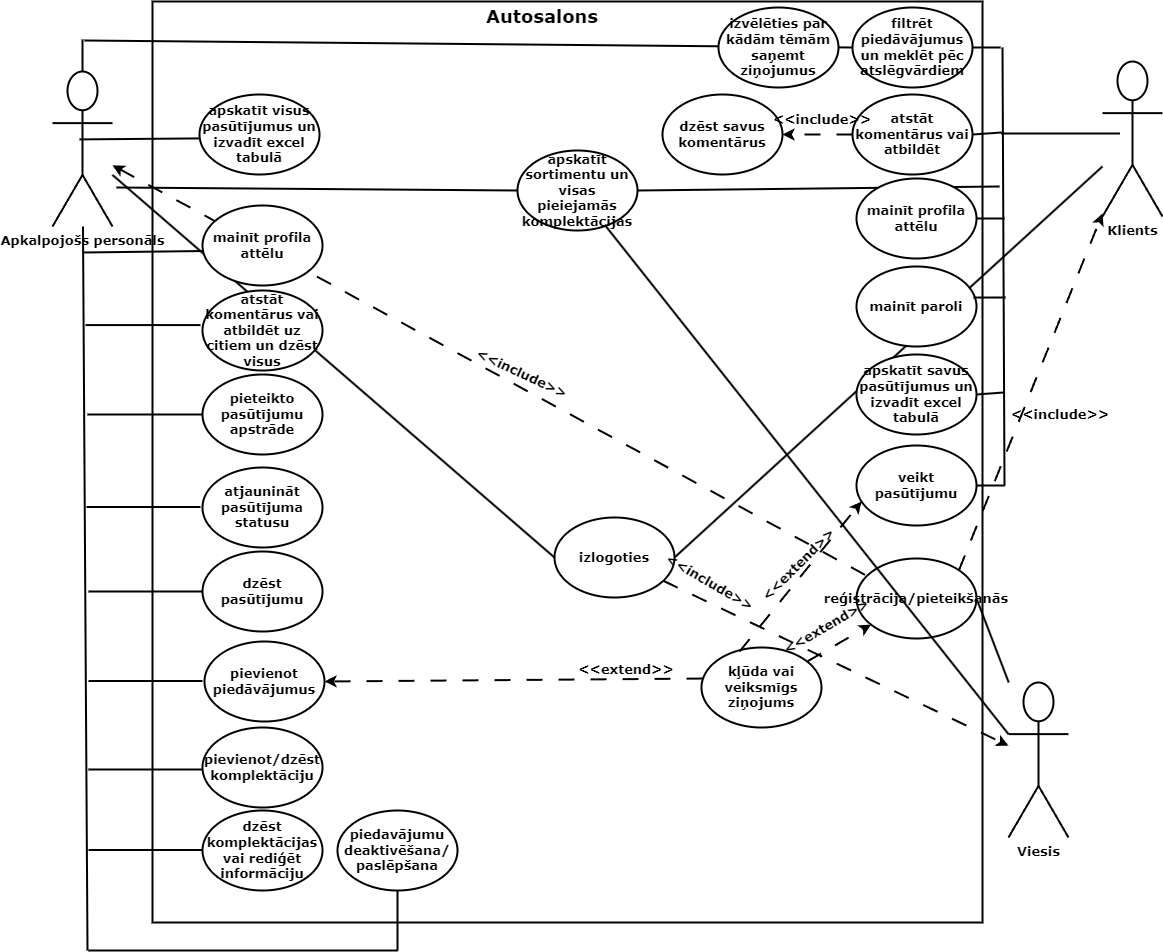
# **UZDEVUMA NOSTĀDNE**

Kvalifikācijas darba uzdevums ir izveidot tīmekļa lietotni, kas atvieglos automašīnu meklēšanu un pārdošanu tiešsaistē . Sistēmā nepieciešams realizēt iespēju efektīvi meklēt un filtrēt automašīnas, nodrošinot lietotājiem vienkāršu un ērtu veidu, kā izvēlēties automašīnu, kas atbilst viņu vajadzībām un budžetam*.*

Šī vietne būs ļoti aktuāla, ņemot vērā nepārtraukto automašīnu pieprasījumu un nepieciešamību rīkoties ar plašu informāciju par pieejamajiem auto piedāvājumiem. Tīmekļa vietne tiks nodrošināta personām, kuru vecums ir no 18 gadiem un kuriem jau ir transportlīdzekļa vadītāja apliecība, lietotājiem, kuri, iespējams, nav ļoti labi orientējušies automobiļu markās, lai viņiem būtu vieglāk izvēlēties sev piemērotu automašīnu, kā arī tīmekļa vietne būs piemērota arī cilvēkiem, kuri jau ir izvēlējušies.

Ir plānotas vairākas funkcijas (skat. 1.1.att.):

* detalizētas informācijas par automašīnu apskate un nepieciešamās preces meklēšana pēc atslēgvārdiem un filtrēšanas kritērijiem;
* apskatīt piedāvājumu visas pieejamās konfigurācijas. Pievienot vai dzēst konfigurāciju variantus. Paslēpt/deaktivēt konfigurāciju.;
* piedavājumu pievienošana vai dzēšana un informācijas rediģēšanas, piemēram, mainīt cenu, krāsu un citus datus par piedāvājumu;
* veikt pasūtījumu. Apskatīt visus savus pasūtījumus un izvadīt excel tabulā. Apkalpojošajam personālām būs iespēja apskatīt visus pasūtījumus un dzēst tos;
* apkalpojošs personāls noteiks pasūtījuma statusu un pasūtījuma izpildes procesā mainīs pasūtījuma izpildes statusu;
* atbildēt uz jautājumu vai uzdot savu atsevišķajā lapā un dzēst tos. Apkalpojošajam personālām būs iespēja dzēst jebkuru komentāru;
* iespēja mainīt sava profila datus, tādus kā: profila attēlu un paroli. Apkalpojošajam personālām būs iespēja mainīt tikai attēlu;
* iespēja izvēlēties par kādām tēmām saņemt ziņojumus;
* reģistrācija un autorizācija.



1.1.att. Lietojumgadījuma diagramma

# **PRASĪBU SPECIFIKĀCIJA**

## **Ieejas un izejas informācijas apraksts**

### *Ieejas informācijas apraksts*

Sistēmā tiks nodrošināta šādas ieejas informācijas apstrāde.

1. Informācija par lietotāju sastāvēs no šādiem datiem.

* Lietotājvārds – lietotājvārds, kuru izdomās lietotājs – burtu un ciparu teksts ar izmēru līdz 20 rakstzīmēm. (piem. “Roberts”).
* Ē-pasts – lietotāja ē-pasts – burtu un ciparu teksts ar izmēru līdz 40 rakstzīmēm. (piem. “epasts@gmail.com”).
* Parole – lietotāja parole – burtu un ciparu teksts ar izmēru līdz 255 rakstzīmēm. (piem. “324dapasQtEord34”).
* Lietotāja privilēģijas – lietotāja loma – ciparu teksts ar izmēru līdz 10 rakstzīmēm. (piem. “0”).
* Attēls – lietotāja profila attēls – burtu un ciparu teksts bez ierobežojuma uz rakstzīmēm. (piem. ‘../img/avatar/attēls.png’).
* Pieņemto noteikumu statuss - šis lauks norāda uz lietotāja statusu attiecībā uz noteikumu pieņemšanu forumā - ciparu teksts ar ierobežojumu līdz 1 ciparam. (piem. “1”).

Visi dati tiks ievadīti no tastatūras, izņemot lietotāja privilēģijas un pieņemto noteikumu statusu, to piešķirs sistēma automātiski.

2. Informācija par piedāvājumiem sastāvēs no šādiem datiem.

* Marka – automašīnas marka – burtu un ciparu teksts ar izmēru līdz 20 rakstzīmēm. (piem. “MBW”).
* Modelis – automašīnas modelis – burtu un ciparu teksts ar izmēru līdz 20 rakstzīmēm. (piem. “X5”).

Visi dati tiks ievadīti no tastatūras.

3. Informācija par piedāvājuma informāciju sastāvēs no šādiem datiem.

* Cena – automašīnas sākuma cena – ciparu teksts bez ierobežojuma uz rakstzīmēm. (piem. “32000”).
* Izgatavošanas gads – automašīnas izdošanas gads – ciparu teksts ar izmēru līdz 10 rakstzīmēm. (piem. “21.09.2012”).
* Massa – automašīnas massa – ciparu teksts bez ierobežojuma. (piem. “2500”).
* Virsbūves tips - šis lauks norāda uz transportlīdzekļa korpusa tipu - burtu un ciparu teksts ar izmēru līdz 20 rakstzīmēm. (piem. “Sedan”).

Visi dati tiks ievadīti no tastatūras.

4. Informācija par specifiskas detaļas sastāvēs no šādiem datiem.

* Izveidošanas datumu – komplektācijas izveidošanas datums – burtu un ciparu teksts bez ierobežojuma uz rakstzīmēm. (piem. 2024-04-18 15:09:13).
* Komplektācijas statuss – šīs statuss attēlo vai komplektācija ir pieiejama lietotājiem – ciparu teksts ar izmēru līdz 10 rakstzīmēm. (piem. “0”).

Visus datus piešķirs sistēma automātiski.

5. Informācija par dzinējs sastāvēs no šādiem datiem:

* Dzinēja tips – komplektācijas dzinēja tips – burtu un ciparu teksts ar izmēru līdz 50 rakstzīmēm. (piem. “Diesel”).
* Dzinēja cena – komplektācijas dzinēja cena – ciparu teksts bez ierobežojuma uz rakstzīmēm. (piem. “5000”).

Visi dati tiks ievadīti no tastatūras.

6. Informācija par transmisija sastāvēs no šādiem datiem:

* Transmisijas tips – komplektācijas transmisijas tips – burtu un ciparu teksts ar izmēru līdz 20 rakstzīmēm. (piem. “Manual”).
* Transmisijas cena – komplektācijas transmisijas cena – ciparu teksts bez ierobežojuma uz rakstzīmēm. (piem. “1500”).

Visi dati tiks ievadīti no tastatūras.

7. Informācija par automašīnu krāsām sastāvēs no šādiem datiem.

* Krāsa – automašīnas krāsas variants – burtu un ciparu teksts ar izmēru līdz 50 rakstzīmēm. (piem. “Balts”).
* Krāsas cena – automašīnas krāsas cena – ciparu teksts bez ierobežojuma uz rakstzīmēm. (piem. “150”).
* Attēls – attēls ar konkrētu piedāvājuma krāsu – burtu un ciparu teksts bez ierobežojuma uz rakstzīmēm. (piem. ‘../img/attēls.png’).

Visi dati tiks ievadīti no tastatūras.

8. Informācija par komentāriem sastāvēs no šādiem datiem.

* Sūtīšanas laiks – komentārija atstāšanas datums – datums. (piem. “2023-10-05 19:56:59”).
* Komentārs – komentārijs, kuru ieraksta lietotājs – burtu un ciparu teksts ar izmēru līdz 700 rakstzīmēm. (piem. “Ļoti labs serviss, visiem ieteicu!”).
* Vecāku komentārs - šis lauks ir saite uz vecāku komentāru, kas attiecas uz konkrēto ierakstu - ciparu teksts ar izmēru līdz 10 rakstzīmēm. (piem. “314”).

Visi dati tiks ievadīti no tastatūras, izņemot nosūtīšanas laiku un saiti uz vecāku komentāriju, to piešķirs sistēma automātiski.

9. Informācija par pasūtijumu sastāvēs no šādiem datiem.

* Vārds – pasūtītāja vārds – burtu un ciparu teksts ar izmēru līdz 20 rakstzīmēm. (piem. “Janis”).
* Uzvārds – pasūtītāja uzvārds – burtu un ciparu teksts ar izmēru līdz 20 rakstzīmēm. (piem. “Ozols”).
* Pasūtīšanas datums – pasūtijuma datums – datums. (piem. “11.04.2023”).
* Pasūtījuma izpildes statuss – pasūtijuma apstiprinājums un izpildes posms – burtu un ciparu teksts ar izmēru līdz 15 rakstzīmēm. (piem. “New”).
* Telefona numurs – pasūtītāja telefonas numurs – burtu un ciparu teksts ar izmēru līdz 20 rakstzīmēm. (piem. “+371 20014552”).

Visi dati tiks ievadīti no tastatūras, izņemot pasūtījuma izpildes statusu un pasūtīšanas datumu, to piešķirs sistēma automātiski.

10. Informācija par ziņojuma tēmām sastāvēs no šādiem datiem.

* Tēmas nosaukums - šis lauks norāda uz tēmas nosaukumu, kurš attiecas uz ziņojumu - burtu un ciparu teksts ar izmēru līdz 255 rakstzīmēm. (piem. “Orders”).
* Status - šis lauks norāda uz stāvokli, kas attiecas uz attēlojumu uznirstošajiem paziņojumiem lietotājam. "Enable" nozīmē, ka uznirstošie paziņojumi ir aktīvi un redzami lietotājam, kamēr "Disable" nozīmē, ka uznirstošie paziņojumi ir deaktivēti un netiks attēloti lietotājam. Šie paziņojumi tiek rādīti visiem lietotājiem, izmantojot modālo logu - burtu un ciparu teksts ar izmēru līdz 10 rakstzīmēm. (piem. “enable”).

Visus datus piešķirs sistēma automātiski, vienīgais lietotājs varēs pārslēgt "enable/disable".

11. Informācija par ziņojumiem sastāvēs no šādiem datiem.

* Ziņojums – teksts, kas informē lietotāju – burtu un ciparu teksts bez ierobežojuma uz rakstzīmēm. (piem. “You have successfully registered! If you need Help, please visit the Help section.”).
* Saņemšanas laiks – ieraksts ar ziņojuma aizsūtīšanas laiku – datums un laiks. (piem. “2023-10-05 19:56:59”).
* Izlasīšanas statuss – ieraksts, kur glabāsies vai lietotājs izlasījis ziņu, vai nē – ciparu teksts ar izmēru līdz 1 rakstzīmei. (piem. “0”).

Visus datus piešķirs sistēma automātiski.

### *Izejas informācijas apraksts*

1. **Tabula ar pasūtījumiem Excel formātā.** Kopumā būs 2 tabulas, lietotājam būs tikai viņa konkrētie pasūtījumi, bet apkalpojošam personālam būs tabula ar visu lietotāju pasūtījumiem. Katrs verificēts lietotājs varēs skatīt pasūtījumu vēsturi un detalizētu informāciju par tiem Excel tabulā. Tur atradīsies kontaktinformācija un informācija par automašīnu, kā arī krāsas cena un kopējā summa.
2. **Lietotāja profils tīmekļu lapā.** Visa informācija par lietotāju: lietotājvārdu un e-pastu, tiks parādīta atsevišķā lapā. Un arī tiks parādīta visu pasūtījumu kopējā summa un tabula ar visiem pasūtījumiem (lietotāja un piedāvājuma detalizēta informācija).
3. **Paziņojumi lietotājam.** Vietnē lietotājam parādīsies īpašs attēls, kad viņš veiks tādas darbības kā reģistrācija un pasūtījuma noformēšana. Paziņojums informēs lietotāju, ka darbība ir veiksmīgi pabeigta un kur ir jāvēršās, ja ir kādi jautājumi.
4. **Piedavājuma lapa.** Visa informācija par piedavājumu: cena, massa, izdošanas gads un krāsu varianti, tiks parādīta atsevišķā lapā.
5. **Lapa ar komentāriem.** Visas atsauksmes par servisu vai jautājumi no lietotājiem būs pieiejamas atsevišķajā lapā.
6. **Pasūtījuma apstiprināšanas logs.** Pasūtījuma apstiprinājuma lapā izvadīt visus datus par pasūtījumā izvēlēto automašīnu un iekārtu, kā arī lietotāja kontaktinformācijas izvadīšanu.

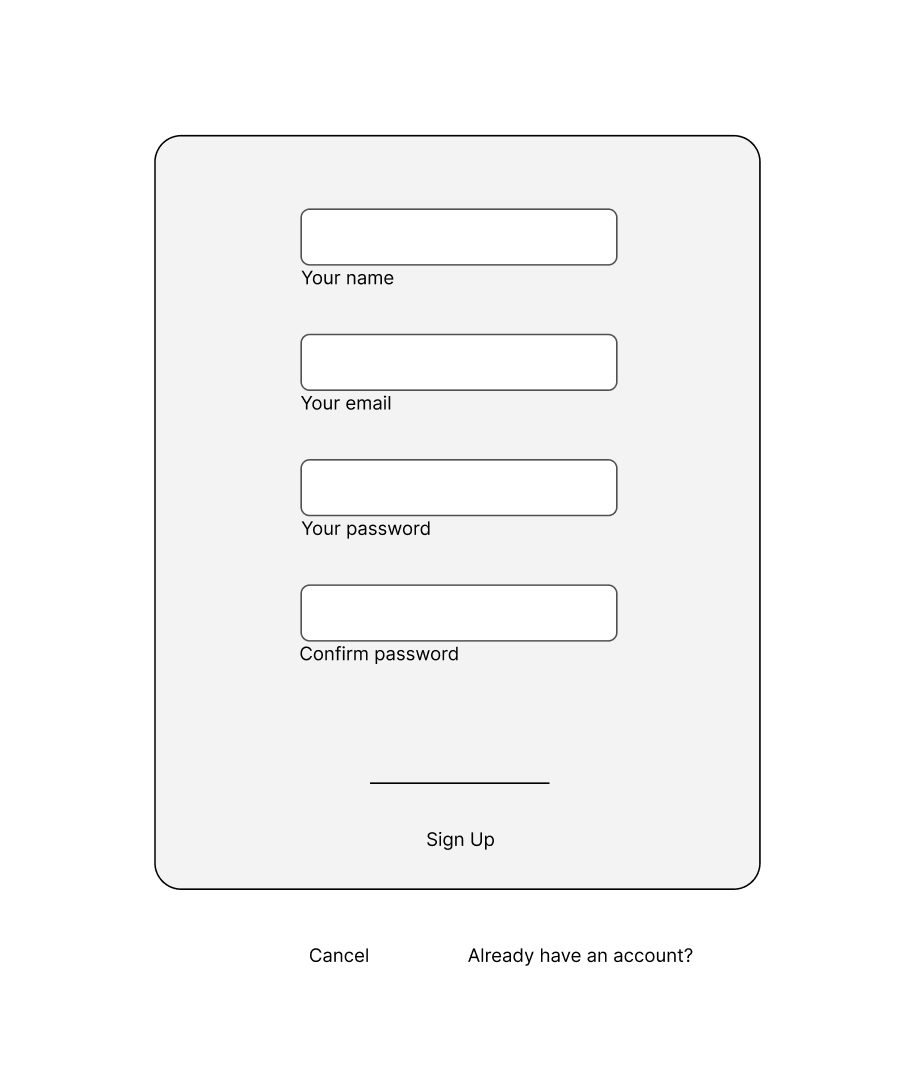
## **Funkcionālās prasības**

1. Jaunu lietotāju reģistrācija:
   1. Jāparedz ieejas informācijas par lietotāju ievadīšana, pārbaude uz simbolu skaitu un formāta pareizību.
   2. Obligātu lauku pārbaude un kļūdas paziņojumu izvadīšana.
   3. Lietotāja vārda un ē-pasta salīdzināšana ar esošajiem lietotājiem. Izvadīs kļūdu ar atbilstošo tekstu.
   4. Minimālās prasības paroles ilgumam un sarežģītībai.
   5. Paroles glabāšana, izmantojot drošu hašēšanas algoritmu. Parole tiks ievadīta divas reizes, lai izvairītos no drukas kļūdām. Ja parole, kura tiks ievadīta atkārtoti, nesakrītīs ar sākotnējo variantu, tiks parādīta kļūda. -Pēc veikmīgās lietotāja reģistrācijas, ziņojuma logā atnāks vēstule.
2. Lietotāja autorizācija:
   1. Ja lietotājs eksistē datu bāzē, sistēmai ir jānodrošina autorizācija, pieslēdzoties ar lietotājvārdu un paroli.
   2. Ja kāds no laukiem nav ievadīts, izvadīt par to paziņojumu.
   3. Lauku ievades pārbaude uz pareizību un kļūdas paziņojumi, ja parole vai vārds nesakrīt.
   4. Kļūdu izvade, ja ievadītais lietotājvārds neeksistē datu bāzē.
3. Meklēšanas funkcija automašīnu klāstā:
   1. Ja lietotājs ievada marku vai modeli, vai visu kopā, sistēmai ir jāparāda visi attiecīgie piedāvājumi.
   2. Ja nav atrasti piedāvājumi, izvadīsies atbilstošs teksts.
   3. Sistēmai ir jānodrošina meklēšanu pēc filtru saraksta, lai lietotāji varētu viegli atlasīt vajadzīgos kritērijus, tādus kā cenu intervāls, krāsa, ātrumkārba, marka, modelis, virbūves tips, izdošanas gads.
   4. Sistēmai ir jānodrošina meklēšana pēc kritērijiem un atslēgvārdiem kopā.
4. Forma automašīnas pirkšanai ar visiem nepieciešamajiem laukiem:
   1. Obligātu lauku pārbaude, telefona numurs, vārds un uzvārds, un arī ka lietotājs piekrīt nosacījumiem.
   2. Sistēmai ir jāparbauda vai lietotājs ievada tieši ciparus laukā priekš telefona numura.
   3. Sistēmai jāsaglabā informāciju par izvēlēto automašīnas komplektāciju.
   4. Jaizvada logu ar visu informāciju par pasūtijumu un saziņas informāciju priekš apstiprināšanai.
5. Piedāvājumu pievienošanas funkcija:
   1. Datu ivedīšana un obligātu lauku pārbaude.
   2. Pārbaude uz maksimālo simbolu skaitu.
6. Komplektāciju pievienošana/dzēšana:
   1. Iespēja pie jau eksistējošā modeļa pievienot koplektācijas. Datu ivedīšana.
   2. Obligātu lauku pārbaude.
   3. Rakstzīmju ierobežojums.
   4. Ir iespēja dzēst izvēlēto komplektāciju.
7. Piedāvājumu paslēpšana/deaktivēšana:
   1. Iespēja paslēpt konfigurāciju, lai lietotājam netiktu rādīti piedāvājumi, kas ir atzīmēti kā neaktīvi.
8. Pasūtījumu apskate un apstrāde:
   1. Apkalpojošajam personālam jābūt iespējai apsktīt visus lietotāju pasūtījumus.
   2. Iespēja mainīt pasūtījuma izpildes statusu.
   3. Lietotājam iespēja apskatīt savus pasūtījumus.
   4. Katram autorizētam lietotājam jābūt iespējai izvadīt pasūtījumus excel tabulā.
9. Komentāru atstāšana:
   1. Obligātu lauku pārbaude un kļūdas paziņojumu izvadīšana.
   2. Lietotāju iespēja atstāt savus komentārus.
   3. Iespēja atbildēt uz komentāriem.
   4. Komentāru laukā jābūt pārbaude uz simbolu skaitu.
10. Komentāru dzēšana:
    1. Lietotāju iespēja dzēst savus komentārus.
    2. Apkalpojošajam personālam jābūt iepējai dzēst jebkādu komentāru.
11. Personīgais lietotāja konts un to rediģēšana:
    1. Konta informācija un personalizācijas iespējas, konta attēla maiņa.
    2. Lietotāja paroles rediģēšanas. Ja tagadēja parole ir ievadīta nepareizi vai jauna parole un atkārtoti ievadīta nesakritīs, tad izvadīs kļūdas. Jaunai parolei jābūt vismaz 6 rakstzīmju garumā.
    3. Sistēmai ir jānodrošina visu pasūtījumu apskati priekš katra lietotāja un izvadi excel formātā.
12. Iziešana no profila:
    1. Katram lietotājam jābūt iespēja iziet no lietotāja konta.
13. Ziņojumu saņemšanas sistēma:
    1. Sistēmai jānosūta paziņojums par veiksmīgu reģistrāciju.
    2. Sistēmai jānosūta paziņojums par veiksmīgu pasūtīšanu.
    3. Sistēmai jānosūta paziņojums, ja uz lietotāja komentāru atbildēja.
14. Ziņojumu paslēpšana pēc tēmatiem:
    1. Lietotājiem jābūt iespējai paslēpt ziņas par tēmām, kas viņu neinteresē ("Forums", "Pasūtījumi", "Profila izmaiņas").
15. Ziņojuma dzēšana:
    1. Lietotājam vajadzētu būt iespēja dzēst ziņojumus no sava konta pasta.
    2. Lietotājam vajadzētu būt iespēja dzēst ar vienu pogu visus ziņojumus no sava konta pasta.
16. Transportlīdzekļa komplektāciju variantu attēlošana:
    1. Visiem lietotājiem jābūt iespējai apskatīt automobiļa komplektāciju variantus, un sistēmai jāuzrāda visas komplektācijas, kas ir saistītas ar izvēlēto modeli.

## **Nefunkcionālās prasības**

1. Veiktspēja: sistēmai jānodrošina atbilde lietotājam ne ilgāk kā 2 sekundes laikā un datu bāzes jautājumiem jābūt izpildītiem ne ilgāk kā 5 sekundēs.
2. Drošība: lietotāja paroles jābūt saglabātām, izmantojot šifrēšanu. Piekļuve administratīvajām funkcijām jābūt tikai autorizētiem darbiniekiem.
3. Dizains: lietotāja saskarnē jābūt viegli saprotamai un draudzīgai lietotājiem no visdažādākajām paaudzēm. Lietotāja saskarnei jābūt responsīvai, lai nodrošinātu labu pieredzi gan datoriem, gan mobilajiem ierīcēm.
4. Aizsardzība pret SQL injekcijām. Šī prasība paredz nodrošināt, ka sistēma ir pasargāta no iespējamām SQL injekcijām, kas var rasties, ja nepietiekami tiek pārbaudīta un filrēta lietotāja ievadītā informācija.
5. Saprotama un viegla navigācija. Šī prasība attiecas uz lietotāja pieredzi, nodrošinot sistēmu ar saprotamu un intuitīvu navigāciju.
6. Saprotama administrāciju lapa priekš apkalpojošā personāla. Šeit svarīgs ir sistēmas pārvaldības skatījums, kas ir saprotams un vienkāršs, it īpaši apkalpojošam personālam vai administratoriem.
7. Vietne darbotos bez kļūdām visās populārākajās pārlūkprogrammās, piemēram, Google Chrome, Opera, Firefox, Edge, Safari.
8. Sistēmai ir nepieciešama spēja viegli pielāgoties jaunām funkcijām vai vides izmaiņām, nodrošinot tās atjaunināšanu un paplašināšanu.
9. Sistēmai jābūt aprīkotai ar automatizētu darbību iespējām, lai nodrošinātu efektīvu darbību un minimizētu cilvēka iesaisti, it īpaši attiecībā uz rutīnas procesiem vai ilgstošām darbībām.
10. Jābūt nodrošinātam vispusīgam un pieejamam sistēmas atbalstam, lai ātri novērstu jebkādas darbības vai sistēmas kļūmes, kā arī nodrošinātu lietotāju ar palīdzību un atbalstu sistēmas izmantošanā.

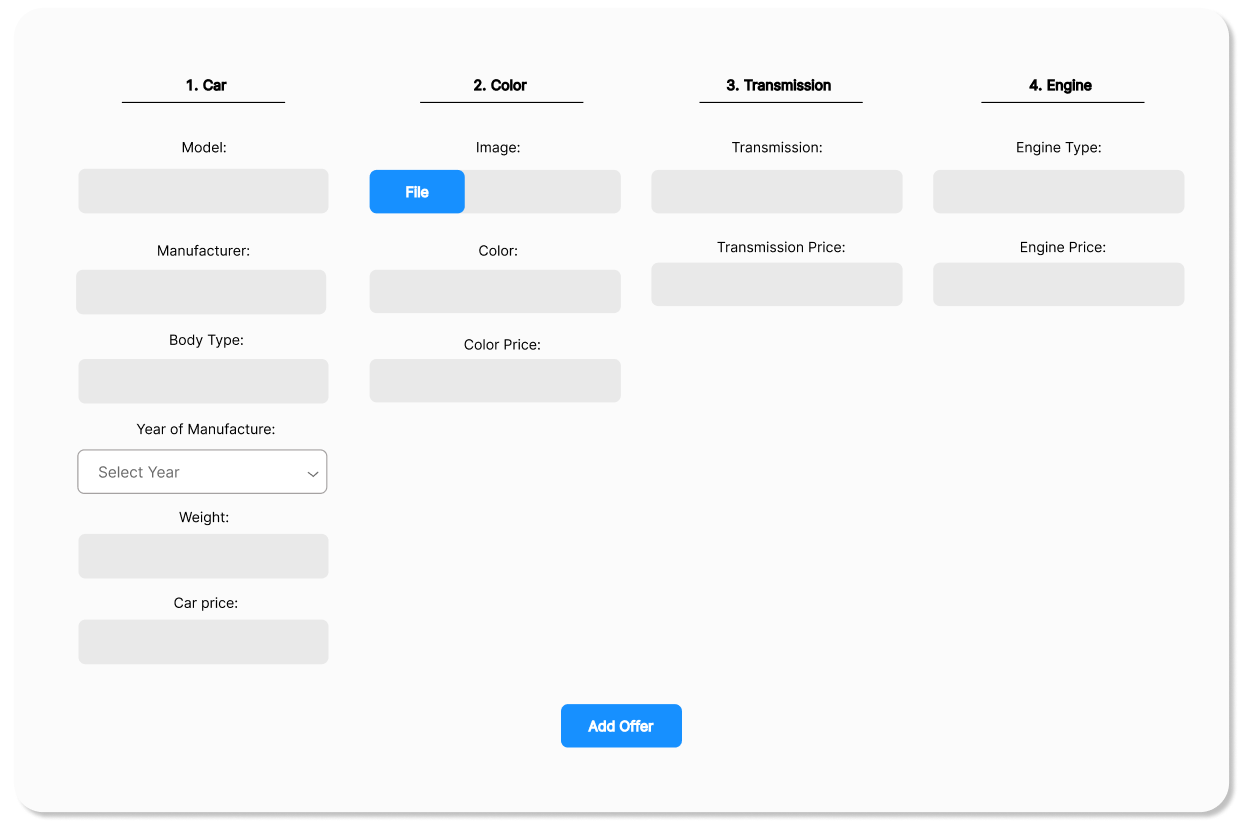
* Sistēmas reģistrācijas formas skice (skat. 2.1.att.)



2.1.att. Sistēmas reģistrācijas formas skice

Šī skice attēlo sistēmas interfeisu, kurā lietotāji var reģistrēties.Šajā formā ir 4 lauki un 3 pogas “Sign Up”, lai piereģistrētu jauno lietotāju, “Cancel”, lai atceltu darbību un poga “Already have an account?”, ja lietotājam jau ir konts šajā vietnē.

* Automašīnu pievienošanas formas skice (skat. 2.2.att.)



2.2.att. Automašīnu pievienošanas formas skice

Šī skice attēlo sistēmas interfeisu, kurā apkalpojošs personāls var pievienot jaunas automašīnas.

# **UZDEVUMA RISINĀŠANAS LĪDZEKĻU IZVĒLES PAMATOJUMS**

Sistēma ir izstrādāta kā pārlūkprogramma, kas ir pielāgota gan personālo datoru, gan mobilo tālruņu izšķirtspējas prasībām, nodrošinot optimālu lietojamību un atbilstošu vizuālo pieredzi neatkarīgi no ierīces veida. Šādas prasības ir noteikušas sistēmas arhitektūras izvēli un tehnoloģiju risinājumu izstrādi gan lietotāja, gan servera pusē. Izvēlētie risinājumi ir veidoti, lai sasniegtu augstu veiktspēju un sniegtu lietotājiem nepieciešamo funkcionalitāti. Šī pieeja nodrošina sistēmas darbību, pievēršot uzmanību efektivitātei un ilgtspējīgai darbībai.

Lietotāja daļā tika izmantots:

* **HTML** (**versijā** **5.3)** ir standartizēta programmēšanas valoda, kas tiek izmantota tīmekļa lapu izstrādē un strukturizēšanā. Tā ir galvenā valoda, kas nosaka, kā informācija tiek organizēta un attēlota tīmekļa lapās. HTML 5.3 ir jaunākā HTML versija, piedāvājot daudzveidīgus elementus un funkcijas, kas padara lapas izveidi un interaktivitāti daudz vienkāršāku un efektīvāku.
* **CSS (versijā 3)** ir valoda, kas tiek izmantota, lai definētu tīmekļa lapas izskatu un stilu. Tā nodrošina iespēju vizuāli noformēt tīmekļa lapas, piemēram, definēt teksta izmērus, krāsas, pozicionēšanu, fonu un citas vizuālas īpašības. CSS 3 ir pēdējā stabilā versija, kas piedāvā plašas iespējas un funkcijas stilu definēšanai tīmekļa lapās. Izmantojot CSS, ir iespējams veidot tīmekļa lapu izskatu, pielāgojot to saskaņā ar dizaina prasībām un lietotāja vēlmēm.
* **JavaScript (versijā 1.8.5)** ir viena no populārākajām programmēšanas valodām tīmekļa izstrādei. Tā tiek izmantota dinamiskiem un interaktīviem elementiem, piemēram, animācijām un veidlapu validācijai. Šī valoda ļauj veidot dinamisku saskarni, uzlabot vietņu funkcionalitāti un nodrošināt lietotājiem interaktīvu pieredzi.
* **Bootstrap** ir front-end (klienta puse) rīks, kas tiek izmantots, lai veidotu stilizētas un responzīvas tīmekļa lapas. Tas piedāvā daudzus gatavus komponentus un stilus, lai veidotu modernu un pievilcīgu interfeisu tīmekļa vietnēm.
* **SweetAlert (1. versijā)** ir JavaScript bibliotēka, kas tiek izmantota, lai izveidotu pievilcīgas un pielāgotas modālas dialoga logus, kas var tikt izmantoti, lai parādītu ziņojumus, brīdinājumus vai pieprasītu lietotāja apstiprinājumu darbībai. Tas padara lietotāja pieredzi vairāk interaktīvu un patīkamu.

Servera daļā tika izmantots:

* **PHP (versija 8)** ir viena no visplašāk izmantojamajām programmēšanas valodām tīmekļu izstrādei. PHP ir spēcīgs rīks, kas tiek izmantots dinamisku un interaktīvu vietņu izveidei, datu apstrādei un daudzām citām tīmekļa lietojumprogrammu funkcijām. Tajā ir plaša funkciju bibliotēka, kas padara PHP par vienu no vadošajām valodām tīmekļa izstrādē, nodrošinot daudzveidīgas iespējas tīmekļa vietņu un aplikāciju izveidei.

Izstrādes vides, kas tika izmantotas:

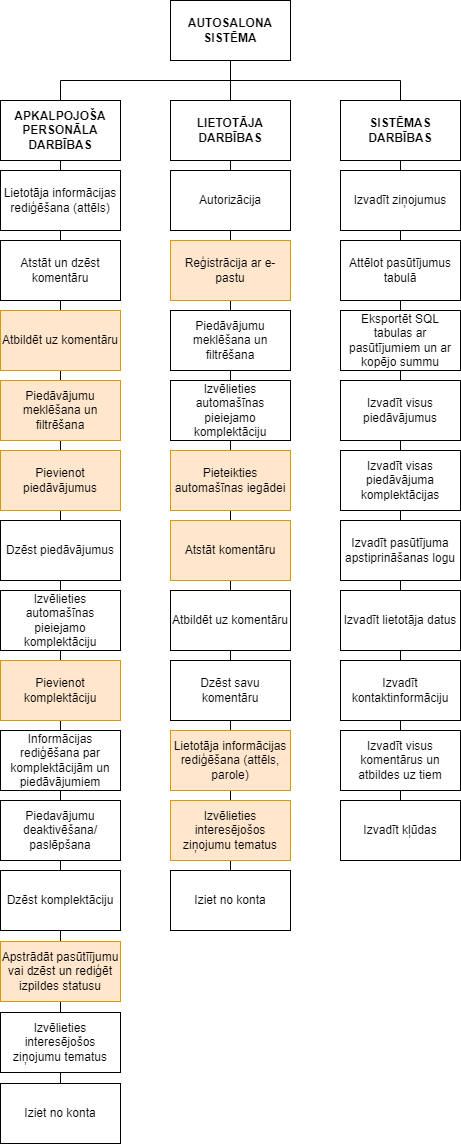
* **Laragon (versijā 6.0)** ir viegls un jaudīgs serveris, kas ļauj izstrādātājiem ātri instalēt un iestatīt lokālo tīmekļa serveri savā datorā. Tas ļauj ātri un ērti pārbaudīt un atkļūdot tīmekļa lietojumprogrammas, kā arī izvietot tās attālos serveros. Laragon Full (64 bitu) versija satur visus nepieciešamos komponentus, lai izstrādāt un palaistu tīmekļa lietojumprogrammas.
* **GIT (versijā 2.40.0)** ir izplatīta versiju kontroles sistēma, kas ļauj izstrādātājiem izsekot un pārvaldīt avota koda izmaiņas. Tas ļauj efektīvāk pārvaldīt tīmekļa lietojumprogrammu izstrādi un uzturēšanu, kā arī vajadzības gadījumā ērti atgriezties pie iepriekšējām koda versijām.
* **Visual Studio Code (versija 1.85.0)** ir jaudīgs un viegls koda redaktors, kas nodrošina plašu funkciju klāstu ērtai un efektīvai kodēšanai. Tam ir arī daudzi paplašinājumi, kas var atvieglot jūsu darbu.

# **PROGRAMMATŪRAS PRODUKTA MODELĒŠANA UN PROJEKTĒŠANA**

## **Sistēmas struktūras modelis**

### *Sistēmas arhitektūra*

Sistēmu veido trīs apakšsistēmas (skat. 4.1. attēlu): apkalpojošo personālu modulis, lietotāja modulis, sistēmas modulis. Ar oranžu krāsu ir atzīmētas funkcijas, kuras tiks izmantotas kā piemēri 4.2.1. punktā datu plūsmu diagrammās.



4.1.att. FD diagramma

* **Lietotāju modulis.** Nereģistrētam lietotājam būs iespēja veikt šādas funkcijas: apskatīt automašīnu klāstu, meklēt automašīnu pēc nosaukuma, izgatavotāja, minimālās un maksimālās cenas, izdošanas gada, krāsas, virsbūves tipa, transmisijas. Būs iespējams arī pierēğistrēties, ja viņš piereğistrēsies vai ieies kontā, tad tam parādīsies vairāk iespēju un viņš būs "klients" vai “apkalpojošs personāls” (atkarībā no tā, kādu paroli un lietotājvārdu ievadīs). Reģistrēts lietotājs varēs veikt šādas funkcijas: atstāt komentāru vai atbildēt uz citiem un dzēst to, aizpildīt formu, lai veiktu pasūtījumu, mainīt profila attēlu un paroli, apskatīt savus datus un informāciju par saviem pasūtījumiem, ka arī izvadīt excel tabulā, ir iespēja izvēlēties par kadām tēmām saņemt ziņojumus un kādus paslēpt, kā arī dzēst.
* **Apkolpojošo personālu modulis.** Apkalpojošajam personālam būs iespēja veikt šādas funkcijas: apskatīt visu automašīnu klāstu, mainīt profila attēlu, apstiprināt pasūtijumu un pasūtījuma izpildes procesā mainīt pasūtījuma izpildes statusu, dzēst pasūtijumu, apskatīt klienta informāciju, izvadīt tabulu ar klientu pasūtījumiem excel tabulā, atbildēt uz jautājumu vai dzēst to, pievienot vai rediģēt piedāvājumus un dzēst tos, pievienot vai dzēst komplektāciju, paslēpt piedāvājumu, lai lietotājam netiktu rādīti piedāvājumi, kas ir atzīmēti kā neaktīvi, ir iespēja izvēlēties par kadām tēmām saņemt ziņojumus.
* **Sistēmas modulis.** Platforma automātiski parādīs lietotāja izdarītās kļūdas, apkopos visus komentārus un atbildes atsevišķā sadaļā, nodrošinās informāciju lietotājam, kā arī attēlos lietotāja informāciju profila lapā un pasūtījumus Excel failā, izvadīs ziņojumus atsevišķajā logā, attēlos visus piedāvājumus un komplektācijas saistītas ar tiem, ka arī izvadīs pasūtījuma apstiprināšanas logu pēc kontaktinformācijas ievadīšanas.

### *Sistēmas ER modelis*

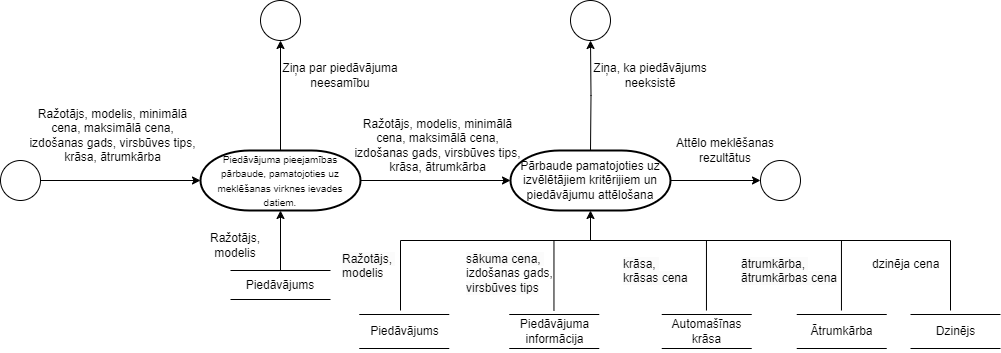
Datu bāzes projektēšanā datu kopu un saišu starp tām attēlošanai tika lietota relācijusaišu diagramma, kas sastāv no divām veidu objektiem – entītijām (loģiskais objekts) un relācijām(attiecības). Sistēmas ER modelis (sk.1. pielikumu) sastāvēs no 11 entītijām. Tas nodrošina pamat informācijas uzglabāšanu un apstrādi. Tie ir:

* “Komentārijs” – apraksta komentārus, kas raksta lietotāji. Tās atribūtu kopums sevī ietver datumu, paša komentāra saturu un saiti uz vecāku komentāriju (Glabā saiti uz komentāriju uz kuru atbildēja. Ja tas ir oriģināls, neko neglabās.). Tabula “Komentārijs” ir saistīta ar tabulu “Lietotājs”. Vienam lietotājam var būt vairāki komentāri (sk.4. attēlu), bet komentāram tikai viens lietotājs.
* “Piedavājums” – superklase, apraksta visus piedāvājumus. Tabula “Piedavājums” ir saistīta ar tabulu “Piedavājuma informācija”, “Pasūtījums” un “Automašīnas krāsa”. Vienam piedavājumam var būt vairākie krāsu varianti. Piedavājums var būt vairākos pasūtījumos, bet pasūtījumā tikai viena automašīna. Tās atribūtu kopums sevī ietver, automašīnas marku un modeli.
* “Piedavājuma informācija” – “Piedavājums” klases apakšklase, apraksta piedāvājumus, dodot plašāku informāciju, tika izmantots, lai optimizēt tabulu “Piedavājums”. Tabula “Piedavājuma informācija” ir saistīta ar tabulu “Piedavājums”. Attiecība būs viens pret vienu, katram piedāvājumam piederēs atbilstoša informācija. Tās atribūtu kopums sevī ietver cenu, izdošanas gadu, virsbūves tipu un massu.
* “Specifiskas detaļas” – starptabula, kas glabā ārējās atslēgas no tabulām: "Automašīnas krāsa", “Transmisija”, “Dzinējs”, apraksta automašīnu komplektācijas (komplektācija sastāv no: dzinēja tipa, dzinēja cenas, transmisijas tipa, transmisijas cenas, krāsas un krāsas cenas.), dodot plašāku izvēli lietotājam. Šī tabula tika izveidota, lai glabātu datus, kas vienam un tam pašam modelim var atšķirties. Tabula “Specifiskas detaļas” ir saistīta ar tabulu “Piedavājuma informācija” un attiecība būs viens pret daudziem, katram piedāvājumam piederēs vairākās automašīnu komplektācijas. Vēl šī tabula glabā arējās atslēgas no tabulām - "Automašīnas krāsa", “Transmisija”, “Dzinējs” -, lai vienam automašīnas modelim varētu būt vairākas komplektācijas. Tās atribūtu kopums sevī ietver izveidošanas datumu un komplektācijas statusu.
* “Transmisija” – satur informāciju par pieejamajiem transmisijas variantiem, kuri ir izmantoti komplektāciju veidošanā. Tabula “Transmisija” ir saistīta ar tabulu “Specifiskas detaļas”, kur katrā komplektācijā ir piešķirta atbilstoša transmisija, attiecība būs viens pret daudziem. Atribūti ietver transmisijas tipu un cenu.
* “Dzinējs” – satur informāciju par pieejamajiem dzinēju variantiem, kuri ir izmantoti komplektāciju veidošanā. Tabula “Dzinējs” ir saistīta ar tabulu “Specifiskas detaļas”, kur katrai komplektācijas variantam ir piešķirts atbilstošs dzinējs, attiecība būs viens pret daudziem. Atribūti ietver dzinēja tipu un cenu.
* “Automašīnas krāsa” – satur informāciju par pieejamajām automašīnu krāsām, kuri ir izmantotas komplektāciju veidošanā. Tabula “Automašīnas krāsa” ir saistīta ar tabulu “Specifiskas detaļas”, kur katrai komplektācijas variantam ir piešķirta atbilstoša krāsa, attiecība būs viens pret daudziem. Atribūti ietver krāsas nosaukumu, attēlu, kas attēlo krāsu, un cenas informāciju.
* “Lietotājs” – superklase, apraksta cilvēkus, kas pieder sistēmai: gan klientus (potenciālie pircēji), gan apkalpojošo personālu, kas sazinās ar klientu, palīdz tiem un apstiprina pasūtijumu. Atkarībā no lietotāja statusa vērtības viņš būs personāls vai klients. Tās atribūtu kopums sevī ietver statusu, paroli, lietotājvārdu, attēlu, pieņemto noteikumu statusu un ē-pastu. Tabula “Lietotājs” ir saistīta ar tabulu “Komentārijs”, “Pasūtījums” un “Ziņojums”. Katram lietotājam var būt vairākie pasūtījumi, bet pasūtījums pieder vienam konkrētam lietotājam.
* “Pasūtījums” – ir starp tabula, kas apraksta klienta pasūtijumu. Tās atribūtu kopums sevī ietver apstaprināšanas statusu un pasūtijuma datumu, pasūtītāja vārdu, uzvārdu un telefonu. Tabula “Pasūtījums” ir saistīta ar tabulu “Piedavājums”, “Lietotājs” un “Automašīnas krāsa”. Vienam klientam būs iespēja veikt vairākus pasūtījumus, bet vienam pasūtījumam var būt tikai viens klients. Piedāvājums var piederēt vairākiem pasūtijumiem, bet pasūtijumā var būt tikai viens piedāvājums.
* “Ziņojuma tēma” – tabula, kura satur paziņojumu tēmas, kuras ir saistītas ar lietotāju. Tā atribūtu kopa ietver tēmas nosaukumu un paziņojumu tēmas statusu. Tabula “Ziņojuma tēma” ir saistīta ar tabulu “Lietotājs”. Vienam lietotājam var piederēt vairākas ziņojumu tēmas, bet katra tēma var piederēt tikai vienam lietotājam.
* “Ziņojums” – “Ziņojuma tēma” klases apakšklase, tabula, kurā glabājās paziņojumu saturs, ko saņem lietotāji. Tā atribūtu kopa ietver ziņojuma tekstu, laiku un izlasīšanas statusu. Tabula “Ziņojums” ir saistīta ar tabulu “Ziņojuma tēma”. Katrs paziņojums pieder vienai no tēmām, tāpēc vienā tēmā var būt vairāki paziņojumi.

## **Funkcionālais sistēmas modelis**

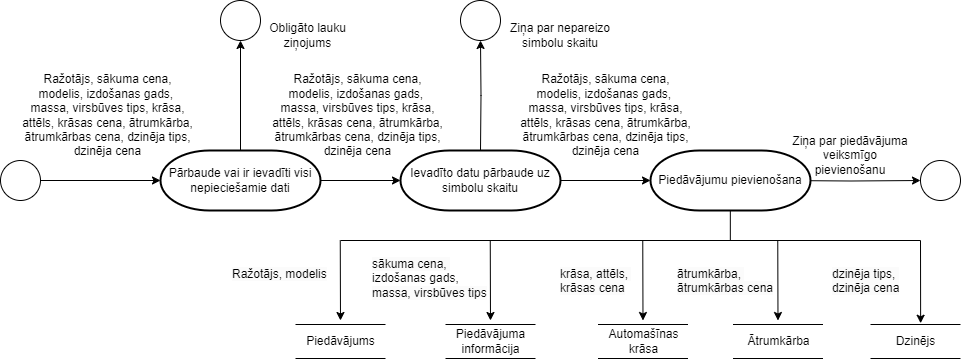
### *Datu plūsmu modelis*

1. **Piedāvājumu meklēšana un filtrēšana**. Lietotājs ievada meklēšanas virknē ražotāju un modeli (skat. 4.2. att.). Vēl ir iespējams izvēlēties meklēšanas kritērijus, piemēram: ražotājs, modelis, minimālā cena, maksimālā cena, izdošanas gads, virsbūves tips, krāsa un transmisija, kuriem ir jau iepriekš sagatavoti parametri un kuri darbosies kopā ar meklēšanas virkni. Sistēma veic atbilstošu automobiļu meklēšanu pēc ievadītajiem un izvēlētājiem kritērijiem, ņemot atbilstošos datus no tabulām: "Piedāvājums", "Piedāvājuma informācija", "Automašīnas krāsa", "Ātrumkārba(transmisija)", "Dzinējs". Ja ir atrasti atbilstoši piedāvājumi, tie tiek attēloti lietotājam. Ja nekādi atbilstoši piedāvājumi netiek atrasti, tiks parādīts paziņojums, ka nav atrasts neviens atbilstošs rezultāts.



4.2.att. Pedāvājumu meklēšanas un filtrēšanas datu plūsmu diagramma

1. **Piedāvājumu pievienošana.** Apkalpojošs personāls atver piedāvājumu pievienošanas formu un aizpilda tajā nepieciešamos laukus (skat. 4.3. att.): ražotājs, sākuma cena, modelis, izdošanas gads, massa, virsbūves tips, krāsa, attēls, krāsas cena, ātrumkārba, ātrumkārbas cena, dzinēja tips, dzinēja cena. Pēc datu ievadīšanas sistēma veic ievadīto informācijas pārbaudi, lai nodrošinātu tās precizitāti un atbilstību prasībām. Ja dati tiek ievadīti nepareizi, piemēram, ne visi lauki ir aizpildīti vai ievadīto simbolu skaits pārsniedz pieļaujamo, tiks izvadīts kļūdas ziņojums. Ja dati ir derīgi un atbilst kritērijiem, sistēma saglabā piedāvājumu datubāzē 5 tabulās: "Piedāvājums", "Piedāvājuma informācija", "Automašīnas krāsa", "Ātrumkārba(transmisija)", "Dzinējs" un izvada ziņojumu par veiksmīgo izpildi.



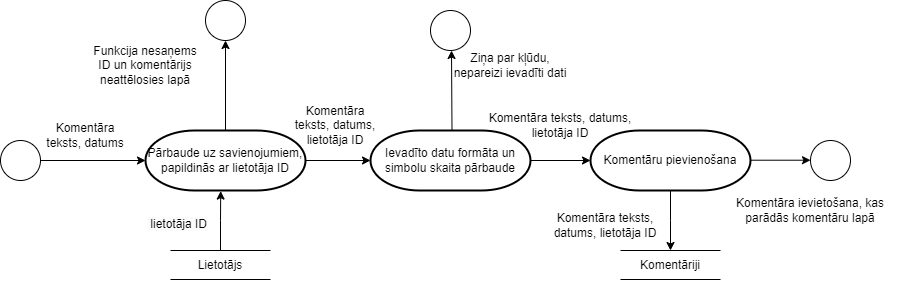
4.3.att. Piedāvājuma pievienošanas datu plūsmu diagramma

1. **Pasūtījuma izpildes statusa rediģēšana**. Apkalpojošais personāls apstrādās klientu pasūtījumus (skat. 4.4. att.), kas tiks sagrupēti trīs dažādās tabulās - "Jauns", "Procesā" un "Pabeigts". Ja klients ir nosūtījis pieprasījumu, pasūtījums automātiski tiks novietots tabulā "Jauns". Kad pasūtījuma izpildes process sāksies, apkalpojošais personāls mainīs pasūtījuma statusu uz "Procesā", visi izpildīti pasūtījumi tiks atzīmēti kā pabeigtie. Tiks pārbaudīts identificējošais numurs, lai statuss mainītos tieši izvēlētajai rindai. Ja apkalpojošais personāls veiks jebkādas izmaiņas attiecībā uz pasūtījuma statusu, šīs izmaiņas tiks atjaunotas un saglabātas datubāzē tabulā “Pasūtījumi” un izvadīs ziņu par veiksmīgo izmaiņu. Ja notiks servera kļūda, izvadīs atbilstošo kļūdu.



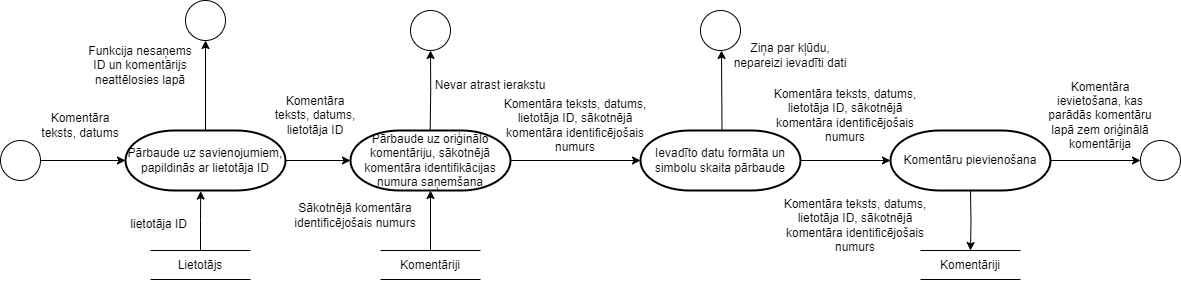
4.4.att. Pasūtījuma izpildes statusa rediģšanas datu plūsmu diagramma

1. **Oriģinālo komentāru pievienošana.** No sākuma sistēma paņems lietotāja ID no tabulas "Lietotājs", lai identificēt kam piederēs šīs komentārijs, pēc tam tiks pārbaudīts komentāra teksts, lai noskaidrotu, vai tas ir derīgs (skat. 4.5.att.), kā arī vai tas nepārsniedz norādīto simbolu skaitu. Kad pārbaude ir veikta un komentārs ir derīgs, sistēma saglabās šo komentāru datubāzē tabulā “Komentārijs”.



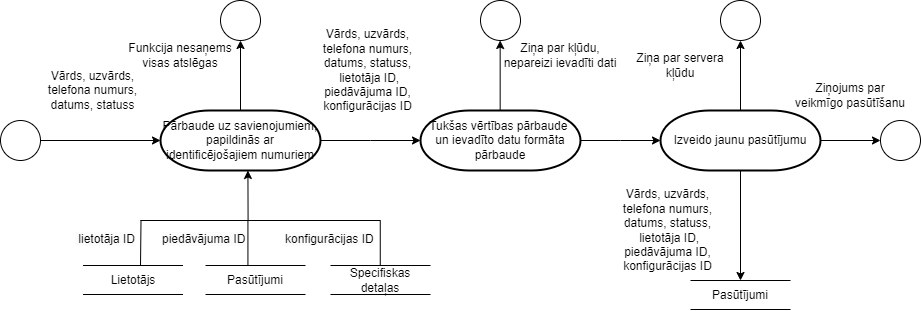
4.5.att. Komentāru pievienošanas datu plūsmu diagramma

1. **Atbilde uz oriģinālo komentāru.** Izmantojot šo funkciju, lietotājam ir iespēja ne tikai atstāt komentārus, bet arī atbildēt uz jau atstātajiem komentāriem. Nosūtot atbildi no tabulas “Lietotājs”, tiek ņemts lietotāja ID numurs (skat. 4.6. att.), lai noteiktu, kurš atstājis komentāru, ja šī vērtība netiek saņemta, komentārs netiks parādīts. Pēc tam no tabulas “Komentāriji” tiek ņemts sākotnējā komentāra identificējošais numurs, ja šī vērtība netiek iegūta, komentārs tiks uzskatīts par patstāvīgu komentāru, bet ne par atbildi. Pēc tam tiek pārbaudīts ievadītā ziņojuma formāts un garums, un, ja viss ir pareizi, komentārs tiek pievienots tabulā “Komentāriji”.



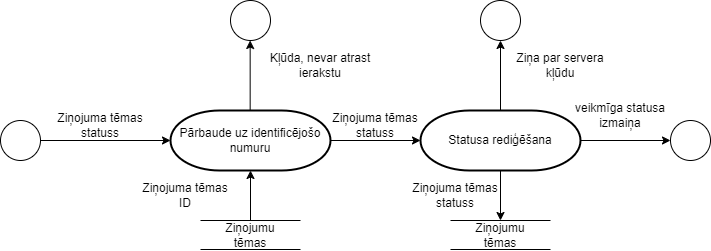
4.6.att. Atbildes uz oriģinālo komentāriju pievienošanas datu plūsmu diagramma

1. **Pasūtījumu veikšana.** Ja lietotājs nolemj veikt pasūtījumu, vispirms ir jāievada šādi dati: vārds, uzvārds, tālruņa numurs, un dati: datums un statuss tiks aizpildīti automātiski (skat. 4.7. att.). Pasūtījuma informācijai nepieciešami arī dati par pašu transportlīdzekli, tāpēc identifikācijas numuri tiek ņemti no tabulām “Lietotājs”, “Pasūtījumi”, “Specifiskas detaļas”. Pēc tam tiek pārbaudīti ievadītie dati, ja trūkst kontaktinformācijas, tiek parādīta attiecīga kļūda, ja viss ir kārtībā, pasūtījums tiek veiksmīgi ievadīts tabulā “Pasūtījumi” un lietotājam tiek nosūtīts paziņojums.



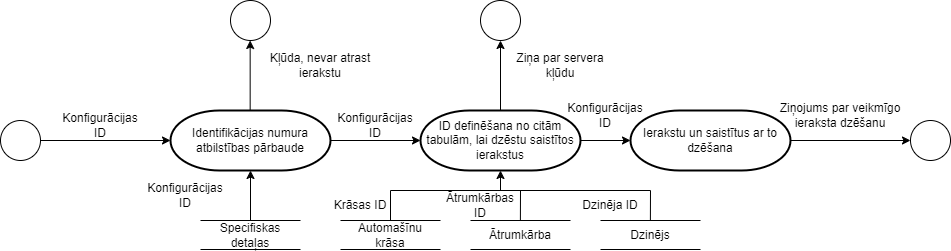
4.7.att. Pasūtījumu veikšanas datu plūsmu diagramma

1. **Ziņojumu tēmu statusu rediģēšana.** Lietotājam ir iespēja atsevišķā logā izvēlēties, par kurām tēmām viņš vēlas saņemt paziņojumus. Ir 3 galvenās tēmas (skat. 4.8. att.): “Forums” - paziņojumi, kad tiek atbildēts uz lietotāja komentāru sesijā, “Pasūtījumi” - kad tiek izpildīts pasūtījums, lietotājs tiek informēts par veiksmīgu pasūtījumu, “Profils” - kad tiek mainīta parole vai kad lietotājs reģistrējas. Vispirms tiek iegūts ID, lai identificētu lietotāju un pašu tēmu, un tad, ja serverī nav kļūdas, ieraksts tiek veiksmīgi ievadīts tabulā.



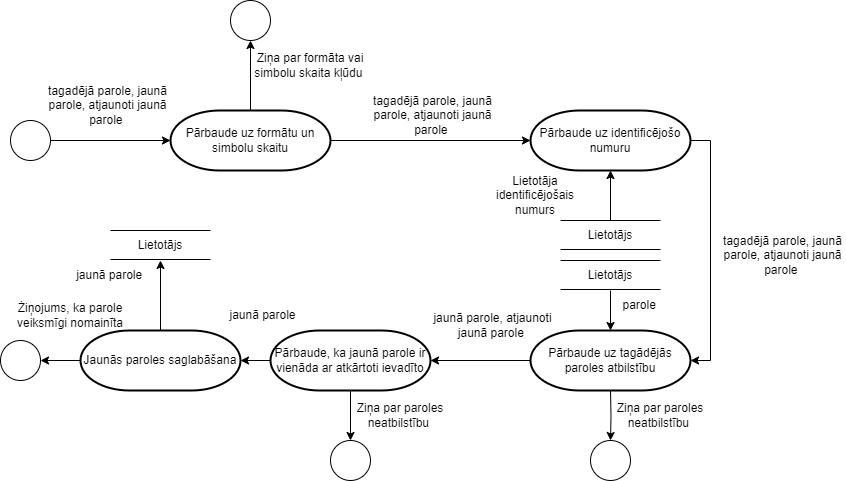
4.8.att. Ziņojumu tēmu statusu rediģēšanas datu plūsmu diagramma

1. **Komplektācijas dzēšana.** Apkalpojošs personāls var dzēst izvēlēto konfigurāciju, vispirms tiek automātiski ņemts komplektācijas ID (skat. 4.9. att.), lai noteiktu konkrēto konfigurāciju, pēc tam tiek iegūts ID no tabulām “Automašīnu krāsa”, “Ātrumkārba”, “Dzinējs”, lai dzēstu saistītos ar konfigurāciju ierakstus. Pēc tam, ja nav kļūdas, ieraksts tiek veiksmīgi dzēsts.



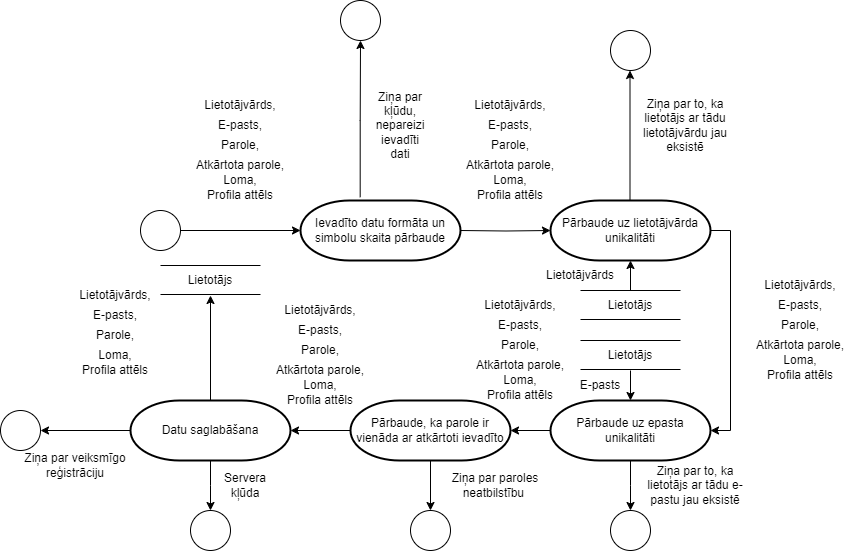
4.9.att. Komplektācijas dzēšanas datu plūsmu diagramma

1. **Paroles maiņa.** Ja lietotājs vēlas labot profilu, viņam būs iespēja mainīt paroli. Kad lietotājs ievadīs tagadējo paroli, jauno paroli un atkārtoto paroli, sistēma pārbaudīs, vai iepriekšējā parole ir ievadīta pareizi (skat. 4.10. att.) un vai jaunā parole atbilst atkārtoti ievadītajai. Informācija tiks ņemta no tabulas “Lietotājs”. Ja ievadītās paroles nesakrīt, vai tagadējā parole neatbilst, vai ja kāda no tām ir ievadīta nepareizi, sistēma parādīs kļūdas paziņojumu, ļaujot lietotājam labot ievadītos datus un nodrošinot drošu profila informācijas labošanu.



4.10.att. Paroles maiņas datu plūsmu diagramma

1. **Lietotāja reģistrācija.** Kad lietotājs reģistrējas, sistēma veic ievadīto datu pārbaudi (skat. 14.11. att.), lai nodrošinātu to kvalitāti un drošību. Tas ietver lietotājvārda un e-pasta adreses unikāluma pārbaudi, kā arī paroles drošības novērtējumu. Lietotāja lomu un profila attēlu sistēma piešķirs automātiski. Ja kāds no ievadītajiem datiem neatbilst prasībām vai nav derīgs, sistēma uzrādīs attiecīgu kļūdas paziņojumu, sniedzot norādījumus, kādēļ reģistrācija nevar tikt pabeigta. Ja visi dati ir derīgi, sistēma saglabā lietotāja informāciju datubāzē tabulā “Lietotājs”, nodrošinot drošu un uzticamu reģistrācijas procesu.



4.11.att. Lietotāja reģistrācijas datu plūsmu diagramma

# **DATU STRUKTŪRU APRAKSTS**

Datu bāze sastāv no 11 tabulām, kas satur sevī informāciju par lietotāju, komentāriem, piedāvājumiem, piedāvājuma informāciju, pasūtījumiem, krāsām un ziņām.

1. Tabula "**user**" glabā datus par lietotāju.
2. Tabula "**comments**" glabā datus par komentāriem.
3. Tabula "**offers**" glabā datus par piedāvājumiem.
4. Tabula "**offersinfo**" glabā datus par piedāvājuma detalizētu informāciju.
5. Tabula "**specific\_details**" starptabula, kas glabā ārējās atslēgas no citām tabulām, lai veidotu dažādus komplektācija variantus, kas vienam un tam pašam modelim var atšķirties.
6. Tabula "**transmission**" glabā datus par komplektācijas transmisiju.
7. Tabula "**engine**" glabā datus par komplektācijas dzinēju.
8. Tabula "**car\_colors**" glabā datus par komplektācijas krāsu.
9. Tabula "**order**" glabā datus par pasūtījumiem.
10. Tabula "**notifications**" glabā datus par pašu ziņojumu un saturu.
11. Tabula "**notification\_topics**" glabā datus par ziņojumu tēmām un kādas no tiem parādās lietotājam.

Tabula “**user**” ir saistīta ar tabulu “**comments**”, “**order**” un “**notification\_topics**” caur lauku userID.

5.1. tabula

Tabulas **“user”** struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| **1** | userID | int | 10 | lietotāja identifikatora kods, primārā atslēga; |
| **2** | password | varchar | 255 | lietotāja parole; |
| **3** | username | varchar | 20 | lietotājvārds; |
| **4** | roleID | int | 10 | lietotāja loma\*; |
| **5** | email | varchar | 40 | lietotāja ē-pasts; |
| **6** | rules\_accepted | tinyint | 1 | identifikators, vai lietotājs ir pieņēmis foruma noteikumus; |
| **7** | picture | blob | - | lietotāja profila attēls; |

\*Lietotājs ir klients vai apkalpojošais personāls (0 vai 1).

Tabula **“comments”** ir saistīta ar tabulu **“user”** caur lauku userID.

5.2. tabula

Tabulas **“comments”** struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| **1** | commentID | int | 10 | komentārija identifikatora kods, primārā atslēga; |
| **2** | date | date | - | komentārija datums; |
| **3** | comment | varchar | 700 | komentārijs, kuru ieraksta lietotājs; |
| **4** | userID | int | 10 | ārēja atslēga, kas norāda uz lietotāju, kurš atstāja komentāru; |
| **5** | parent\_comment\_id | int | 10 | saite uz vecāku komentāru, ja tā ir atbilde; |

\*Saite uz vecāku komentāru ir lauks datubāzē, kas norāda uz komentāru, uz kuru konkrētais komentārs atbild.

Tabula **“offers”** ir saistīta ar tabulām **“offersinfo”** un **“order”** caur lauku offerID**.**

5.3. tabula

Tabulas **“offers”** struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| **1** | offerID | int | 10 | piedāvājuma identifikatora kods, primārā atslēga; |
| **2** | type | varchar | 20 | automašīnas modelis; |
| **3** | manufacture | varchar | 20 | automašīnas marka; |

Tabula **“offersinfo”** ir saistīta ar tabulu “**offers**” caur lauku offersID un “**specific\_details**” caur lauku offersInfoID.

5.4. tabula

Tabulas **“offersinfo”** struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| **1** | offerInfoID | int | 10 | Piedāvājuma informācijas identifikatora kods, primārā atslēga; |
| **2** | price | float | - | automašīnas cena; |
| **3** | yearOfManufacture | int | 10 | automašīnas izdošanas gads; |
| **4** | weight | float | - | automašīnas svars; |
| **5** | body\_type | varchar | 20 | automašīnas virsbūves tips; |
| **6** | offersID | int | 10 | ārēja atslēga, kas norāda uz piedāvājumu; |

5.4. tabulas turpinājums

Tabula “**specific\_details**” ir saistīta ar tabulām: “**offersinfo**” caur lauku offersInfoID, “**order**” caur lauku detailsID, "**car\_colors**" caur lauku colorID, “**transmission**” caur lauku transmissionID, “**engine**” caur lauku engineID.

5.5. tabula

Tabulas “**specific\_details**” struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| **1** | detailsID | int | 10 | Konfigurācijas/komplektācijas identifikatora kods, primārā atslēga; |
| **2** | colorID | int | 10 | ārēja atslēga, kas norāda uz krāsu; |
| **3** | offersInfoID | int | 10 | ārēja atslēga, kas norāda uz piedāvājumu (automašīnu, konkrēto modeli); |
| **4** | transmissionID | int | 10 | ārēja atslēga, kas norāda uz transmisiju; |
| **5** | engineID | int | 10 | ārēja atslēga, kas norāda uz dzinēju; |
| **6** | active\_status | int | 10 | statuss, lai lietotājam netiktu rādīti piedāvājumi, kas ir atzīmēti kā neaktīvi; |
| **7** | created\_at | timestamp | - | ārēja atslēga, kas norāda uz piedāvājumu; |

Tabula **“engine”** ir saistīta ar tabulu **“specific\_details”** caur lauku engineID.

5.6. tabula

Tabulas “**engine**” struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| **1** | engineID | int | 10 | dzinēja identifikatora kods, primārā atslēga; |
| **2** | engine\_type | varchar | 50 | dzinēja tips, kas pieder konfigurācijai; |
| **3** | engine\_price | float | - | dzinēja cena, kas pieder konfigurācijai; |

5.6. tabulas turpinājums

Tabula **“transmission”** ir saistīta ar tabulu **“specific\_details”** caur lauku transmissionID**.**

5.7. tabula

Tabulas “**transmission**” struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| **1** | transmissionID | int | 10 | transmisijas identifikatora kods, primārā atslēga; |
| **2** | transmission\_type | varchar | 20 | transmisijas tips, kas pieder konfigurācijai; |
| **3** | transmission\_price | float | - | transmisijas cena, kas pieder konfigurācijai; |

Tabula **“car\_colors”** ir saistīta ar tabulu **“specific\_details”** caur lauku colorID**.**

5.8. tabula

Tabulas “**car\_colors**” struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| **1** | colorID | int | 10 | krāsas identifikatora kods, primārā atslēga; |
| **2** | color | varchar | 50 | krāsas nosaukums, kas pieder konfigurācijai; |
| **3** | image | blob | - | piedavājuma attēls, kas pieder konfigurācijai; |
| **4** | color\_price | float | - | krāsas cena, kas pieder konfigurācijai; |

Tabula **“order”** ir saistīta ar tabulu “**offers**” caur lauku orderOfferID, “**user**” caur lauku orderUserID un “**specific\_details**” caur lauku orderDetailsID.

5.9. tabula

Tabulas **“order”** struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| **1** | orderID | int | 10 | pasūtijuma identifikatora kods, primārā atslēga; |
| **2** | orderDate | date | - | pasūtijuma datums; |
| **3** | name | varchar | 20 | pasūtītāja vārds; |
| **4** | surname | varchar | 20 | pasūtītāja uzvārds; |
| **5** | telephone | varchar | 20 | pasūtītāja telefonas numurs; |
| **6** | status | varchar | 15 | pasūtijuma apstiprinājums; |
| **7** | orderOfferID | int | 10 | ārēja atslēga, kas norāda uz piedāvājumu; |
| **8** | orderUserID | int | 10 | ārēja atslēga, kas norāda uz lietotāju; |
| **9** | orderDetailsID | int | 10 | ārēja atslēga, kas norāda uz izvēlēto konfigurāciju; |

5.9. tabulas turpinājums

Tabula **“notifications”** ir saistīta ar tabulu **“notification\_topics”** caur lauku topic\_id**.**

5.10. tabula

Tabulas **“notifications”** struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| **1** | notification\_id | int | 10 | ziņojuma identifikatora kods, primārā atslēga; |
| **2** | message | text | - | ziņojuma teksts; |
| **3** | created\_at | timestamp | - | ziņojuma datums un laiks; |
| **4** | is\_read | tinyint | 1 | ziņojuma izlasīšanas statuss; |
| **5** | topic\_id | int | 10 | ārēja atslēga, kas norāda uz ziņojuma tēmu; |

\*Izlasīšanas statuss mainās, kad lietotājs atver logu ar ziņojumiem (“NULL” vai 1).

Tabula **“notification\_topics”** ir saistīta ar tabulu **“user”** caur lauku userID un ar tabulu **“notification\_topics”** caur lauku topic\_id.

5.11. tabula

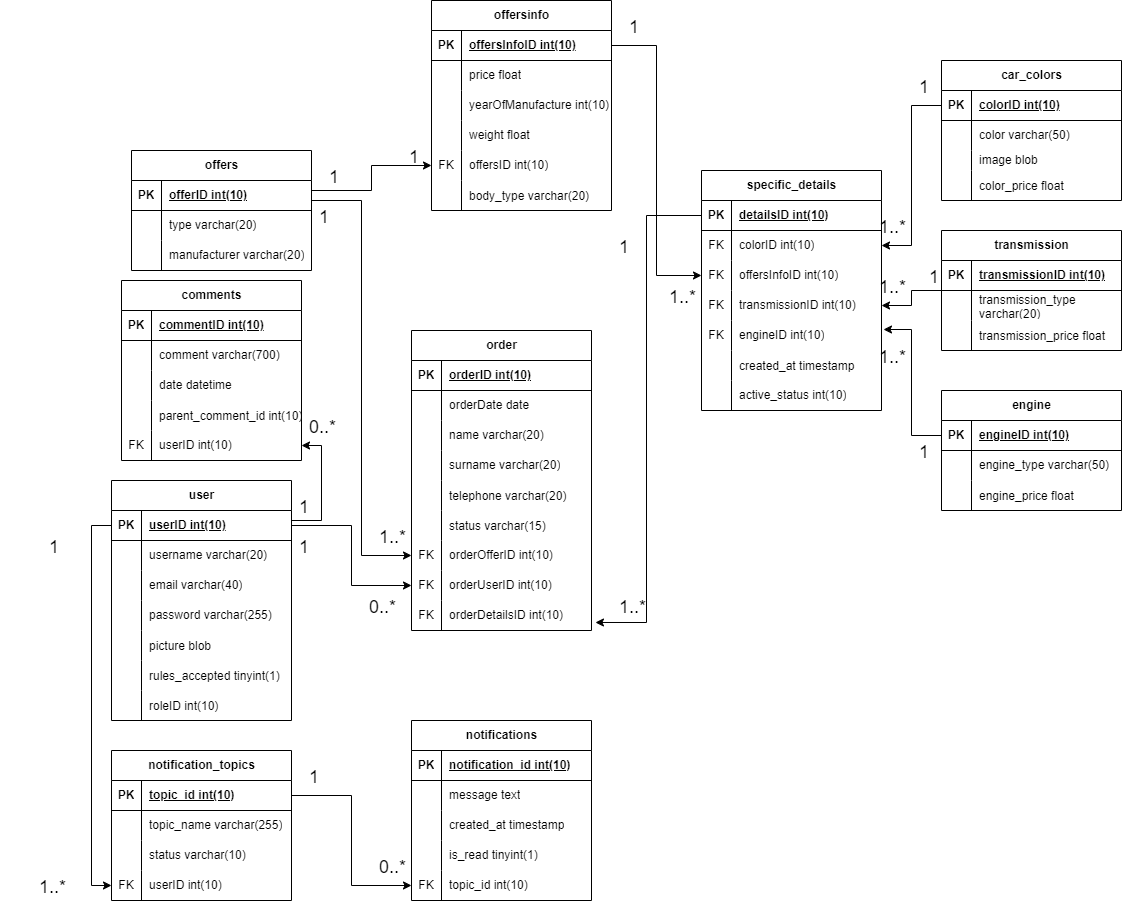
Tabulas **“notification\_topics”** struktūra

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nr.** | **Nosaukums** | **Tips** | **Garums** | **Piezīme** |
| **1** | topic\_id | int | 10 | ziņojuma tēmas identifikatora kods, primārā atslēga; |
| **2** | topic\_name | varchar | 255 | ziņojuma tēmas nosaukums; |
| **3** | status | varchar | 10 | ziņojuma tēmas attēlošanas statuss; |
| **4** | userID | int | 10 | ārēja atslēga, kas norāda uz lietotāju; |

5.11. tabulas turpinājums

\*Attelosanas statuss ietekmē to, vai lietotājs redzēs paziņojumus par konkrētām tēmām (“disable” vai “enable”).

Fiziskajā līmenī dati ir organizēti un glabājas, izmantojot tabulu saišu shēmu. Šī struktūra nodrošina efektīvu datu glabāšanu un atgriešanu, nodrošinot labu veiktspēju un optimizētu datu piekļuvi.



5.1.att. Datubāzes fiziskās struktūras shēma

# **LIETOTĀJA CEĻVEDIS**

## **Sistēmas prasības**

Tīmekļa vietne ir veidota tā, lai būtu pieejama un ērti lietojama, neprasot augstas sistēmas prasības. Lai palaistu šo tīmekļa vietni lokāli, izmantojot Laragon, ir jānodrošina, ka ierīce atbilst šādām minimālajām prasībām:

6.1. tabula

Tabula **“**Minimālās prasības tīmekļa vietnei**”**

|  |  |  |
| --- | --- | --- |
| **Nr.** | **Komponents** | **Minimālās prasības** |
| **1.** | Operētājsistēma | Windows 7 vai jaunāka, MacOS |
| **2.** | Laragon versija | Laragon 4.0 vai jaunāka |
| **3.** | Procesors | Intel Core i3 vai jaunāks |
|  | Atmiņa (RAM) | 1024MB vai vairāk |
| **4.** | Cietā diska vieta | Vismaz 500 MB brīvas vietas (vietnei un Laragon instalācijai) |

Lai palaistu vietni lokāli, jums vispirms jālejupielādē un jāuzstāda Laragon, kas ir ērti lietojama izstrādes vide, kas ietver Apache, MySQL, PHP un citus svarīgus komponentus, nepieciešamus vietnes lokālai palaišanai. Šeit ir dažas piemērotas versijas, kuras varat izmantot bez kļūdām:

* Laragon 4.0
* Laragon 4.1
* Laragon 4.2
* Laragon 5.0
* Laragon 5.1
* Laragon 5.2
* Laragon 6.0

Lai pilnībā izmantotu vietni un pārliecinātos, ka tā darbojas pareizi, ir nepieciešams arī viens no populārajiem mūsdienu pārlūkiem:

* Microsoft Edge (jaunākā versija)
* Mozilla Firefox (jaunākā versija)
* Google Chrome (jaunākā versija)
* Opera GX (jaunākā versija)

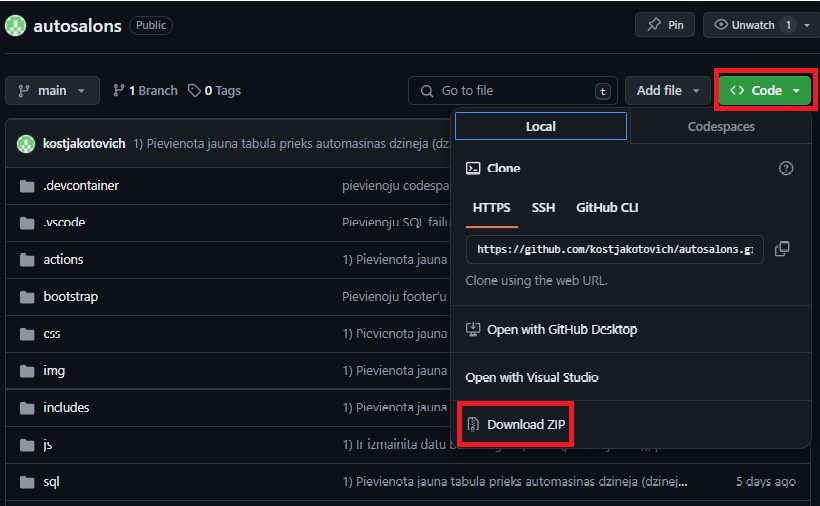
Papildus, lai varētu izpakot lejupielādēto projektu ZIP formātā, ir nepieciešams uzstādīt arhivatoru. Šeit ir daži piemēri populāru arhivatoru:

* 7-Zip
* WinRAR
* WinZip

## **Sistēmas instalācija un palaišana**

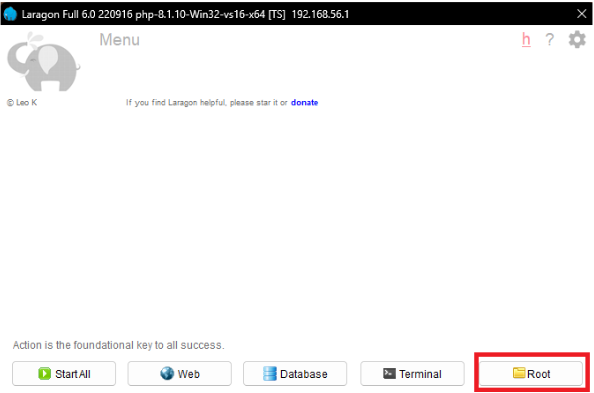
Lai vietne darbotos uz jūsu paša datora, ir nepieciešams veikt instalācijas un palaišanas procesu. Zemāk esošajā instrukcijā ir sniegta detalizēta informācija par katru soli, lai nodrošinātu veiksmīgu vietnes iedarbināšanu lokāli:

1. Vietnes failu lejupielāde no GitHub:
   1. Apmeklējiet GitHub repozitoriju.
   2. Noklikšķiniet uz zaļā pogas "Code" un izvēlieties "Download ZIP" (skat. 6.1. att.).



6.1.att. GitHub repozitorija

1. Laragon uzstādīšana:
   1. Lejupielādējiet un instalējiet Laragon no oficiālās mājas lapas.
   2. Palaidiet instalēto Laragon.
2. Vietnes failu novietošana Laragonā:
   1. Atveriet Laragon un noklikšķiniet uz pogas "Root" (skat. 6.2. att.).



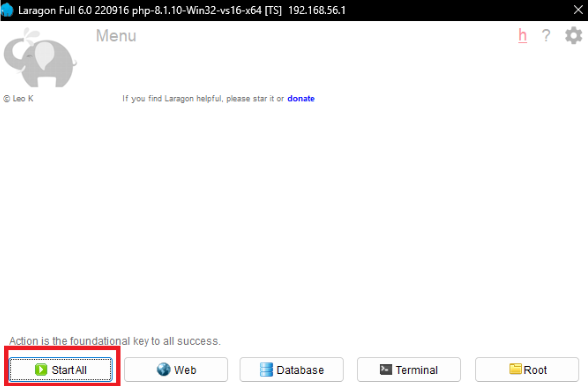
6.2.att. Laragon’a logs, mape ar projektu

* 1. Atarhivējiet iepriekš lejupielādēto ZIP failu mapē "www".
  2. Lejupielādētā mape tiks nosaukta kā "autosalons-main". Pārdēvējiet šo mapi uz "autosalons" (skat. 6.3. att.), jo GitHub pievieno "-main" galveno zara nosaukumam.



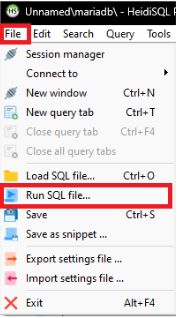
6.3.att. Mapes nosaukums

1. Servera un datu bāzes palaišana:
   1. Noklikšķiniet uz pogas "Start All" galvenajā Laragon logā, lai palaidinātu tīmekļa serveri un MySQL datu bāzi (skat. 6.4. att.).



6.4.att. Laragon’a logs, starta poga

1. Datu bāzes izveide:
   1. Pēc servera palaišanas atveriet datu bāzi, noklikšķinot uz pogas "Database" galvenajā Laragon logā.
   2. Pēc datu bāzes atvēršanas noklikšķiniet uz "File" kreisajā augšējā stūrī un izvēlieties "Run SQL file" (skat. 6.5. att.).
   3. Atlasiet failu "database.sql", kas atrodas lejupielādētā repozitorija mapē "sql".
   4. Šis fails izveidos datu bāzi ar visām nepieciešamajām tabulām.



6.5.att. Logs ar datubāzes pievienošanu

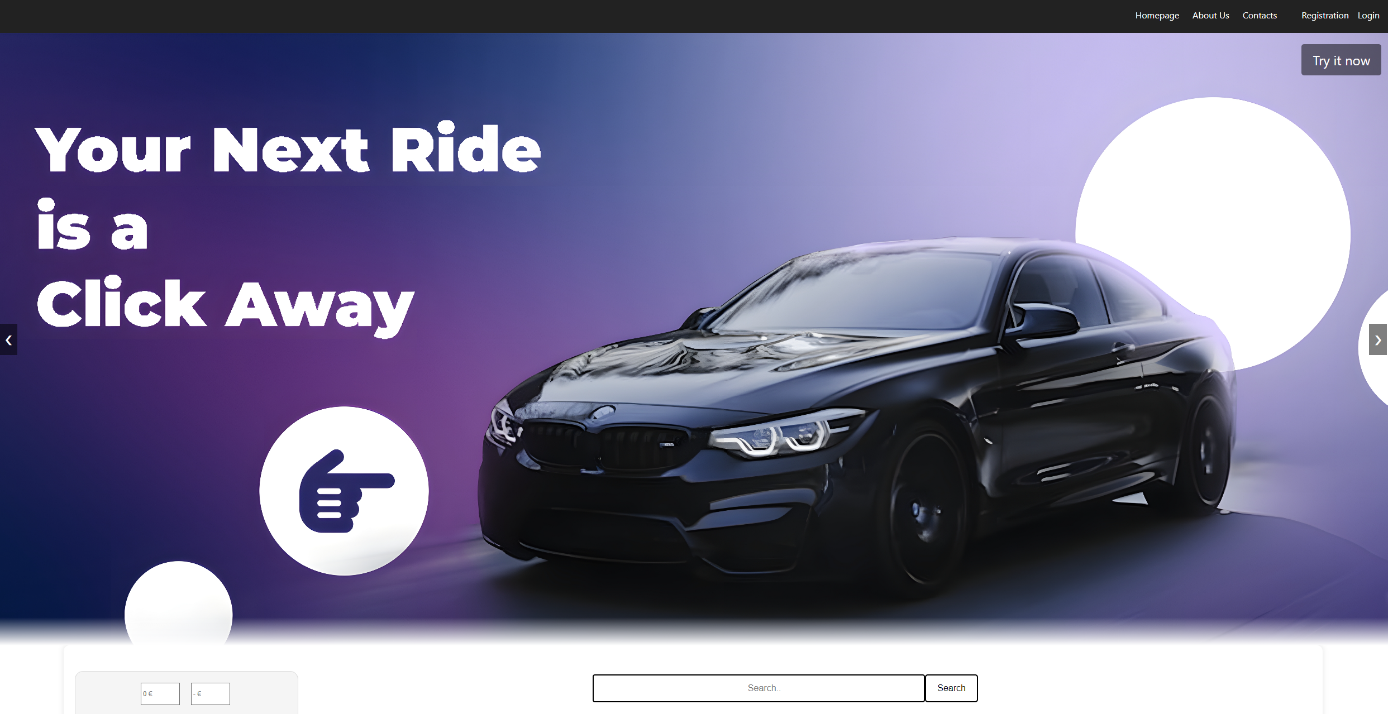
1. Vietnes palaišana:
   1. Atveriet jebkuru interneta pārlūku datorā.
   2. Adreses joslā ievadiet: http://localhost/autosalons/index.php.
   3. Nospiediet Enter, lai atvērtu vietni.

Pēc šo soļu izpildes, vietne būs gatava lietošanai lokālajā vidē.

## **Programmas apraksts**

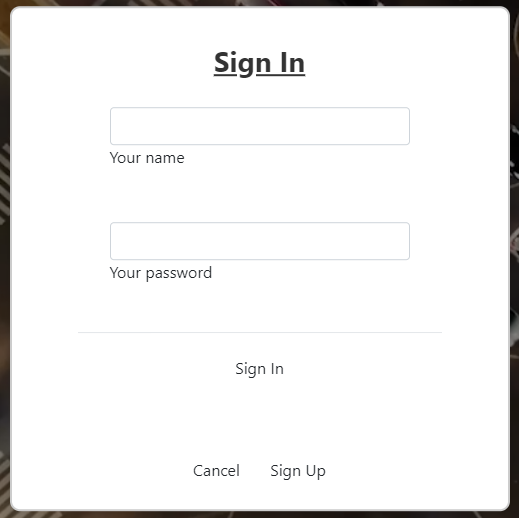
Vispirms jāatver tīmekļa vietnes sākuma lapu: http://localhost/autosalons/index.php, (skat. 6.6. att.).

* + 1. ***Reģistrācija vai reģistrētu lietotāju autorizācija***



6.6.att. Mājaslapas skats

Ievadam visus nepieciešamos laukus un noklikšķinam uz pogas “Sign In” (skat. 6.7. att.), ja nav reģistrēta neviena profila, nospiežam pogu “Sign Up”.



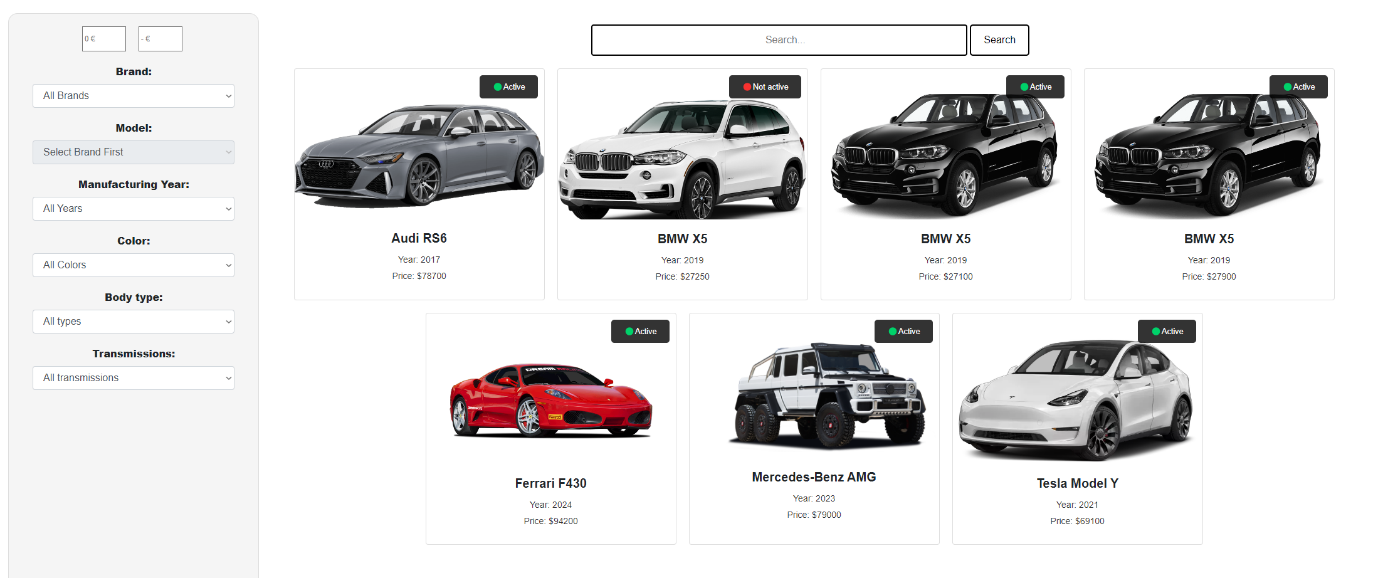
6.7.att. Autorizācijas forma

Reģistrācijas lapā ir jāaizpilda vairāk lauku un jāņem vērā formāts un rakstzīmju skaits (skat. 6.8. att.).



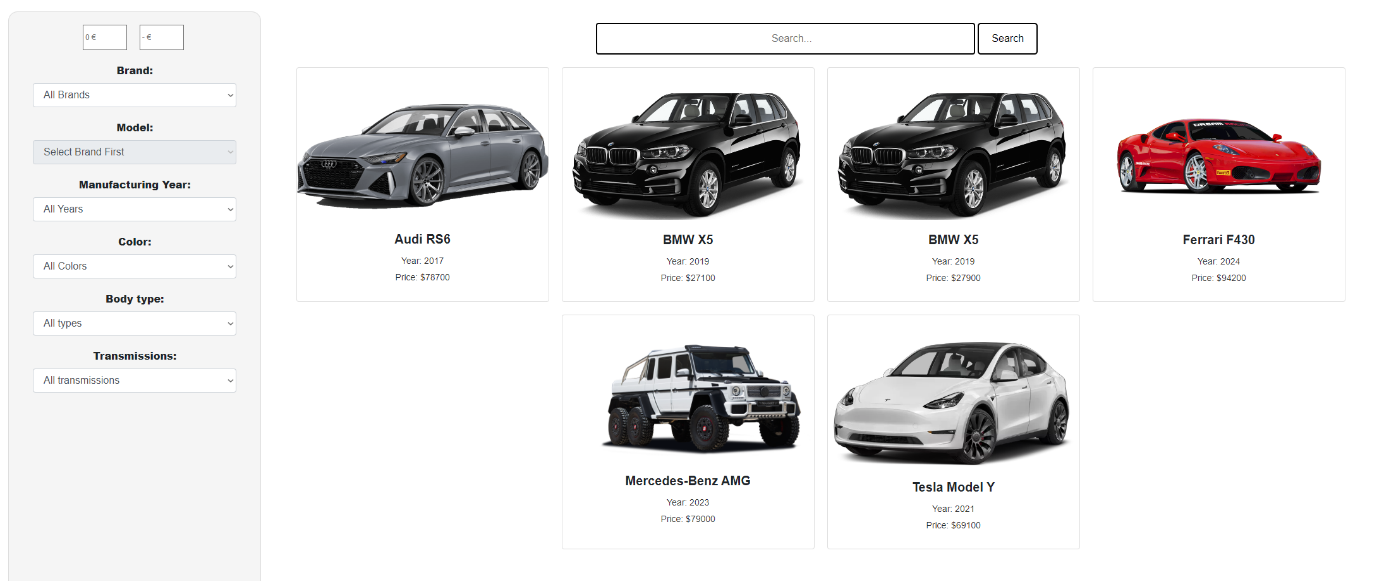
6.8.att. Reģistrācijas forma

Ja ievadāt visus datus, tiksiet novirzīts uz sākumlapu, taču atkarībā no ievadītajiem datiem tā tiks rādīta atšķirīgi. Ja pierakstīsieties apkalpojošā personāla kontā, uz kartītēm ar piedāvājumiem redzēsiet statusus (skat. 6.9. att.), un šie statusi noteiks, vai piedāvājumi tiks rādīti lietotājam vai ne.



6.9.att. Piedāvājumu atspoguļošanas lapa personālam

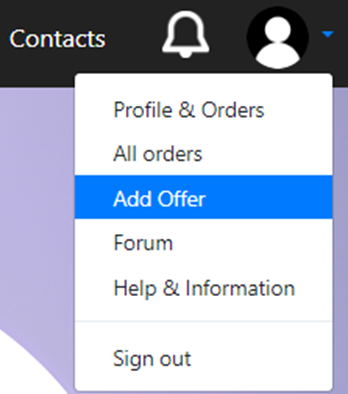
Ja pierakstīsieties kā parasts lietotājs, statusi un slēptie piedāvājumi nebūs redzami (skat. 6.10. att.).



6.10.att. Piedāvājumu atspoguļošanas lapa letotājam

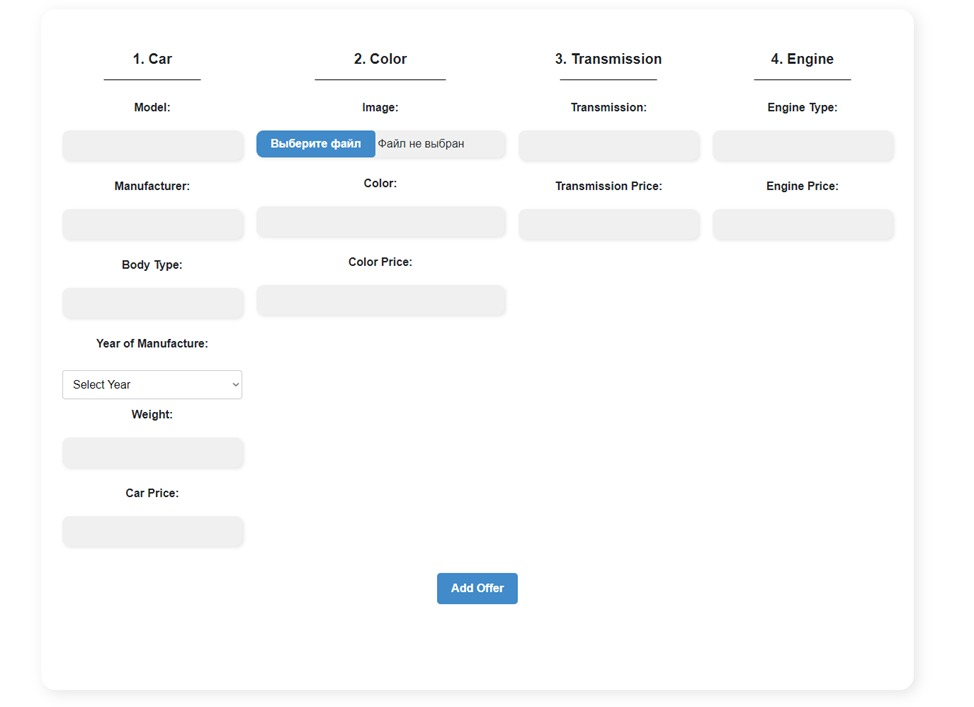
* + 1. ***Jaunu transportlīdzekļu/piedāvājumu pievienošana***

Administratora augšējā joslā varam noklikšķināt uz profila attēla, pēc tam parādīsies saraksts, kurā izvēlamies “Add Offer” (skat. 6.11. att.), un parādīsies lapa, kurā jāaizpilda visi lauki.



6.11.att. Navigācijas joslā, nolaižamajs saraksts

Varam redzēt 4 kolonnas, pirmajā ir vispārīga informācija par automobili, bet pārējās 3 ir saistītas ar konfigurāciju, kas vienam un tam pašam modelim var atšķirties (skat. 6.12. att.).



6.12.att. Piedāvājumu pievienošanas forma

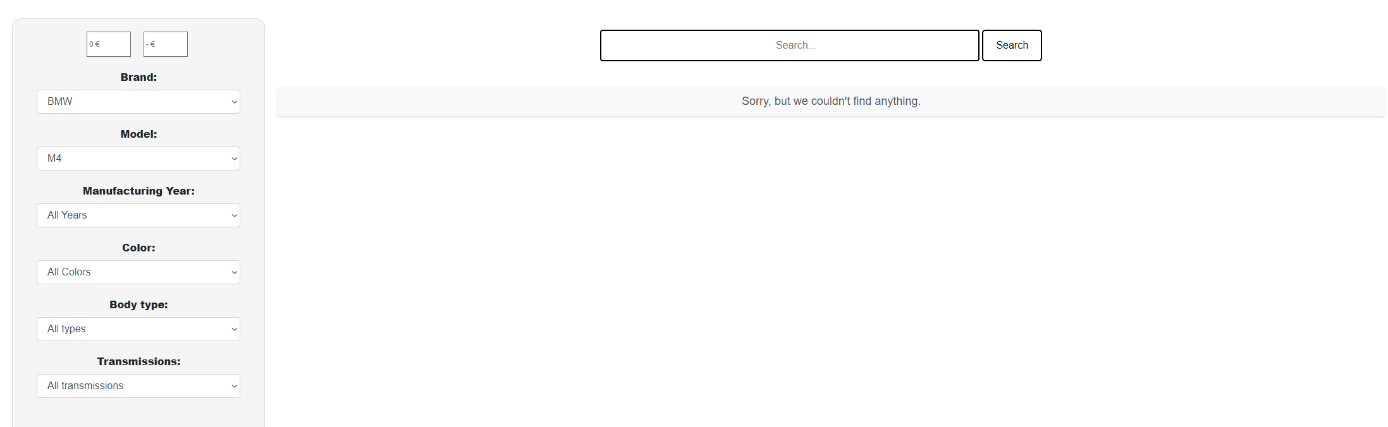
Laukā, kur ir nepieciešams izvēlēties automašīnas ražošanas gadu, ērtības dēļ jau ir sagatavoti varianti (skat. 6.13. att.), jauni gadi tiks pievienoti automātiski.



6.13.att. Ražošanas gada izvēles lauks

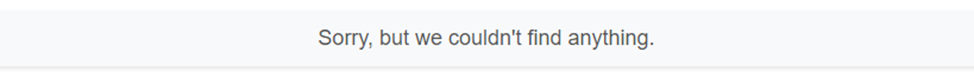
* + 1. ***Piedāvājumu meklēšana un filtrēšana***

Galvenajā lapā mēs varam redzēt meklēšanas joslu un filtrus blakus (skat. 6.14. att.), tie darbojas kopā.



6.14.att. Meklēšanas josla un filtri

Ja netiks atrasti atbilstoši piedāvājumi, tiks parādīts uzraksts: “Sorry, but we couldn't find anything.” (skat. 6.15. att.), bet, ja būs sakritības, tiks parādīti atbilstoši automobiļi.



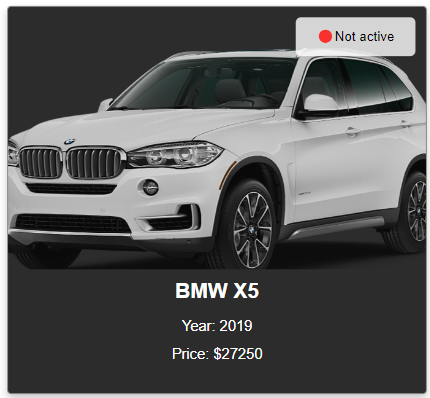
6.15.att. Izejas virkne, ja nekas nav atrasts

Ja mēs ievadām gan meklēšanas joslā, gan filtros, lai atlasītu parametrus, tad tiks ņemti vērā visi atlasītie vai ievadītie lauki (skat. 6.16. att.).



6.16.att. Piedāvājumu sadaļa

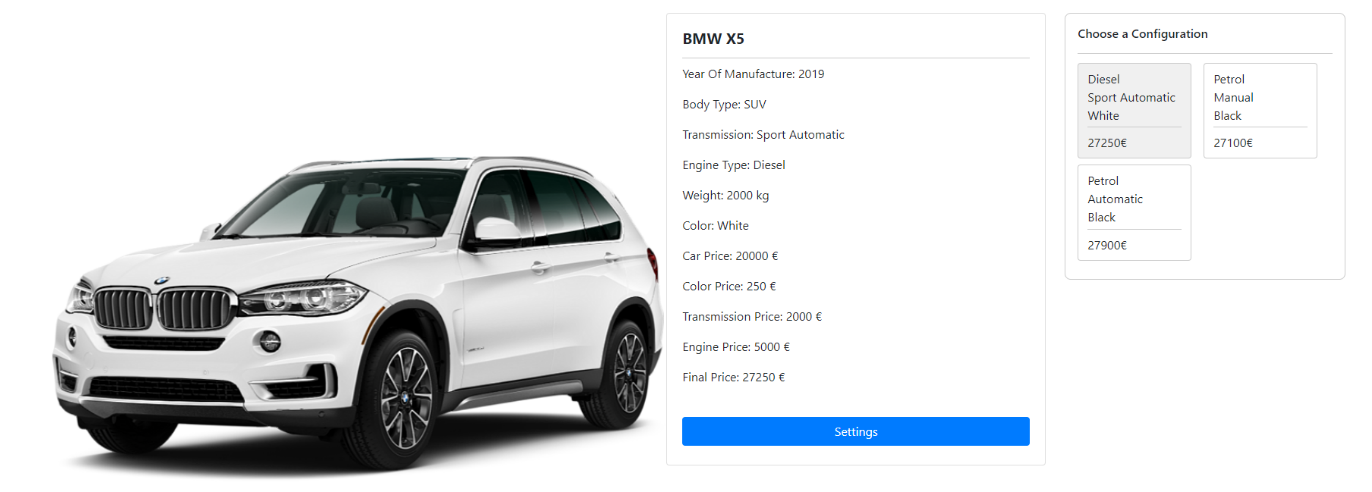
Lai apskatītu informāciju par piedāvājumu un ar to saistīto funkcionalitāti, ir jānovieto kursors virs attiecīgās kartiņas un jānoklikšķina uz to (skat. 6.17. att.).



6.17.att. Piedāvājumu kartiņa

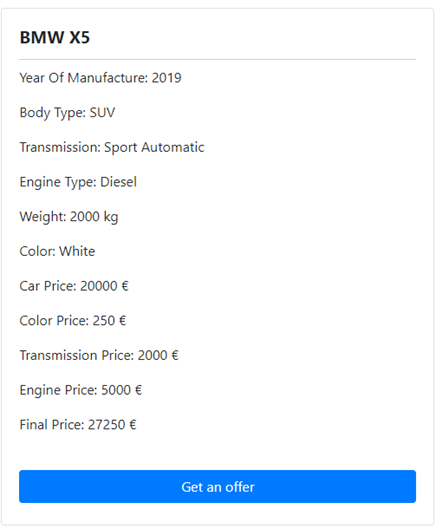
* + 1. ***Jaunu transportlīdzekļu/piedāvājumu pievienošana***

Šajā lapā mēs varam redzēt visu detalizēto informāciju par izvēlēto piedāvājumu (skat. 6.18. att.).



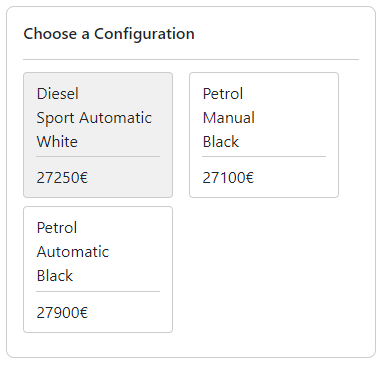
6.18.att. Piedāvājuma lapa no apkalpojošā personāla skata

Atšķirībā no administratora lietotājam nav pogas “Settings” (skat. 6.19. att.).



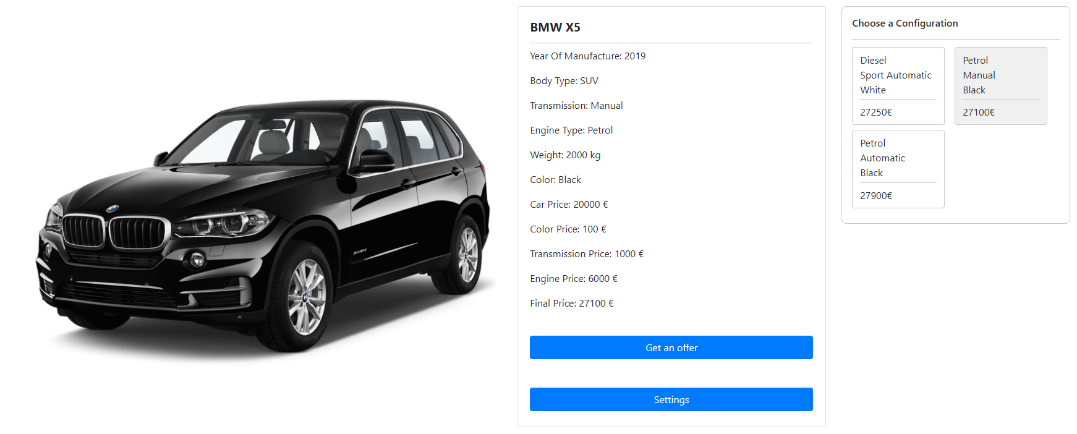
6.19.att. Piedāvājuma detalizētā informācija

Atsevišķā logā tiek parādītas visas konfigurācijas, kas ir saistītas ar šo modeli, un izvēlētā konfigurācija ir pelēkā krāsā (skat. 6.20. att.).



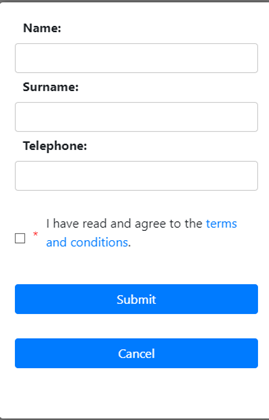
6.20.att. Konfigurāciju izvēles logs

Izvēloties citu konfigurāciju, noklikšķinot uz to (skat. 6.21. att.), mēs tiksim novirzīti uz tā paša modeļa lapu, bet ar citu konfigurāciju.



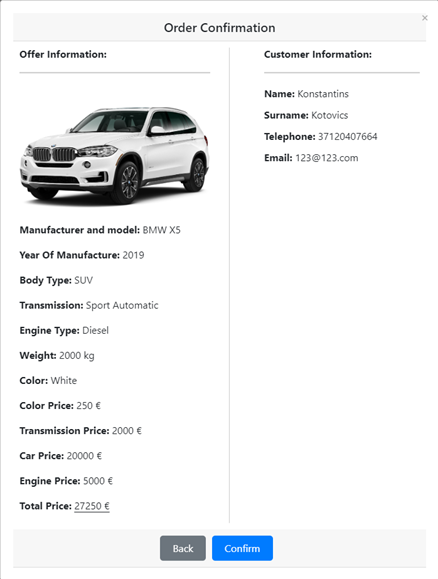
6.21.att. Piedāvājuma lapa no lietotāja skata

Lai pasūtītu izvēlēto automašīnu, nospiežam pogu “Get an offer”. Pēc tam aizpildam nepieciešamos laukus ar kontaktinformāciju un atzīmējam, ka piekrītam noteikumiem (skat. 6.22. att.).



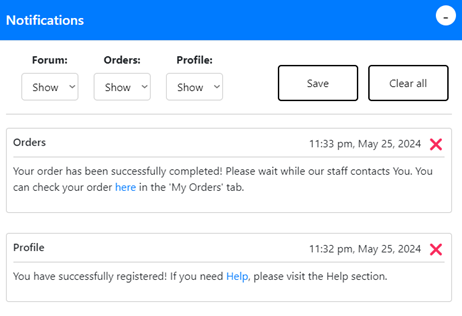
6.22.att. Pasūtīšanas forma

Pēc tam tiek parādīts apstiprinājuma logs ar visu informāciju par automašīnu un lietotāju (skat. 6.23. att.), kas veicis pasūtījumu.



6.23.att. Apstiprinājuma logs

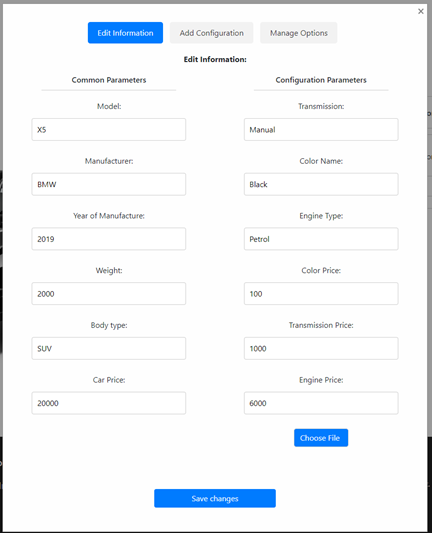
Ja pasūtījums ir veiksmīgi nosūtīts, lietotājs saņems paziņojumu (skat. 6.24. att.).



6.24.att. Ziņojumu logs

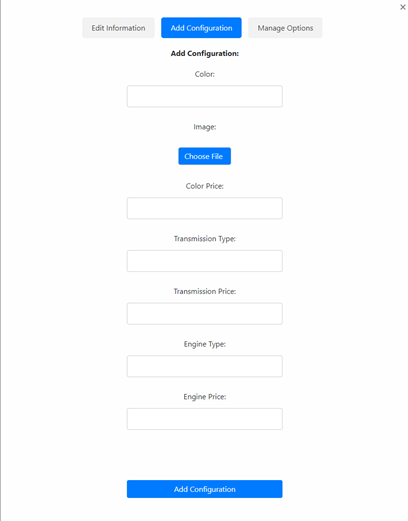
* + 1. ***Transportlīdzekļa konfigurācijas dzēšana vai informācijas rediģēšana***

Administratoram piedāvājuma lapā ir poga ar parametriem, noklikšķinot uz tās, parādās logs, kura pirmajā sadaļā var rediģēt informāciju par pašu modeli un konkrēto konfigurāciju (skat. 6.25. att.).



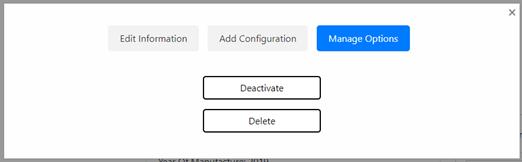
6.25.att. Informācijas rediģēšanas sadaļa

Otrā sadaļa ļauj pievienot jaunas konfigurācijas esošajam modelim (skat. 6.26. att.).



6.26.att. Konfigurāciju pievienošanas sadaļa

Un trešajā sadaļā ir iespēja izdzēst konkrēto konfigurāciju vai pārveidot to par neaktīvu (skat. 6.27. att.), tad tā netiks rādīta lietotājam.



6.27.att. Kontroles pogas sadaļa

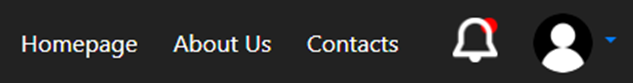
* + 1. ***Ziņojumu saņemšana un to dzēšana***

Katram reģistrētajam lietotājam ir paziņojumu attēls navigācijas joslā, ja nav nelasītu paziņojumu, tā būs neaktīva (skat. 6.28. att.).



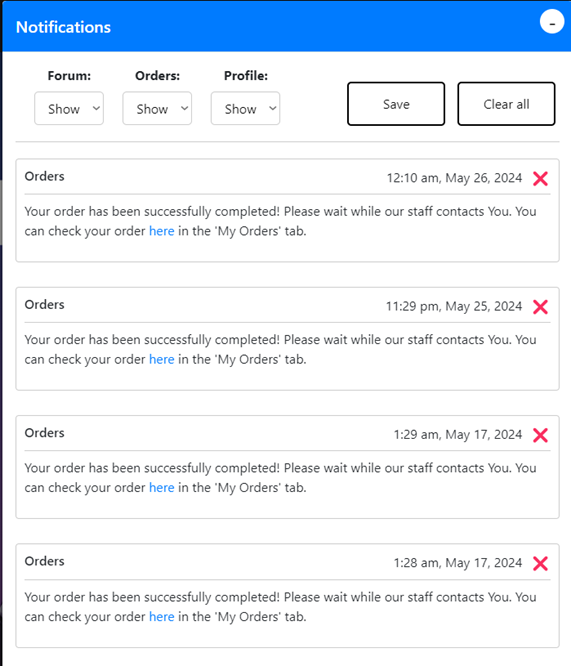
6.28.att. Navigācijas josla ar neaktīvo attēlu

Ja ir jauni paziņojumi, kurus lietotājs nav izlasījis, tā tiks mainīta uz aktīvu (skat. 6.29. att.).



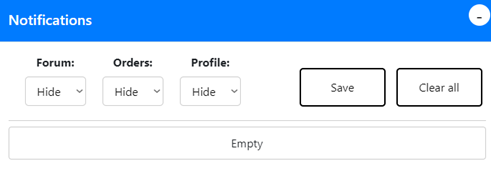
6.29.att. Navigācijas josla ar aktīvo attēlu

Noklikšķinot uz šī attēla, tiks atvērts logs ar saņemtajiem paziņojumiem (skat. 6.30. att.). Visus paziņojumus var dzēst, noklikšķinot uz krustiņa pret tiem, vai arī var dzēst tos visus uzreiz, noklikšķinot uz pogas “Clear all”. Ziņojumi ir sadalīti 3 tēmās: “Forum”, “Orders”, “Profile”, katrs paziņojums ir saistīts ar attiecīgo tēmu, piemēram, ja tiek atbildēts uz lietotāja sesijā atstāto komentāru, tiks nosūtīts paziņojums ar tēmu “Forums”.



6.30.att. Ziņojumu logs

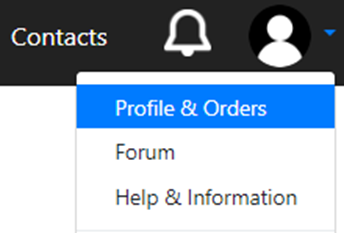
Lietotājs var paslēpt tēmas, kas viņu neinteresē, izvēloties iespēju “Hide” pie konkrētām tēmām (skat. 6.31. att.), un paziņojumi netiks rādīti.



6.31.att. Tukšs ziņojumu logs

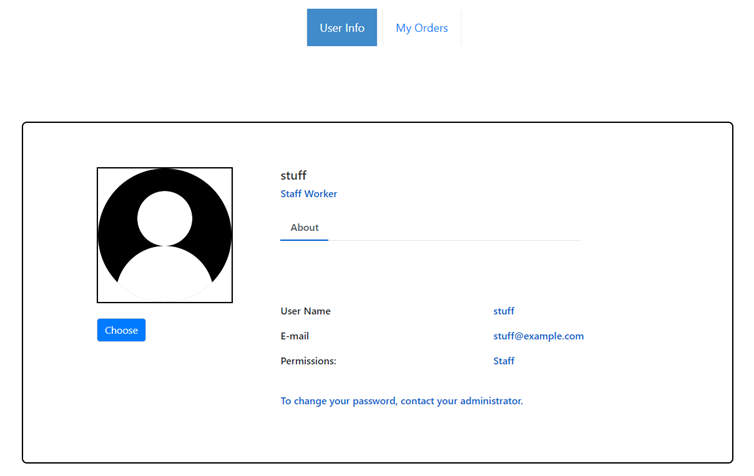
* + 1. ***Lietotāja lapa un tās rediģēšana***

Lai nokļūtu profila lapā, navigācijas joslā no nolaižamajā sarakstā jāizvēlas “Profile & Orders” (skat. 6.32. att.).



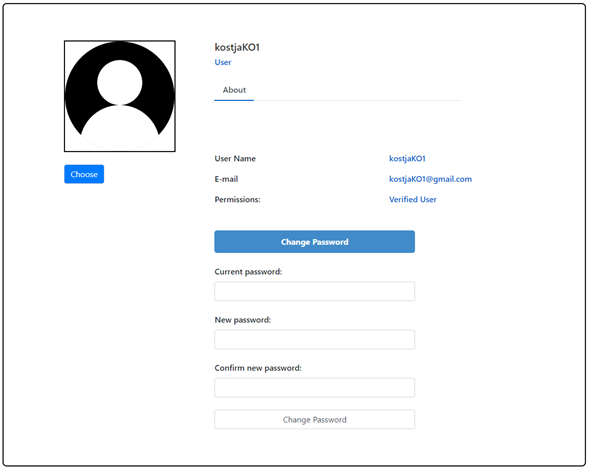
6.32.att. Navigācijas joslā, nolaižamajs saraksts

Lapā var mainīt profila attēlu (skat. 6.33. att.), bet paroli apkalpojošs personāls nevar mainīt patstāvīgi, to var darīt tikai administrators.



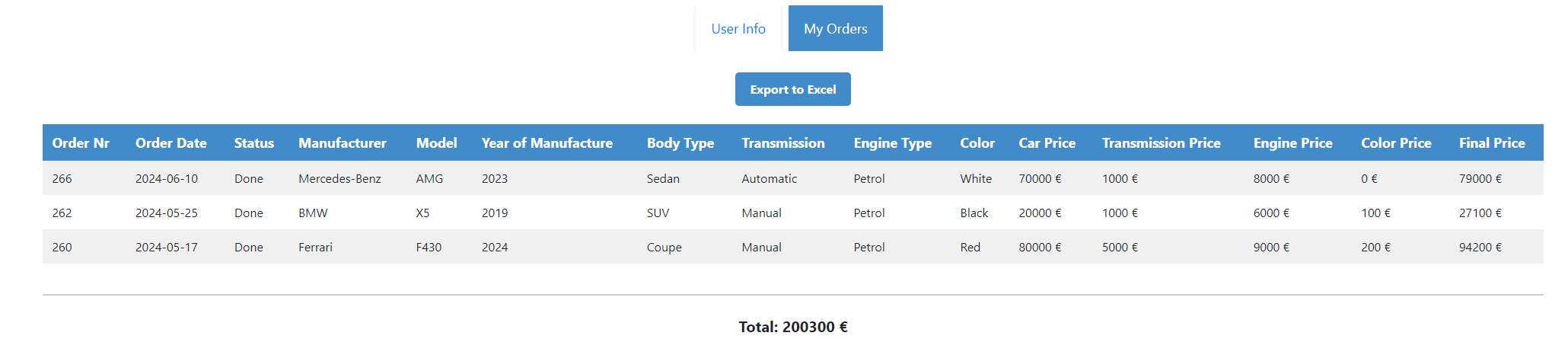
6.33.att. Personāla profila lapa

Atšķirībā no apkalpojošā personāla lietotāji var mainīt savu paroli (skat. 6.34. att.), ievadot pašreizējo paroli, jauno paroli un atkārtoti jauno paroli.



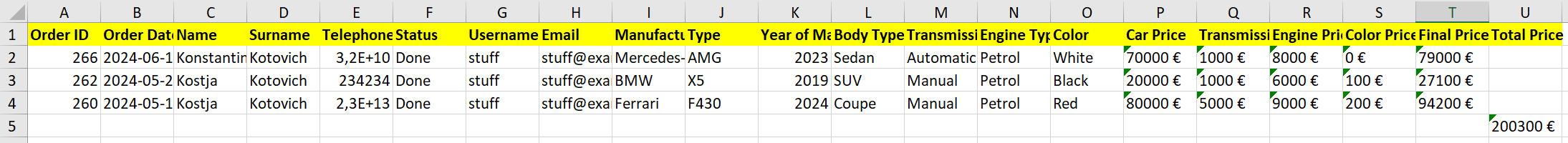
6.34.att. Lietotāja profila lapa

Izvēloties otro sadaļu “My Orders”, var apskatīt savu pasūtījumu vēsturi (skat. 6.35. att.).



6.35.att. Lietotāja pasūtījumu vēstures lapa

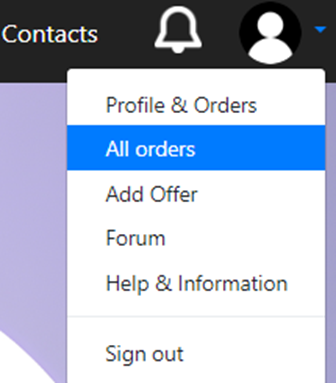
Tabulu ar pasūtījumiem var eksportēt Excel failā, noklikšķinot uz pogas “Export To Excel” (skat. 6.36. att.).



6.36.att. Lietotāja pasūtījumu vēstures Excel tabula

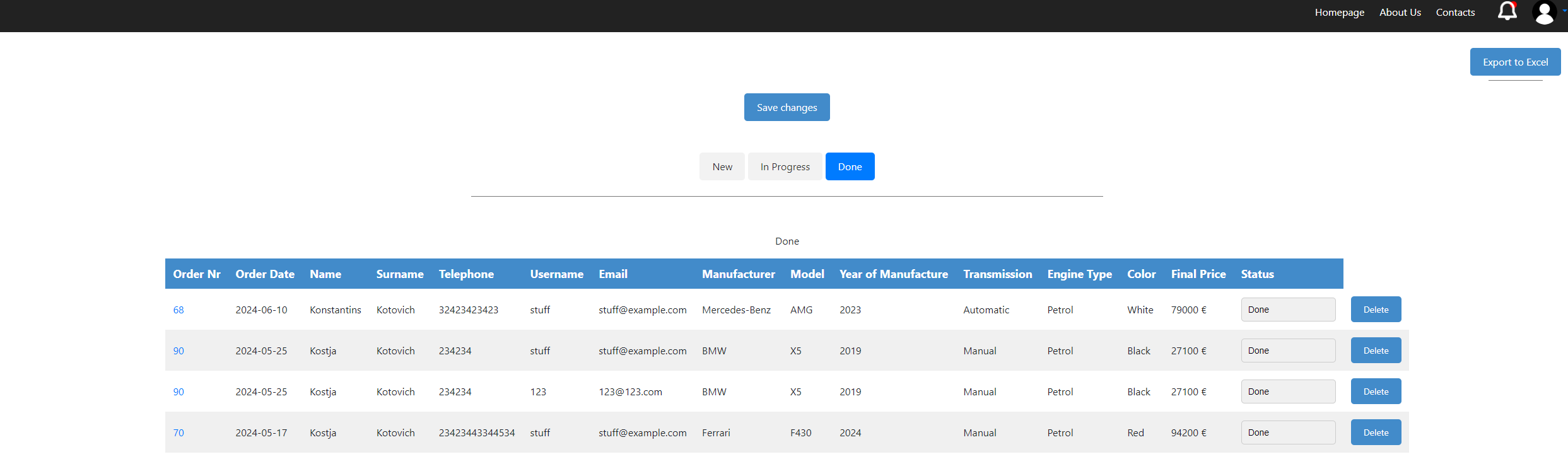
* + 1. ***Visu lietotāju pasūtījumu vēsture un izpildes statusa rediģēšana***

Lai apskatīt visu lietotāju pasūtījumu vēsturi, navigācijas joslā no nolaižamajā sarakstā jāizvēlas “All orders” (skat. 6.37. att.).



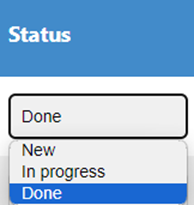
6.37.att. Navigācijas joslā nolaižamajs saraksts

Šī lapa ir paredzēta tikai apkalpojošajam personālam (skat. 6.38. att.).



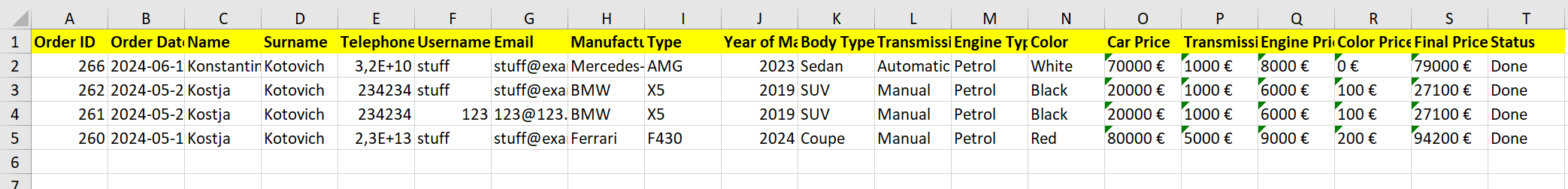
6.38.att. Visu lietotāju pasūtījumu vēstures lapa

Šajā lapā ir 3 sadaļas: “New”, “In progress”, “Done”. Tas nozīmē, ka pasūtījumi ir iedalīti 3 kategorijās, kurās var mainīt katra pasūtījuma statusu, un pēc pogas “Save changes” nospiešanas ieraksti tiek pārvietoti uz attiecīgajām tabulām (skat. 6.39. att.).



6.39.att. Statusa rediģēšanas forma

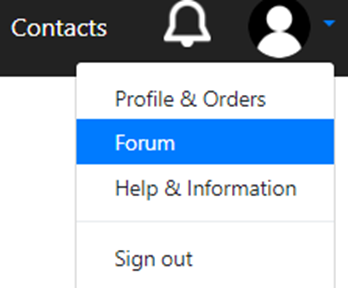
Šo tabulu var arī izvadīt Excel failā (skat. 6.40. att.), noklikšķinot uz pogas “Export to Excel” augšējā labajā stūrī.



6.40.att. Lietotāja pasūtījumu vēstures Excel tabula

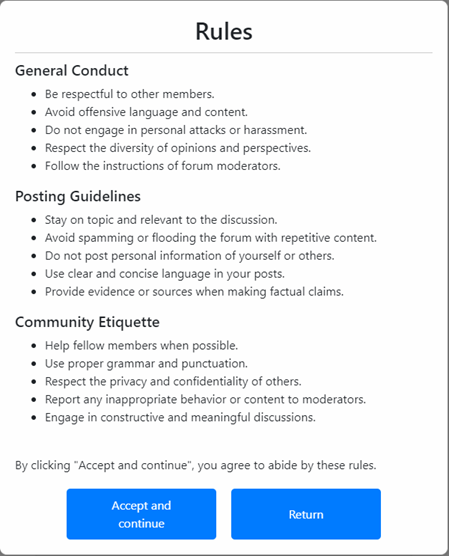
* + 1. ***Komentāru pievienošana un dzēšana***

Lai nokļūtu šajā lapā, navigācijas joslā izvēlieties “Forum” (skat. 6.41. att.).



6.41.att. Navigācijas joslā, nolaižamajs saraksts

Pirmo reizi atverot šo lapu, lietotājam tiks parādīti foruma noteikumi (skat. 6.42. att.), kuri ir jāpieņem, lai varētu lasīt un atstāt komentārus, tas ir jādara tikai vienu reizi, pēc tam šis logs vairs netiks rādīts.



6.42.att. Foruma noteikumi

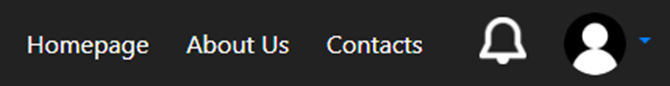
Autorizētiem lietotājiem ir iespēja publicēt komentārus līdz 700 rakstzīmēm vai atbildēt uz citiem komentāriem, kā arī iespēja dzēst savus komentārus (skat. 6.43. att.). Personāls var dzēst visus komentārus.



6.43.att. Komentāru lapa

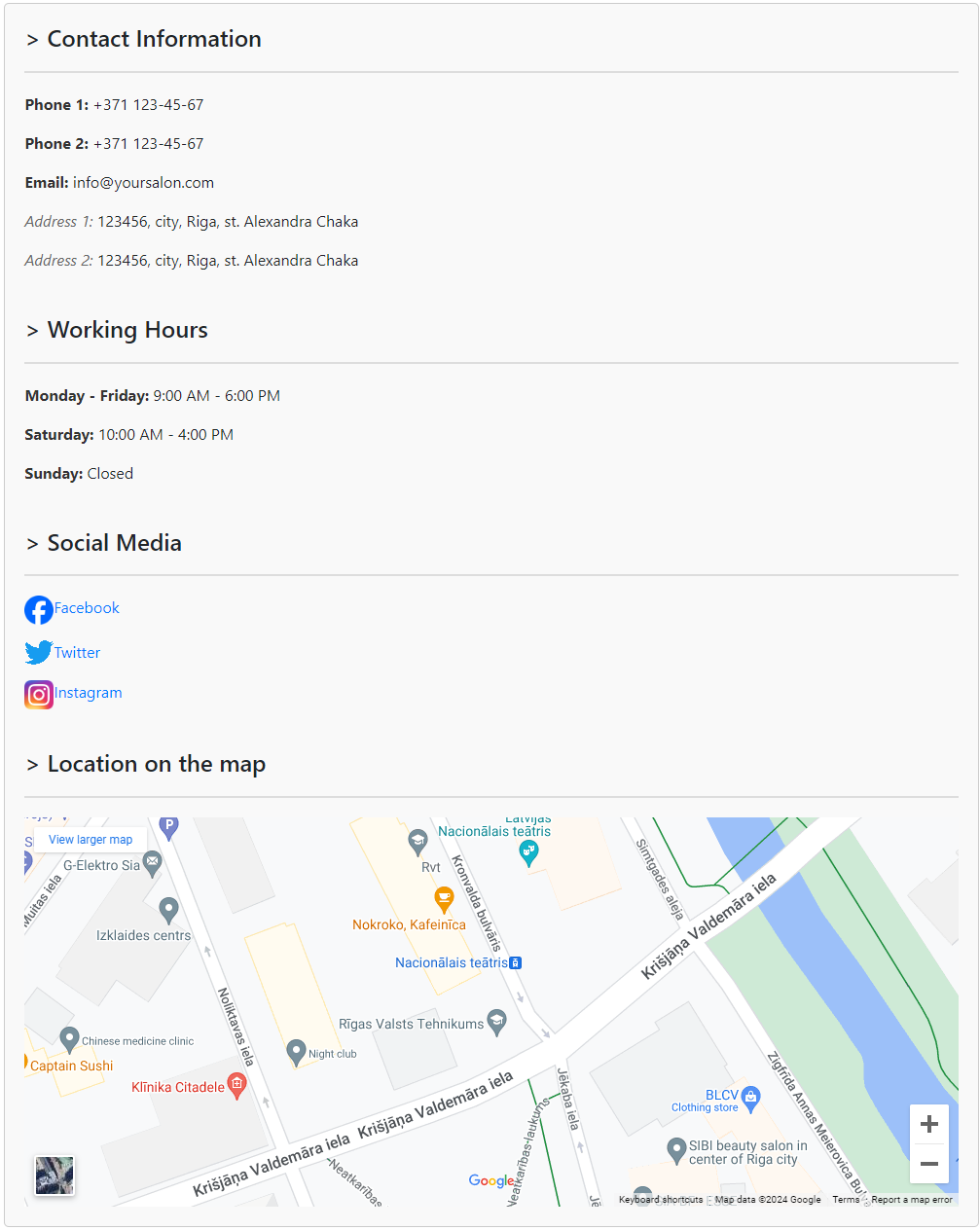
* + 1. ***Informācijas lapu apskate***

Lai skatītu kontaktinformāciju, navigācijas joslā noklikšķiniet uz pogas “About Us” (skat. 6.44. att.).



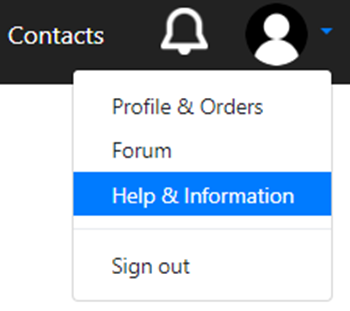
6.44.att. Navigācijas joslā

Šajā lapā ir saites uz pakalpojuma sociālajiem tīkliem, kā arī visa kontaktinformācija (skat. 6.45. att.).



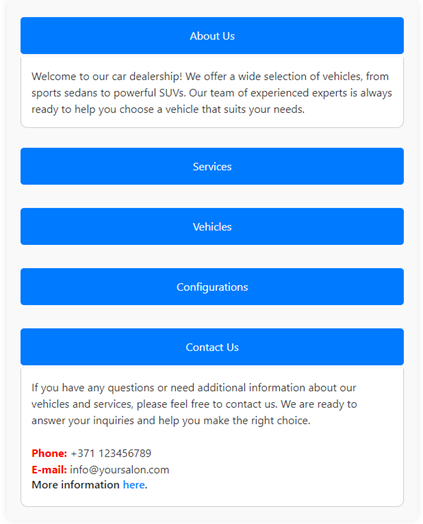
6.45.att. Kontaktinformācijas lapa

Navigācijas joslā ir arī punkts “Help & Information” (skat. 6.46. att.).



6.46.att. Navigācijas joslā, nolaižamajs saraksts

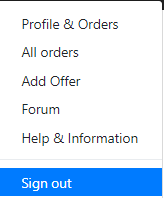
Šajā lapā ir arī neliela informācija par pakalpojumu (skat. 6.47. att.).



6.47.att. Informācijas lapa

* + 1. ***Izrakstīšanās no konta***

Izvēlamies opciju "Sign out" (skat. 6.48. att.), pēc kā sesija tiks pārtraukta.



6.48.att. Navigācijas joslā, nolaižamajs saraksts, izrakstīšanās no konta

## **Testa piemērs**

Šajā sadaļā ir sniegti piemēri, kas parāda, kā pārbaudīt sistēmas funkcionalitāti dažādās situācijās. Katram testam ir aprakstīti ievaddati un sagaidāmais rezultāts, kas palīdzēs saprast, kā sistēma reaģē uz konkrētiem ievaddatiem un kādi kļūdu ziņojumi tiek rādīti lietotājam. Tas palīdzēs nodrošināt, ka sistēma darbojas atbilstoši specifikācijām un lietotāju prasībām.

6.2. tabula

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nr.** | **Prasības numurs** | **Prasības nosaukums** | **Ievaddati/situācijas apraksts** | **Sagaidāmais rezultāts** | **Statuss** |
| 1. | 1. | Jauna lietotāja reģistrācija | Pareizi dati lietotāja vārds, lietotāja e-pasts, 1.parole, 2.parole | Ieraksts ir pievienots datubāzei, un lietotājam tiek nosūtīts paziņojums. | Pareizi |
| 2. | konstantins@gmail.com | Paziņojums, ka šis e-pasts jau ir aizņemts | Pareizi |
| 3. | 1. un 2. parole = k18t | Paziņojums, ka parole ir pārāk īsa – tai jābūt vismaz 6 simbolu garai | Pareizi |
| 4. | 11.2. | Lietotāja paroles maiņa | Pareizi dati spēkā esošā parole, jauna parole, atkārtoti jauna parole | Ieraksta rediģēšana datubāzē un paziņojums par veiksmīgu paroles maiņu | Pareizi |
| 5. | Spēkā esošā parole: r18opX2  Jauna parole: r18opX2 | Tiek parādīta kļūda, ka jaunajai parolei un esošajai jābūt atšķirīgām | Pareizi |
| 6. | Jauna parole: r18opX2  atkārtoti ievadīta jaunā parole: irwv8opX2 | Paziņojums, ka jaunā un atkārtoti ievadītā parole nesakrīt | Pareizi |
| 7. | 7.1. | Piedāvājumu paslēpšana/  deaktivēšana | Vispirms uzklikšķinot uz pogas “Settiings” izvēlētajā konfigurācijas lapā, pēc tam atverot cilni “Manage Options” un noklikšķinot uz pogas “Deactivate” | Datubāzē ir mainījies statuss, pēc kā piedāvājums sākumlapā tiks atzīmēts kā neaktīvs. | Pareizi |
| 8. | 4. | Pasūtījuma veikšana | Pareizi dati Vārds, Uzvārds, telefona numurs, izvēles rūtiņa priekš nosacījumiem | Ieraksts ir pievienots datubāzei, un lietotājam tiek nosūtīts paziņojums par veikmīgo pasūtīšanu. | Pareizi |
| 9. | Ir ievadīti visi lauki, izņemot vārdu, un arī nav atzīmēta atbilstošā izvēles rūtiņa priekš nosacījumiem | Tiek parādīts logs ar vispārējo tekstu, kur norādīts, kā aizpildīt šo veidlapu. Šajā gadījumā ir nepieciešams aizpildīt visus laukus | Pareizi |
| 10. | Telefona numurs: 2413 | Tiks parādīts kļūdas ziņojumu logs, jo nosacījumos ir minēts, ka telefona numuram jābūt vismaz 5 simbolus garumā | Pareizi |
| 11. | 10.1. | Lietotāju komentāru dzēšana | Peles klikšķis uz pogas “Delete” blakus attiecīgajam komentāram | Ieraksts tiek dzēsts no datubāzes, un šis ieraksts nav redzams komentāru lapā | Pareizi |
| 12. | 3.1. | Meklēšanas josla, meklēšana pēc atslēgvārda | Ražotājs: BMW, modelis X5 | Šajā lapā parādās visi piedāvājumi saistībā ar BMW markas automašīnu un X5 modeli | Pareizi |
| 13. | Ražotājs: Tesla | Šajā lapā parādās visi automobiļi ar Tesla zīmolu un visi ar šo ražotāju saistītie modeļi | Pareizi |
| 14. | Ražotājs: BMW, modelis AMG | Tiek parādīts ziņojums, ka nekas nav atrasts, jo šāda modeļa šai automašīnas zīmolam nav | Pareizi |

6.2. tabulas turpinājums

# **NOBEIGUMS**

Nobeigumā ir sasniegts šāds rezultāts: veiksmīgi izveidota tīmekļa lietotne, kas pilnībā atbilst Kvalifikācijas darba izstrādes laikā izvirzītajam uzdevumam. Lietotne nodrošina detalizētu informāciju par automašīnu apskati un piedāvājumiem, ļauj veikt pasūtījumus un rediģēt sava profila datus. Šī tīmekļa lietotne tika izstrādāta, lai nodrošinātu lietotājiem vienkāršu un ērtu veidu, kā atrast automašīnu, kas atbilst viņu vajadzībām un budžetam. Papildus tam ir pieejama konfigurāciju rediģēšana un pasūtījumu pārvaldība, ieskaitot apkalpojošā personāla pilnvaras. Lietotāji var atbildēt uz jautājumiem un mainīt savu profilu, turklāt ir iespēja izvēlēties interesojošos ziņojumus un veikt drošu reģistrāciju un autorizāciju.

Šobrīd šī tīmekļa lietotne tiek izmantota manā personīgajā projektā, kvalifikācijas darba izstrādes kontekstā. Turklāt tā ir potenciāla platforma, kas varētu paplašināt savu pielietojumu nākotnē, piedāvājot klientiem ziņojumus par jaunumiem un piedāvājumiem automašīnu tirgū, kā arī nodrošinot vērtīgu informāciju klientiem. Tomēr šobrīd tā nav izmantota citās vides vai uzņēmējdarbības kontekstos. Attiecībā uz turpmāko attīstību, šī platforma varētu tikt paplašināta un attīstīta par universālu transportlīdzekļu pārdošanas un meklēšanas platformu, sniedzot plašāku pakalpojumu klāstu un paplašinot tās tirgus satura klātbūtni. Tā kā automašīnu pieprasījums nepārtraukti pieaug un ir nepieciešamība pēc efektīvām automašīnu meklēšanas un pārdošanas platformām, ir iespējams, ka šī tīmekļa lietotne varētu tikt attīstīta uz kaut ko lielāku, piemēram, kļūstot par plašāku transportlīdzekļu pārdošanas un meklēšanas platformu, kurā būtu iespējams atrast arī citu veidu transportlīdzekļus, piemēram, motociklus vai kravas automašīnas. Tādējādi šī tīmekļa lietotne var kļūt par svarīgu resursu automašīnu industrijā un piedāvāt vērtīgu pakalpojumu klientiem visā pasaulē.

# **INFORMĀCIJAS AVOTI**

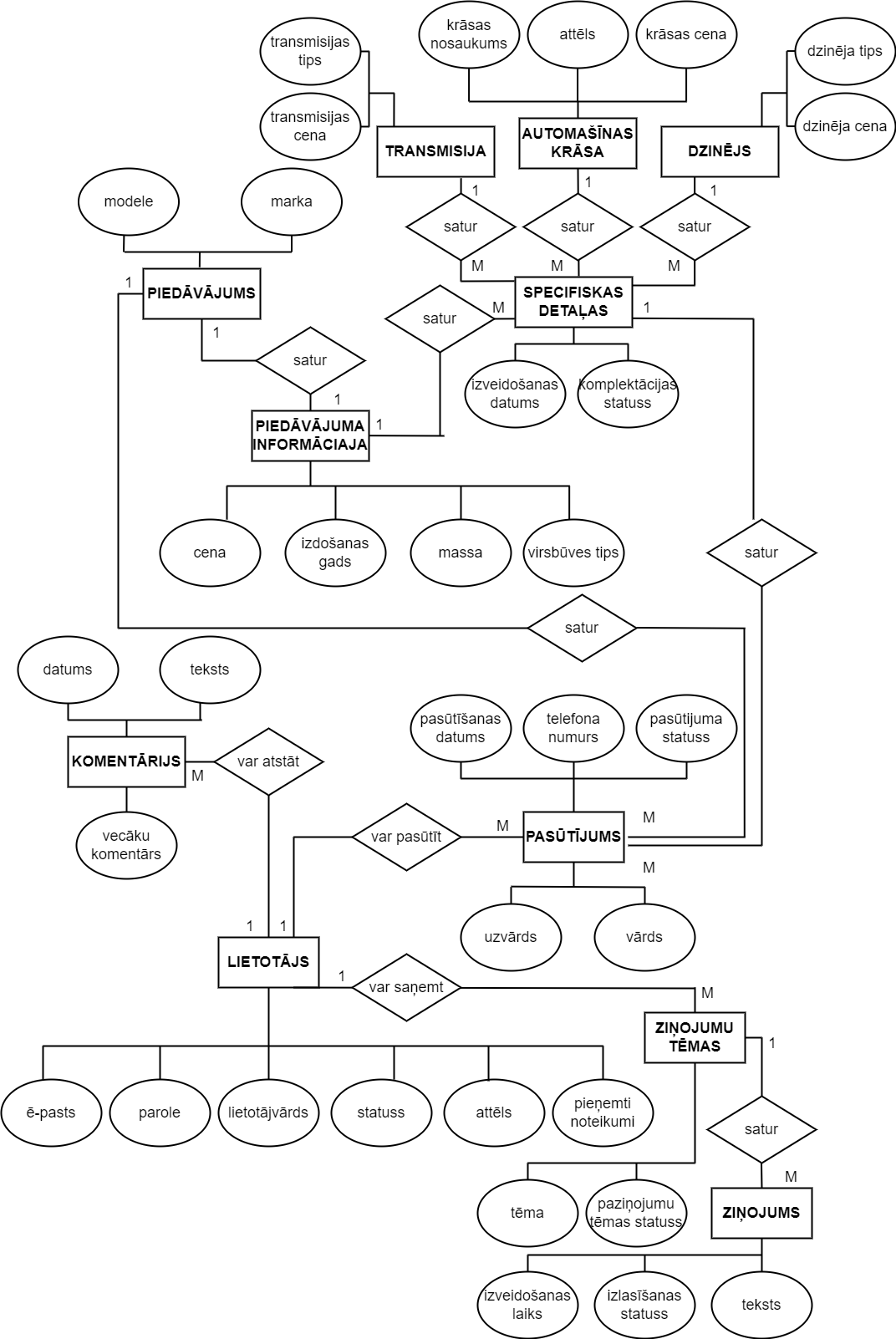
1. Priekš skiču zīmēšanai - <https://www.figma.com> – (Resurss apskatīts 07.11.2023.).
2. Informācija par ER diagrammu – https://www.lucidchart.com/pages/er-diagrams#:~:text=Make%20an%20ERD-,What%20is%20an%20ER%20diagram%3F,each%20other%20within%20a%20system.. – (Resurss apskatīts 09.11.2023.).
3. Informācija par UML diagrammu – <https://www.lucidchart.com/pages/uml-use-case-diagram> - (Resurss apskatīts 09.11.2023.).
4. Diagrammu izveidei - draw.io - (Resurss apskatīts 09.11.2023.).
5. Laragon dokumentācija - <https://laragon.org/docs/index.html> - (Resurss apskatīts 09.11.2023.).
6. Problēmu un nesaprošanu risināšanai - https://stackoverflow.com – (Resurss apskatīts 10.11.2023.).
7. DPD apraksts - <https://elsmar.com/pdf_files/DPD%20FLOW%20DIAGRAM.pdf> - (Resurss apskatīts 12.11.2023.).
8. HTML un CSS sintakse - <https://www.w3schools.com/html/html_css.asp> - (Resurss apskatīts 12.11.2023.).
9. Hypertext Preprocessor (PHP) valodas sintakse - <https://www.php.net/docs.php> - (Resurss apskatīts 16.11.2023.).
10. Apstiprinājuma logs - <https://sweetalert2.github.io> - (Resurss apskatīts 09.02.2024.).
11. PhpSpreadsheet dokumentācija Excel izklājlapu izveidei - <https://phpspreadsheet.readthedocs.io/en/latest/topics/accessing-cells/> - (Resurss apskatīts 15.03.2024.).
12. Bootstrap dokumentācija - <https://getbootstrap.com/docs/4.1/getting-started/introduction/> - (Resurss apskatīts 17.03.2024.).
13. Mans projekts GitHub’ā - <https://github.com/kostjakotovich/autosalons> - (Resurss apskatīts 25.03.2024.).
14. Elementu izmantošana priekš dizaina - <https://uiverse.io> - (Resurss apskatīts 15.04.2024.).
15. Sweetalert dokumentācija - <https://sweetalert2.github.io> - (Resurss apskatīts 03.05.2024.).

# **PIELIKUMI**

## 

## 1. pielikums

## **Entity relationship modelis**



## 2. pielikums

**Programmas pirmkods**

**//Datubāzes pieslēgšana**

<?php

class database {

private $DBusername = "root";

private $DBpassword = "";

private $dbname = "mariadb";

private $DBconnection;

public function connect(){

try{

$this->DBconnection = new PDO("mysql:host=localhost; dbname=$this->dbname" , $this->DBusername, $this->DBpassword);

$this->DBconnection->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

return $this->DBconnection;

}

catch(PDOException $ex){

echo "Exception error: " . $ex->getMessage();

die($ex->getMessage());

}

}

}

**?>**

**//Komentāru attēlošana, pievienošana, dzēšana, atbildes pievienošana (komentāru sistēma)**

<?php

session\_start();

require\_once 'connection.php';

require\_once 'User.php';

require\_once 'Comment.php';

if (!isset($\_SESSION['userID'])) {

header("Location: login.php");

exit();

}

$commentObj = new Comment();

$commentsPerPage = 7;

$startIndex = 0;

$endIndex = $startIndex + $commentsPerPage;

$totalComments = $commentObj->getTotalOriginalCommentsCount();

$totalPages = ceil($totalComments / $commentsPerPage);

if (isset($\_GET['page'])) {

$currentPage = $\_GET['page'];

} else {

$currentPage = 1;

}

$startIndex = ($currentPage - 1) \* $commentsPerPage;

$endIndex = $startIndex + $commentsPerPage;

if ($currentPage == $totalPages) {

$commentsOnLastPage = $totalComments - ($totalPages - 1) \* $commentsPerPage;

$endIndex = $startIndex + $commentsOnLastPage;

}

$comments = $commentObj->getOriginalCommentsForPage($startIndex, $commentsPerPage);

if (isset($\_POST['comment'])) {

$comment = $\_POST['comment'];

$userID = $\_SESSION['userID'];

$commentObj->addComment($comment, $userID);

header("Location: forum.php");

exit;

}

if (isset($\_POST['delete'])) {

$commentID = $\_POST['commentID'];

$commentObj->deleteComment($commentID);

header("Location: forum.php");

exit;

}

if (isset($\_POST['reply'])) {

$reply = $\_POST['reply'];

$userID = $\_SESSION['userID'];

$commentID = $\_POST['parentCommentID'];

$commentObj->addReply($reply, $userID, $commentID);

header("Location: forum.php");

exit;

}

$userObj = new UserMain($\_SESSION['userID']);

if (!$userObj->isRulesAccepted()) {

include 'includes\rules\_modal.php';

}

if (isset($\_POST['accept\_rules'])) {

$userID = $\_SESSION['userID'];

$userObj = new UserMain($userID);

$userObj->acceptRules();

header("Location: " . $\_SERVER['PHP\_SELF']);

exit();

}

?>

<script>

document.querySelectorAll('.reply-btn').forEach(function(btn) {

btn.addEventListener('click', function() {

var parentContainer = this.parentNode.parentNode;

var replyForm = parentContainer.querySelector('.reply-form');

if (replyForm) {

replyForm.style.display = replyForm.style.display === 'none' ? 'block' : 'none';

}

});

});

document.querySelectorAll('.show-replies-btn').forEach(function(btn) {

btn.addEventListener('click', function() {

var repliesContainer = this.nextElementSibling;

if (repliesContainer) {

repliesContainer.style.display = repliesContainer.style.display === 'none' ? 'block' : 'none';

this.textContent = repliesContainer.style.display === 'none' ? 'Show Replies (' + repliesContainer.querySelectorAll('.reply-comment').length + ')' : 'Hide Replies';

}

});

});

</script>

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

var modal = document.getElementById('rulesModal');

var returnButton = document.getElementById('returnButton');

if (returnButton) {

returnButton.onclick = function() {

window.location.href = '../autosalons/index.php';

}

}

});

**//Komentāru klase**

<?php

require\_once 'connection.php';

require\_once 'User.php';

class Comment {

private $conn;

public function \_\_construct() {

$this->conn = (new Database())->connect();

}

public function addComment($comment, $userID) {

date\_default\_timezone\_set('Europe/Riga');

$date = date("Y-m-d H:i:s");

$sql = "INSERT INTO comments (comment, userID, date) VALUES (:comment, :userID, :date)";

$stmt = $this->conn->prepare($sql);

$stmt->bindParam(':comment', $comment);

$stmt->bindParam(':userID', $userID);

$stmt->bindParam(':date', $date);

$stmt->execute();

}

public function deleteComment($commentID) {

$sql = "SELECT parent\_comment\_id FROM comments WHERE commentID = :commentID";

$stmt = $this->conn->prepare($sql);

$stmt->bindParam(':commentID', $commentID, PDO::PARAM\_INT);

$stmt->execute();

$parentCommentID = $stmt->fetchColumn();

$this->deleteCommentAndChildren($commentID);

}

private function deleteCommentAndChildren($commentID) {

$sql = "SELECT commentID FROM comments WHERE parent\_comment\_id = :commentID";

$stmt = $this->conn->prepare($sql);

$stmt->bindParam(':commentID', $commentID, PDO::PARAM\_INT);

$stmt->execute();

$childComments = $stmt->fetchAll(PDO::FETCH\_COLUMN);

foreach ($childComments as $childCommentID) {

$this->deleteCommentAndChildren($childCommentID);

}

$sql = "DELETE FROM comments WHERE commentID = :commentID";

$stmt = $this->conn->prepare($sql);

$stmt->bindParam(':commentID', $commentID, PDO::PARAM\_INT);

$stmt->execute();

}

public function getTotalOriginalCommentsCount() {

$sql = "SELECT COUNT(\*) as total FROM comments WHERE comments.parent\_comment\_id IS NULL";

$stmt = $this->conn->query($sql);

$result = $stmt->fetch(PDO::FETCH\_ASSOC);

return $result['total'];

}

public function getOriginalCommentsForPage($startIndex, $commentsPerPage) {

$sql = "SELECT comments.commentID, comments.userID, comments.comment, user.username, user.picture, comments.date

FROM comments

LEFT JOIN user ON comments.userID = user.userID

WHERE comments.parent\_comment\_id IS NULL

ORDER BY comments.commentID DESC

LIMIT :startIndex, :commentsPerPage";

$stmt = $this->conn->prepare($sql);

$stmt->bindValue(':startIndex', (int) $startIndex, PDO::PARAM\_INT);

$stmt->bindValue(':commentsPerPage', (int) $commentsPerPage, PDO::PARAM\_INT);

$stmt->execute();

$comments = array();

while ($row = $stmt->fetch(PDO::FETCH\_ASSOC)) {

$comment = array(

'commentID' => $row['commentID'],

'userID' => $row['userID'],

'username' => $row['username'],

'picture' => $row['picture'],

'comment' => $row['comment'],

'date' => $row['date']

);

$comments[] = $comment;

}

return $comments;

}

public function addReply($comment, $userID, $parentCommentID) {

date\_default\_timezone\_set('Europe/Riga');

$date = date("Y-m-d H:i:s");

$replyUsername = $this->getUsernameByUserID($userID);

$originalCommentUserID = $this->getUserIDForOriginalComment($parentCommentID);

$sql = "INSERT INTO comments (comment, userID, date, parent\_comment\_id) VALUES (:comment, :userID, :date, :parentCommentID)";

$stmt = $this->conn->prepare($sql);

$stmt->bindParam(':comment', $comment);

$stmt->bindParam(':userID', $userID);

$stmt->bindParam(':date', $date);

$stmt->bindParam(':parentCommentID', $parentCommentID);

$stmt->execute();

$maxMessageLength = 100;

if (strlen($comment) > $maxMessageLength) {

$truncatedMessage = substr($comment, 0, $maxMessageLength) . "...";

$notificationText = "You have received a reply from '$replyUsername' to your comment: '$truncatedMessage'. For the full message, visit the <a href='forum.php'>forum</a>.";

} else {

$notificationText = "You have received a reply from '$replyUsername' to your comment: '$comment'. Visit the <a href='forum.php'>forum</a> to view the response.";

}

$userMain = new UserMain($originalCommentUserID);

$topicName = 'Forum';

$topicID = $userMain->getNotificationTopicIDByName($topicName);

$userMain->addNotification($topicID, $notificationText);

}

public function getRepliesForComment($parentCommentID) {

$replies = array();

$this->getRepliesRecursive($parentCommentID, $replies);

usort($replies, function($a, $b) {

return strtotime($a['date']) - strtotime($b['date']);

});

return $replies;

}

private function getRepliesRecursive($parentCommentID, &$replies) {

$sql = "SELECT comments.commentID, comments.userID, user.username, user.picture, comments.comment, comments.date

FROM comments

INNER JOIN user ON comments.userID = user.userID

WHERE comments.parent\_comment\_id = :parentCommentID

ORDER BY comments.date DESC";

$stmt = $this->conn->prepare($sql);

$stmt->bindParam(':parentCommentID', $parentCommentID, PDO::PARAM\_INT);

$stmt->execute();

while ($row = $stmt->fetch(PDO::FETCH\_ASSOC)) {

$reply = array(

'commentID' => $row['commentID'],

'userID' => $row['userID'],

'username' => $row['username'],

'picture' => $row['picture'],

'comment' => $row['comment'],

'date' => $row['date']

);

$replies[] = $reply;

$this->getRepliesRecursive($row['commentID'], $replies);

}

}

public function getUsernameForOriginalComment($commentID) {

$query = "SELECT u.username

FROM user u

JOIN comments c ON u.userID = c.userID

WHERE c.commentID = (

SELECT parent\_comment\_id

FROM comments

WHERE commentID = :commentID

)";

$stmt = $this->conn->prepare($query);

$stmt->bindParam(':commentID', $commentID);

$stmt->execute();

$result = $stmt->fetch(PDO::FETCH\_ASSOC);

return $result['username'];

}

public function getUserIDForOriginalComment($parentCommentID) {

$query = "SELECT c.userID

FROM comments c

WHERE c.commentID = :parentCommentID";

$stmt = $this->conn->prepare($query);

$stmt->bindParam(':parentCommentID', $parentCommentID);

$stmt->execute();

$result = $stmt->fetch(PDO::FETCH\_ASSOC);

return $result['userID'];

}

public function getUsernameByUserID($userID) {

$query = "SELECT username FROM user WHERE userID = :userID";

$stmt = $this->conn->prepare($query);

$stmt->bindParam(':userID', $userID);

$stmt->execute();

$result = $stmt->fetch(PDO::FETCH\_ASSOC);

return $result['username'];

}

}

**//Piedāvājumu meklēšana pēc atslēgvārdiem un filtrešanas datiem**

<?php

session\_start();

require\_once 'connection.php';

require\_once 'Offer.php';

require\_once 'phpSearchOption.php';

$offer = new Offer();

$offers = $offer->getAllOffers();

$searchOption = new SearchOption();

if (isset($\_GET['searchBtn'])) {

$search = $\_GET['search'];

$selectedBrand = $\_GET['brand'] ?? '';

$selectedModel = $\_GET['model'] ?? '';

$selectedTransmission = $\_GET['transmission'] ?? '';

$selectedType = $\_GET['type'] ?? '';

$selectedYear = $\_GET['year'] ?? '';

$selectedColor = $\_GET['color'] ?? '';

$currentMinPrice = $\_GET['minPrice'] ?? 0;

$currentMaxPrice = $\_GET['maxPrice'] ?? '';

$offers = $searchOption->searchOffers($search, $selectedBrand, $selectedModel, $selectedType, $selectedYear, $selectedColor, $selectedTransmission, $currentMinPrice, $currentMaxPrice);

}

$selectedType = $\_GET['type'] ?? '';

$selectedBrand = $\_GET['brand'] ?? '';

$selectedColor = $\_GET['color'] ?? '';

$selectedTransmission = $\_GET['transmission'] ?? '';

require\_once 'includes/car\_body\_types.php';

require\_once 'includes/car\_colors.php';

require\_once 'includes/car\_brands.php';

require\_once 'includes/car\_models.php';

require\_once 'includes/transmissions\_types.php';

$selectedYear = $\_GET['year'] ?? '';

$currentMinPrice = $\_GET['minPrice'] ?? '';

$currentMaxPrice = $\_GET['maxPrice'] ?? '';

?>

<script>

var modelsByBrand = <?php echo json\_encode($modelsByBrand); ?>;

document.getElementById('brand').addEventListener('change', function() {

var selectedBrand = this.value;

var modelSelect = document.getElementById('model');

var modelGroup = document.getElementById('model-group');

if (selectedBrand) {

modelSelect.innerHTML = '<option value="">Select Brand First</option>';

if (modelsByBrand[selectedBrand]) {

modelsByBrand[selectedBrand].forEach(function(model) {

var option = document.createElement('option');

option.value = model;

option.textContent = model;

modelSelect.appendChild(option);

});

}

modelSelect.disabled = false;

modelSelect.querySelector('option').textContent = 'All models';

modelSelect.value = '';

} else {

modelSelect.disabled = true;

modelSelect.innerHTML = '<option value="">Select Brand First</option>';

}

});

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

var urlParams = new URLSearchParams(window.location.search);

var selectedBrand = urlParams.get('brand');

var selectedModel = urlParams.get('model');

var modelSelect = document.getElementById('model');

modelSelect.innerHTML = '<option value="">Select Brand First</option>';

if (selectedBrand) {

if (modelsByBrand[selectedBrand]) {

modelsByBrand[selectedBrand].forEach(function(model) {

var option = document.createElement('option');

option.value = model;

option.textContent = model;

modelSelect.appendChild(option);

});

}

modelSelect.disabled = false;

modelSelect.querySelector('option').textContent = 'All models';

if (selectedModel && modelsByBrand[selectedBrand] && modelsByBrand[selectedBrand].includes(selectedModel)) {

modelSelect.value = selectedModel;

} else {

modelSelect.value = '';

}

} else {

modelSelect.disabled = true;

modelSelect.querySelector('option').textContent = 'Select Brand First';

}

});

</script>

<?php

require\_once 'connection.php';

class SearchOption extends Database

{

public function searchOffers($search, $selectedBrand, $selectedModel, $selectedType, $selectedYear, $selectedColor, $selectedTransmission, $currentMinPrice, $currentMaxPrice)

{

$query = "SELECT offers.\*, offersinfo.\*, specific\_details.\*, car\_colors.\*, transmission.\*, engine.\*

FROM offers

INNER JOIN offersinfo ON offersinfo.offersID = offers.offerID

INNER JOIN specific\_details ON offersinfo.offersInfoID = specific\_details.offersInfoID

INNER JOIN car\_colors ON specific\_details.colorID = car\_colors.colorID

INNER JOIN transmission ON specific\_details.transmissionID = transmission.transmissionID

INNER JOIN engine ON specific\_details.engineID = engine.engineID

WHERE (manufacturer LIKE :search OR type LIKE :search OR CONCAT(manufacturer, ' ', type) LIKE :search OR CONCAT(manufacturer, type) LIKE :search)";

if (!empty($selectedBrand)) {

$query .= " AND offers.manufacturer=:selectedBrand";

}

if (!empty($selectedModel)) {

$query .= " AND offers.type=:selectedModel";

}

if (!empty($selectedType)) {

$query .= " AND offersinfo.body\_type=:selectedType";

}

if (!empty($selectedYear)) {

$query .= " AND offersinfo.yearOfManufacture=:selectedYear";

}

if (!empty($selectedColor)) {

$query .= " AND car\_colors.color=:selectedColor";

}

if (!empty($selectedTransmission)) {

$selectedTransmission = '%' . $selectedTransmission . '%';

$query .= " AND transmission.transmission\_type LIKE :selectedTransmission";

}

if (!empty($currentMinPrice)) {

$query .= " AND (offersinfo.price + car\_colors.color\_price + transmission.transmission\_price + engine.engine\_price) >= :currentMinPrice";

}

if (!empty($currentMaxPrice)) {

$query .= " AND (offersinfo.price + car\_colors.color\_price + transmission.transmission\_price + engine.engine\_price) <= :currentMaxPrice";

}

$stmt = $this->connect()->prepare($query);

$stmt->bindValue(':search', '%' . $search . '%', PDO::PARAM\_STR);

if (!empty($selectedBrand)) {

$stmt->bindValue(':selectedBrand', $selectedBrand, PDO::PARAM\_STR);

}

if (!empty($selectedModel)) {

$stmt->bindValue(':selectedModel', $selectedModel, PDO::PARAM\_STR);

}

if (!empty($selectedType)) {

$stmt->bindValue(':selectedType', $selectedType, PDO::PARAM\_STR);

}

if (!empty($selectedYear)) {

$stmt->bindValue(':selectedYear', $selectedYear, PDO::PARAM\_STR);

}

if (!empty($selectedColor)) {

$stmt->bindValue(':selectedColor', $selectedColor, PDO::PARAM\_STR);

}

if (!empty($selectedTransmission)) {

$stmt->bindValue(':selectedTransmission', $selectedTransmission, PDO::PARAM\_STR);

}

if (!empty($currentMinPrice)) {

$stmt->bindValue(':currentMinPrice', $currentMinPrice, PDO::PARAM\_INT);

}

if (!empty($currentMaxPrice)) {

$stmt->bindValue(':currentMaxPrice', $currentMaxPrice, PDO::PARAM\_INT);

}

$stmt->execute();

$result = $stmt->fetchAll();

return $result;

}

}

?>

**//Lietotāja reģistrācija**

<?php

session\_start();

require\_once 'connection.php';

require\_once 'User.php';

class UserRegistration extends UserMain {

private $conn;

public function \_\_construct($username, $email, $password) {

parent::\_\_construct(null);

$this->username = $username;

$this->email = $email;

$this->password = $password;

$this->conn = (new Database())->connect();

}

public function registration() {

$errors = array();

if (isset($\_POST['reg'])) {

if (empty($this->username)) {

array\_push($errors, "Username is required");

} elseif (strlen($this->username) > 20) {

array\_push($errors, "Username should be up to 20 characters");

} elseif (strlen($this->username) < 3) {

array\_push($errors, "Username should be at least 3 characters");

}

if (empty($this->email)) {

array\_push($errors, "Email is required");

} elseif (strlen($this->email) > 40) {

array\_push($errors, "Email should be up to 40 characters");

} elseif (!filter\_var($this->email, FILTER\_VALIDATE\_EMAIL)) {

array\_push($errors, "Invalid email format");

}

if (empty($this->password)) {

array\_push($errors, "Password is required");

} elseif (strlen($this->password) > 30) {

array\_push($errors, "Password should be up to 30 characters");

} elseif (strlen($this->password) < 6) {

array\_push($errors, "Password should be at least 6 characters");

}

if (empty($\_POST['password\_confirm'])) { array\_push($errors, "Verify your password");}

if ($this->password !== $\_POST['password\_confirm']) {

array\_push($errors, "Passwords do not match");

}

$result = $this->conn->prepare("SELECT \* FROM user WHERE username=:username OR email=:email LIMIT 1");

$result->execute(array(':username' => $this->username, ':email' => $this->email));

$user = $result->fetch();

if ($user) {

if ($user['username'] === $this->username) {

array\_push($errors, "Username already exists!");

}

if ($user['email'] === $this->email) {

array\_push($errors, "Email already exists!");

}

}

if (count($errors) == 0) {

$passwordHash = password\_hash($this->password, PASSWORD\_DEFAULT);

$result = $this->conn->prepare("INSERT INTO user (username, email, password, roleID) VALUES(:username, :email, :password, :roleID)");

if(!$result->execute(array(':username' => $this->username, ':email' => $this->email, ':password' => $passwordHash, ':roleID' => 0))) {

$result = null;

$\_SESSION['error'] = 'Failed to register';

}

else {

$\_SESSION['success'] = 'Registration successful';

}

$result = null;

$result = $this->conn->prepare("SELECT userID, roleID FROM user WHERE email=:email");

$result->execute(array(':email' => $this->email));

$user = $result->fetch();

$\_SESSION['userID'] = $user['userID'];

$\_SESSION['roleID'] = $user['roleID'];

if ($\_SESSION['success']) {

$defaultAvatarURL = 'img/avatar/default.png';

$userID = $user['userID'];

$user = new UserMain($userID);

$user->addDefaultNotificationTopics();

$user->updatePicture($defaultAvatarURL);

$topicName = 'Profile';

$topicID = $user->getNotificationTopicIDByName($topicName);

$notificationText = "You have successfully registered! If you need <a href='infoPage.php'>Help</a>, please visit the Help section.";

$user->addNotification($topicID, $notificationText);

}

header('location: index.php');

}

else {

$\_SESSION['error'] = implode("<br>", $errors);

header('location: registrationPage.php');

}

}

else {

$\_SESSION['error'] = 'Invalid request';

header('location: registrationPage.php');

}

}

}

$username = $\_POST['username'];

$email = $\_POST['email'];

$password = $\_POST['password'];

$user = new UserRegistration($\_POST['username'], $\_POST['email'], $\_POST['password']);

$user->registration();

?>

<?php

if(isset($\_SESSION['error'])){

?>

<div class="alert alert-danger text-center" style="margin-top:20px;">

<?php echo $\_SESSION['error']; ?>

</div>

<?php

unset($\_SESSION['error']);

}

if(isset($\_SESSION['success'])){

?>

<div class="alert alert-success text-center" style="margin-top:20px;">

<?php echo $\_SESSION['success']; ?>

</div>

<?php

unset($\_SESSION['success']);

}

?>

function closeRegistration() {

window.location.href="../autosalons/index.php";

}

**//Lietotāja autorizācija**

<?php

session\_start();

require\_once 'connection.php';

class Login {

private $db;

public function \_\_construct(database $db) {

$this->db = $db;

}

public function login($username, $password) {

$conn = $this->db->connect();

$sql = "SELECT \* FROM user WHERE username = ?";

$stmt = $conn->prepare($sql);

$stmt->execute([$username]);

$row = $stmt->fetch(PDO::FETCH\_ASSOC);

$\_SESSION["userID"] = $row["userID"];

if($row){

if ($row['roleID'] == 1 && $row['password'] == $password) {

$\_SESSION["roleID"] = $row["roleID"];

$\_SESSION['success'] = 'Login successful';

header('location: index.php');

}

else if (password\_verify($password, $row['password']))

{

$\_SESSION["roleID"] = $row["roleID"];

$\_SESSION['success'] = 'Login successful';

header('location: index.php');

}

else{

$\_SESSION['error'] = 'Password incorrect';

header('location: loginPage.php');

}

}

else{

$\_SESSION['error'] = 'No account with that username';

header('location: loginPage.php');

}

}

}

if(isset($\_POST['username']) && isset($\_POST['password'])){

$username = $\_POST['username'];

$password = $\_POST['password'];

$login = new Login(new database());

$login->login($username, $password);

}

else{

$\_SESSION['error'] = 'Please enter a username and password';

header('location: loginPage.php');

}

?>

<?php

if(isset($\_SESSION['error'])){

?>

<div class="alert alert-danger text-center" style="margin-top:20px;">

<?php echo $\_SESSION['error']; ?>

</div>

<?php

unset($\_SESSION['error']);

}

if(isset($\_SESSION['success'])){

?>

<div class="alert alert-success text-center" style="margin-top:20px;">

<?php echo $\_SESSION['success']; ?>

</div>

<?php

unset($\_SESSION['success']);

}

?>

function closeLogin() {

window.location.href="../autosalons/index.php";

}

**//Lietotāju informācija rediģēšana un ziņojumu sistēma (attēlošana, dzēšana, statusa maiņa)**

<?php

session\_start();

require\_once 'connection.php';

require\_once 'User.php';

require\_once 'Order.php';

if(isset($\_SESSION["userID"])) {

$userID = $\_SESSION["userID"];

$user = new UserMain($userID);

$userInfo = $user->getUserInfo();

if(isset($userInfo)){

$username = $userInfo['username'];

$email = $userInfo['email'];

}

} else {

header('location: loginPage.php');

}

$change\_errors = array();

if (isset($\_POST["changePassword"])) {

$change\_errors = array();

$user = new UserMain($userID);

$userPassword = $user->getPassword();

$currentPassword = $\_POST["currentPassword"];

$newPassword = $\_POST["newPassword"];

$confirmPassword = $\_POST["confirmPassword"];

if (!password\_verify($currentPassword, $userPassword)) {

array\_push($change\_errors,"Current password is incorrect.");

}

if (password\_verify($newPassword, $userPassword)) {

array\_push($change\_errors,"New password must not be the same as current password.");

}

if (strlen($newPassword) > 30) {

array\_push($change\_errors, "New password should be up to 30 characters");

}

if (strlen($newPassword) < 6) {

array\_push($change\_errors, "New password should be at least 6 characters");

}

if ($newPassword !== $confirmPassword) {

array\_push($change\_errors,"New password and confirmation password do not match.");

}

if (count($change\_errors) == 0) {

if ($user->changePassword($currentPassword, $newPassword, $confirmPassword)) {

$\_SESSION['success\_change'] = 'Password change successful';

session\_write\_close();

header('location: profile.php');

} else {

array\_push($change\_errors,"Unable to change password. Please check your current password and make sure the new password fields match.");

}

}

}

$order = new Order();

$orders = $order->getOrderInfo($userID);

$totalSum = $order -> getOrderSum($userID);

if (isset($\_FILES["avatar"])) {

$uploadDir = 'img/avatar/';

$extension = pathinfo($\_FILES['avatar']['name'], PATHINFO\_EXTENSION);

$uniqueName = uniqid() . '.' . $extension;

$uploadFile = $uploadDir . $uniqueName;

if (move\_uploaded\_file($\_FILES['avatar']['tmp\_name'], $uploadFile)) {

$user->updatePicture($uploadFile);

header('Location: profile.php');

exit;

} else {

header('Location: profile.php');

}

}

?>

// Ziņojumu inicializēšana

<script>

$(document).ready(async function() {

await initializeNotificationIcon();

});

async function initializeNotificationIcon() {

try {

const response = await $.ajax({

url: 'get-notification-status.php',

type: 'GET',

dataType: 'json',

});

if (response.status === 'unread') {

$('#notification-bell').attr('src', 'img/icon/bell-active.png');

} else {

}

} catch (error) {

console.error(error);

}

}

function editNotificationIcon() {

var currentIcon = $('#notification-bell').attr('src');

if (currentIcon !== 'img/icon/bell.png') {

$('#notification-bell').attr('src', 'img/icon/bell.png');

$.ajax({

url: 'mark-notification-as-read.php',

type: 'POST',

});

}

}

</script>

<?php

session\_start();

require\_once 'connection.php';

if (isset($\_SESSION['userID'])) {

$userID = $\_SESSION['userID'];

$conn = (new Database())->connect();

$sql = "SELECT COUNT(\*) as unread\_count FROM notifications

WHERE topic\_id IN (

SELECT topic\_id

FROM notification\_topics

WHERE userID = :userID

) AND is\_read = 0";

$stmt = $conn->prepare($sql);

$stmt->bindValue(':userID', $userID, PDO::PARAM\_INT);

if ($stmt->execute()) {

$result = $stmt->fetch(PDO::FETCH\_ASSOC);

$unreadCount = $result['unread\_count'];

$response = array('status' => ($unreadCount > 0) ? 'unread' : 'read');

echo json\_encode($response);

} else {

echo json\_encode(array('status' => 'error'));

}

} else {

echo json\_encode(array('status' => 'invalid'));

}

?>

<?php

session\_start();

require\_once 'connection.php';

if (isset($\_SESSION['userID'])) {

$userID = $\_SESSION['userID'];

$conn = (new Database())->connect();

$sql = "UPDATE `notifications`

SET `is\_read` = 1

WHERE `topic\_id` IN (SELECT `topic\_id`

FROM `notification\_topics`

WHERE `userID` = :userID)";

$stmt = $conn->prepare($sql);

$stmt->bindValue(':userID', $userID, PDO::PARAM\_INT);

if ($stmt->execute()) {

echo 'success';

} else {

echo 'error';

}

} else {

echo 'invalid';

}

?>

<?php

require\_once '../User.php';

$userID = isset($\_POST['userID']) ? $\_POST['userID'] : null;

if ($userID) {

$user = new UserMain($userID);

$user->deleteAllNotifications();

}

$previousURL = $\_SERVER['HTTP\_REFERER'];

header("Location: $previousURL");

exit();

?>

<?php

require\_once '../User.php';

if (isset($\_POST['notificationID'])) {

$notificationID = $\_POST['notificationID'];

$userID = isset($\_SESSION['userID']);

$user = new UserMain($userID);

$user->deleteNotification($notificationID);

$previousURL = $\_SERVER['HTTP\_REFERER'];

header("Location: $previousURL");

exit();

}

?>

//Attēlošanas statusa maiņa

<?php

require\_once 'User.php';

if (isset($\_SESSION['userID'])) {

$userID = $\_SESSION['userID'];

$user = new UserMain($userID);

$notifications = $user->getNotifications();

$forumTopicID = $user->getNotificationTopicIDByName('Forum');

$ordersTopicID = $user->getNotificationTopicIDByName('Orders');

$profileTopicID = $user->getNotificationTopicIDByName('Profile');

$forumStatus = $user->getNotificationTopicStatus($forumTopicID);

$ordersStatus = $user->getNotificationTopicStatus($ordersTopicID);

$profileStatus = $user->getNotificationTopicStatus($profileTopicID);

}

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

if (isset($\_POST['updateFilter'])) {

if (isset($\_POST['forum']) && isset($\_POST['orders']) && isset($\_POST['profile'])) {

$forumStatus = $\_POST['forum'] === 'disable' ? 'disable' : 'enable';

$ordersStatus = $\_POST['orders'] === 'disable' ? 'disable' : 'enable';

$profileStatus = $\_POST['profile'] === 'disable' ? 'disable' : 'enable';

$user->updateNotificationTopicStatus($forumTopicID, $forumStatus);

$user->updateNotificationTopicStatus($ordersTopicID, $ordersStatus);

$user->updateNotificationTopicStatus($profileTopicID, $profileStatus);

?><script>

window.location.href = window.location.href;

</script><?php

}

}

}

?>

**//Lietotāja klase**

<?php

require\_once 'connection.php';

class UserMain {

private $username;

private $email;

private $password;

private $password\_hash;

public function \_\_construct($userID){

$this->userID = $userID;

$this->conn = (new Database())->connect();

}

private $userID;

private $conn;

/\* Profile methods \*/

public function getUserInfo() {

$sql = "SELECT username, email FROM user WHERE userID = ?";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$this->userID]);

$user = $stmt->fetch(PDO::FETCH\_ASSOC);

return $user;

}

public function getPassword() {

$stmt = $this->conn->prepare("SELECT password FROM user WHERE userID = :userID");

$stmt->bindParam(':userID', $this->userID, PDO::PARAM\_INT);

$stmt->execute();

$result = $stmt->fetchAll(PDO::FETCH\_ASSOC);

if (count($result) == 1) {

$row = $result[0];

return $row['password'];

} else {

return "";

}

}

public function verifyPassword($password, $hashedPassword) {

return password\_verify($password, $hashedPassword);

}

public function getPasswordHash() {

return $this->password\_hash;

}

public function changePassword($currentPassword, $newPassword, $confirmPassword) {

try {

$stmt = $this->conn->prepare("SELECT password FROM user WHERE userID = :userID");

$stmt->bindParam(':userID', $this->userID);

$stmt->execute();

$result = $stmt->fetch();

if(password\_verify($currentPassword, $result['password']) && $newPassword == $confirmPassword) {

$newPasswordHash = password\_hash($newPassword, PASSWORD\_DEFAULT);

$stmt = $this->conn->prepare("UPDATE user SET password = :newPassword WHERE userID = :userID");

$stmt->bindParam(':newPassword', $newPasswordHash);

$stmt->bindParam(':userID', $this->userID);

$stmt->execute();

$topicName = 'Profile';

$topicID = $this->getNotificationTopicIDByName($topicName);

$notificationText = "Your password has been successfully changed!";

$this->addNotification($topicID, $notificationText);

return true;

} else {

return false;

}

} catch(PDOException $e) {

echo "Error: " . $e->getMessage();

return false;

}

}

public function getPicture() {

$sql = "SELECT picture FROM user WHERE userID = ?";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$this->userID]);

return $stmt->fetchColumn();

}

public function updatePicture($newPictureURL) {

$sql = "SELECT picture FROM user WHERE userID = ?";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$this->userID]);

$currentPictureURL = $stmt->fetchColumn();

if ($currentPictureURL && basename($currentPictureURL) !== 'default.png') {

$currentPicturePath = \_\_DIR\_\_ . '/../' . $currentPictureURL;

if (file\_exists($currentPicturePath)) {

unlink($currentPicturePath);

}

}

$sql = "UPDATE user SET picture = ? WHERE userID = ?";

$stmt = $this->conn->prepare($sql);

return $stmt->execute([$newPictureURL, $this->userID]);

}

/\* Notifications methods \*/

public function addNotification($topicID, $message) {

$stmt = $this->conn->prepare("INSERT INTO notifications (topic\_id, message) VALUES (:topicID, :message)");

$stmt->bindParam(':topicID', $topicID, PDO::PARAM\_INT);

$stmt->bindParam(':message', $message, PDO::PARAM\_STR);

$stmt->execute();

}

public function getNotifications() {

$sql = "SELECT n.\*, nt.topic\_name

FROM notifications n

INNER JOIN notification\_topics nt ON n.topic\_id = nt.topic\_id

WHERE nt.userID = :userID AND nt.status = 'enable'

ORDER BY n.created\_at DESC";

$stmt = $this->conn->prepare($sql);

$stmt->bindParam(':userID', $this->userID, PDO::PARAM\_INT);

$stmt->execute();

$notifications = $stmt->fetchAll(PDO::FETCH\_ASSOC);

return $notifications;

}

public function deleteNotification($notificationID) {

$sql = "DELETE FROM notifications WHERE notification\_id = :notificationID";

$stmt = $this->conn->prepare($sql);

$stmt->bindParam(':notificationID', $notificationID, PDO::PARAM\_INT);

return $stmt->execute();

}

public function deleteAllNotifications() {

$sql = "DELETE FROM notifications

WHERE topic\_id IN (SELECT topic\_id FROM notification\_topics WHERE userID = :userID)";

$stmt = $this->conn->prepare($sql);

$stmt->bindParam(':userID', $this->userID, PDO::PARAM\_INT);

return $stmt->execute();

}

public function getNotificationTopics() {

$sql = "SELECT \* FROM notification\_topics WHERE userID = :userID";

$stmt = $this->conn->prepare($sql);

$stmt->bindParam(':userID', $this->userID, PDO::PARAM\_INT);

$stmt->execute();

$topics = $stmt->fetchAll(PDO::FETCH\_ASSOC);

return $topics;

}

public function getNotificationTopicIDByName($topicName) {

$sql = "SELECT topic\_id FROM notification\_topics WHERE topic\_name = :topicName AND userID = :userID";

$stmt = $this->conn->prepare($sql);

$stmt->bindParam(':topicName', $topicName, PDO::PARAM\_STR);

$stmt->bindParam(':userID', $this->userID, PDO::PARAM\_INT);

$stmt->execute();

$topicID = $stmt->fetchColumn();

return $topicID;

}

public function updateNotificationTopicStatus($topicID, $status) {

$sql = "UPDATE notification\_topics SET status = :status WHERE topic\_id = :topicID AND userID = :userID";

$stmt = $this->conn->prepare($sql);

$stmt->bindParam(':topicID', $topicID, PDO::PARAM\_INT);

$stmt->bindParam(':status', $status, PDO::PARAM\_STR);

$stmt->bindParam(':userID', $this->userID, PDO::PARAM\_INT);

return $stmt->execute();

}

public function getNotificationTopicStatus($topicID) {

$sql = "SELECT status FROM notification\_topics WHERE topic\_id = :topicID AND userID = :userID";

$stmt = $this->conn->prepare($sql);

$stmt->bindParam(':topicID', $topicID, PDO::PARAM\_INT);

$stmt->bindParam(':userID', $this->userID, PDO::PARAM\_INT);

$stmt->execute();

$status = $stmt->fetchColumn();

return $status;

}

public function addDefaultNotificationTopics() {

$topics = ['Forum', 'Orders', 'Profile'];

foreach ($topics as $topicName) {

$topicID = $this->getNotificationTopicIDByName($topicName);

if (!$topicID) {

$status = 'enable';

$this->insertNotificationTopic($topicName, $status);

}

}

}

private function insertNotificationTopic($topicName, $status) {

$sql = "INSERT INTO notification\_topics (userID, topic\_name, status) VALUES (:userID, :topicName, :status)";

$stmt = $this->conn->prepare($sql);

$stmt->bindParam(':userID', $this->userID, PDO::PARAM\_INT);

$stmt->bindParam(':topicName', $topicName, PDO::PARAM\_STR);

$stmt->bindParam(':status', $status, PDO::PARAM\_STR);

$stmt->execute();

}

/\* Rules methods \*/

public function isRulesAccepted() {

$sql = "SELECT rules\_accepted FROM user WHERE userID = ?";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$this->userID]);

$rulesAccepted = $stmt->fetchColumn();

return $rulesAccepted === 1;

}

public function acceptRules() {

$sql = "UPDATE user SET rules\_accepted = 1 WHERE userID = ?";

$stmt = $this->conn->prepare($sql);

$stmt->bindParam(1, $this->userID, PDO::PARAM\_INT);

$stmt->execute();

if ($stmt->rowCount() > 0) {

return true;

} else {

return false;

}

}

}

?>

**//Piedāvājumu un komplektāciju pievienošana, informācijas rediģēšana, dzēšana (Piedāvājuma klase)**

<?php

session\_start();

require\_once 'connection.php';

require\_once 'Offer.php';

if (!isset($\_SESSION['userID']) || $\_SESSION['roleID'] !== 1) {

header("Location: index.php");

exit();

}

if (isset($\_POST['submit'])) {

$yearOfManufacture = $\_POST['yearOfManufacture'];

$\_POST['color\_price'] = floatval($\_POST['color\_price']);

$\_POST['transmission\_price'] = floatval($\_POST['transmission\_price']);

$\_POST['engine\_price'] = floatval($\_POST['engine\_price']);

$\_POST['price'] = floatval($\_POST['price']);

$\_POST['weight'] = floatval($\_POST['weight']);

$offer = new Offer();

$offer->addOffer($\_POST);

}

?>

<?php

session\_start();

require\_once 'connection.php';

require\_once 'Offer.php';

require\_once 'User.php';

require\_once 'Order.php';

error\_reporting(E\_ALL);

ini\_set('display\_errors', 1);

if(isset($\_SESSION['userID'])) {

$userID = $\_SESSION['userID'];

$user = new UserMain($userID);

$userInfo = $user->getUserInfo();

}

$offerID = isset($\_GET['offerID']);

if (isset($\_GET['offerID'])) {

$offerID = $\_GET['offerID'];

$detailsID = $\_GET['detailsID'];

$offer = new Offer();

$offersInfoID = $offer->getOffersInfoID($offerID);

$isActive = $offer->isOfferActive($detailsID);

$selectedOffer = $offer->getOffer($offerID);

$selectedOfferColor = $offer->getOfferColor($offersInfoID, $detailsID);

$selectedOfferDetails = $offer->getOfferDetails($offersInfoID);

$selectedOfferTransmission = $offer->getOfferTransmission($offersInfoID, $detailsID);

$selectedOfferEngine = $offer->getOfferEngine($offersInfoID, $detailsID);

$selectedOfferInfo = $offer->getOfferInfo($offerID);

if (isset($\_GET['choose\_configuration'])) {

$selectedDetailsID = $\_GET['detailsID'];

$selectedOfferDetails = $offer->getOfferDetailsByID($offersInfoID);

} else {

$selectedDetailsID = null;

$selectedOfferDetails = $offer->getOfferDetails($offersInfoID);

}

if (isset($\_GET['detailsID'])) {

$selectedDetailsID = $\_GET['detailsID'];

$\_SESSION['selectedDetailsID'] = $selectedDetailsID;

$selectedOfferDetails = $offer->getOfferDetailsByID($offersInfoID);

} else {

$selectedDetailsID = $\_SESSION['selectedDetailsID'] ?? null;

$selectedOfferDetails = $offer->getOfferDetailsByID($offersInfoID);

}

$carPriceDisplay = $selectedOfferInfo['price'] + $selectedOfferTransmission['transmission\_price'] + $selectedOfferColor['color\_price'] + $selectedOfferEngine['engine\_price'] . ' € ';

}

if (!isset($\_SESSION['roleID']) || $\_SESSION['roleID'] == 0){

if($isActive == 1){

header("Location: index.php");

}

}

if (isset($\_SESSION['success'])) {

$order = new Order($offerID, $\_SESSION['userID']);

$hasActiveOrders = $order->checkOrdersStatus();

}

if (isset($\_POST['delete\_configuration'])) {

$detailsID = $\_POST['detailsID'];

$offer = new Offer();

$result = $offer->deleteConfiguration($detailsID);

$\_SESSION['showError'] = !$result;

header("Location: offerPage.php?offerID=$offerID&detailsID=$detailsID");

exit();

}

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

if (isset($\_POST["deactivate\_offer"])) {

$result = $offer->deactivateOffer($detailsID);

header("Location: offerPage.php?offerID=$offerID&detailsID=$detailsID");

} elseif (isset($\_POST["activate\_offer"])) {

$result = $offer->activateOffer($detailsID);

header("Location: offerPage.php?offerID=$offerID&detailsID=$detailsID");

}

}

?>

<script>

$(document).ready(function(){

$('button[name="delete\_configuration"]').click(function(e){

e.preventDefault();

var form = $(this).closest('form');

var detailsID = form.find('input[name="detailsID"]').val();

var offerID = form.find('input[name="offerID"]').val();

var deleteConfirmationText = "Are you sure you want to delete this configuration?";

swal({

title: "Delete Confirmation",

text: deleteConfirmationText,

icon: "warning",

buttons: ["Cancel", "Delete"],

dangerMode: true,

}).then((willDelete) => {

if (willDelete) {

$('<input>').attr({

type: 'hidden',

name: 'delete\_configuration',

value: true

}).appendTo(form);

form.unbind('submit').submit();

}

});

});

});

</script>

<?php

if (isset($\_SESSION['success'])) {

if ($hasActiveOrders) {

?>

<br>

<p class="btn" style="margin-top: -0.5%">Wait for a response to Your previous order.</p>

<?php

} else {

if ($isActive == 0): ?>

<input type="button" value="Get an offer" class="btn btn-primary" onclick="on()"></input>

<?php endif;

}

} else {

?>

<p class="btn" style="margin-top: 10%">You need to <a href="loginPage.php">log in</a> to make an offer.</p>

<?php

}

?>

**//Konfigurācijas pievienošāna**

<?php

session\_start();

require\_once '../connection.php';

require\_once '../Offer.php';

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$color = $\_POST['color'];

$color\_price = $\_POST['color\_price'];

$transmission\_type = $\_POST['transmission\_type'];

$transmission\_price = $\_POST['transmission\_price'];

$engine\_type = $\_POST['engine\_type'];

$engine\_price = $\_POST['engine\_price'];

$detailsID = $\_POST['detailsID'];

$offersInfoID = $\_POST['offersInfoID'];

$offerID = $\_POST['offerID'];

$imageFilePath = '';

if ($\_FILES['car\_image']['error'] === UPLOAD\_ERR\_OK) {

$imageFileName = $\_FILES['car\_image']['name'];

$imageFilePath = 'img/' . $imageFileName;

move\_uploaded\_file($\_FILES['car\_image']['tmp\_name'], $imageFilePath);

}

$offer = new Offer();

$result = $offer->addConfiguration($detailsID, $offersInfoID, $color, $imageFilePath, $color\_price, $transmission\_type, $transmission\_price, $engine\_type, $engine\_price);

if ($result) {

header("Location: ../offerPage.php?offerID=$offerID&detailsID=$detailsID");

exit();

} else {

echo "Error adding configuration.";

}

}

?>

**//Piedāvājuma informācijas rediģēšana**

<script>

var modal = document.getElementById('settingsModal');

function openModal() {

modal.style.display = 'block';

document.getElementById('editTab').classList.add('activeSetTab');

document.getElementById('addConfigTab').classList.remove('activeSetTab');

document.getElementById('manageOptionsForm').classList.remove('activeSetTab');

document.getElementById('editForm').style.display = 'block';

document.getElementById('addConfigForm').style.display = 'none';

document.getElementById('manageOptionsForm').style.display = 'none';

}

function closeModal() {

modal.style.display = 'none';

}

function showTab(tabName) {

if (tabName === 'edit') {

document.getElementById('editForm').style.display = 'block';

document.getElementById('addConfigForm').style.display = 'none';

document.getElementById('manageOptionsForm').style.display = 'none';

document.getElementById('editTab').classList.add('activeSetTab');

document.getElementById('addConfigTab').classList.remove('activeSetTab');

document.getElementById('manageOptionsTab').classList.remove('activeSetTab');

} else if (tabName === 'addConfig') {

document.getElementById('editForm').style.display = 'none';

document.getElementById('addConfigForm').style.display = 'block';

document.getElementById('manageOptionsForm').style.display = 'none';

document.getElementById('addConfigTab').classList.add('activeSetTab');

document.getElementById('editTab').classList.remove('activeSetTab');

document.getElementById('manageOptionsTab').classList.remove('activeSetTab');

} else if (tabName === 'manageOptions') {

document.getElementById('editForm').style.display = 'none';

document.getElementById('addConfigForm').style.display = 'none';

document.getElementById('manageOptionsForm').style.display = 'block';

document.getElementById('manageOptionsTab').classList.add('activeSetTab');

document.getElementById('editTab').classList.remove('activeSetTab');

document.getElementById('addConfigTab').classList.remove('activeSetTab');

}

}

</script>

<?php

session\_start();

require\_once 'connection.php';

require\_once 'Offer.php';

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$type = isset($\_POST['type']) ? $\_POST['type'] : null;

$manufacturer = isset($\_POST['manufacturer']) ? $\_POST['manufacturer'] : null;

$body\_type = isset($\_POST['body\_type']) ? $\_POST['body\_type'] : null;

$transmission\_type = isset($\_POST['transmission\_type']) ? $\_POST['transmission\_type'] : null;

$transmission\_price = isset($\_POST['transmission\_price']) ? $\_POST['transmission\_price'] : null;

$engine\_type = isset($\_POST['engine\_type']) ? $\_POST['engine\_type'] : null;

$engine\_price = isset($\_POST['engine\_price']) ? $\_POST['engine\_price'] : null;

$color = isset($\_POST['color']) ? $\_POST['color'] : null;

$color\_price = isset($\_POST['color\_price']) ? $\_POST['color\_price'] : null;

$price = isset($\_POST['price']) ? $\_POST['price'] : null;

$weight = isset($\_POST['weight']) ? $\_POST['weight'] : null;

$yearOfManufacture = isset($\_POST['yearOfManufacture']) ? $\_POST['yearOfManufacture'] : null;

$offerID = $\_POST['offerID'];

$detailsID = $\_POST['detailsID'];

if (empty($type) && empty($manufacturer) && empty($body\_type) && empty($transmission\_type) && empty($transmission\_price) && empty($engine\_type) && empty($engine\_price) && empty($color) && empty($color\_price) && empty($price) && empty($weight) && empty($yearOfManufacture)) {

echo "<script>

window.history.back();

</script>";

exit();

}

if ($\_FILES['car\_image']['error'] === UPLOAD\_ERR\_OK) {

$imageFileName = $\_FILES['car\_image']['name'];

$imageFilePath = 'img/' . $imageFileName;

move\_uploaded\_file($\_FILES['car\_image']['tmp\_name'], $imageFilePath);

} else {

$imageFilePath = null;

}

$offer = new Offer();

$currentOfferInfo = $offer->getOffer($offerID);

$updatedData = [

'type' => $type ?? $currentOfferInfo['type'],

'manufacturer' => $manufacturer ?? $currentOfferInfo['manufacturer'],

'body\_type' => $body\_type ?? $currentOfferInfo['body\_type'],

'transmission\_type' => $transmission\_type ?? $currentOfferInfo['transmission\_type'],

'transmission\_price' => $transmission\_price ?? $currentOfferInfo['transmission\_price'],

'engine\_type' => $engine\_type ?? $currentOfferInfo['engine\_type'],

'engine\_price' => $engine\_price ?? $currentOfferInfo['engine\_price'],

'color' => $color ?? $currentOfferInfo['color'],

'color\_price' => $color\_price ?? $currentOfferInfo['color\_price'],

'price' => $price ?? $currentOfferInfo['price'],

'weight' => $weight ?? $currentOfferInfo['weight'],

'yearOfManufacture' => $yearOfManufacture ?? $currentOfferInfo['yearOfManufacture']

];

$result = $offer->updateOfferInformation($offerID, $detailsID, $updatedData['type'], $updatedData['manufacturer'], $updatedData['body\_type'], $updatedData['transmission\_type'], $updatedData['transmission\_price'], $updatedData['engine\_type'], $updatedData['engine\_price'], $updatedData['color'], $updatedData['color\_price'], $updatedData['price'], $updatedData['weight'], $updatedData['yearOfManufacture'], $imageFilePath);

header("Location: offerPage.php?offerID=$offerID&detailsID=$detailsID");

exit();

}

?>

**//Piedāvājuma klase**

<?php

require\_once 'connection.php';

class Offer {

private $conn;

private $offerID;

private $type;

private $manufacturer;

private $image;

private $offersInfoID;

private $color;

private $price;

private $yearOfManufacture;

private $weight;

public function \_\_construct() {

$this->conn = (new Database())->connect();

}

public function getOffer($offerID) {

$sql = "SELECT \* FROM offers

JOIN offersinfo ON offers.offerID = offersinfo.offersID

JOIN specific\_details ON offersinfo.offersInfoID = specific\_details.offersInfoID

JOIN car\_colors ON specific\_details.colorID = car\_colors.colorID

JOIN transmission ON specific\_details.transmissionID = transmission.transmissionID

JOIN engine ON specific\_details.engineID = engine.engineID

WHERE offerID = ?";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$offerID]);

$result = $stmt->fetch(PDO::FETCH\_ASSOC);

return $result;

}

public function getOffersInfoID($offerID) {

$sql = "SELECT offersInfoID FROM offersinfo WHERE offersID = ?";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$offerID]);

$result = $stmt->fetch(PDO::FETCH\_ASSOC);

if ($result) {

return $result['offersInfoID'];

} else {

return null;

}

}

public function getOfferInfo($offersInfoID) {

$sql = "SELECT \* FROM offersinfo WHERE offersID = ?";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$offersInfoID]);

$result = $stmt->fetch(PDO::FETCH\_ASSOC);

return $result;

}

public function getOfferTransmission($offersInfoID, $detailsID) {

$sql = "SELECT \* FROM transmission

INNER JOIN specific\_details ON specific\_details.transmissionID = transmission.transmissionID

INNER JOIN offersinfo ON specific\_details.offersInfoID = offersinfo.offersInfoID

WHERE offersinfo.offersInfoID = ? AND specific\_details.detailsID = ?";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$offersInfoID, $detailsID]);

$result = $stmt->fetch(PDO::FETCH\_ASSOC);

return $result;

}

public function getOfferEngine($offersInfoID, $detailsID) {

$sql = "SELECT \* FROM engine

INNER JOIN specific\_details ON specific\_details.engineID = engine.engineID

INNER JOIN offersinfo ON specific\_details.offersInfoID = offersinfo.offersInfoID

WHERE offersinfo.offersInfoID = ? AND specific\_details.detailsID = ?";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$offersInfoID, $detailsID]);

$result = $stmt->fetch(PDO::FETCH\_ASSOC);

return $result;

}

public function getOfferColor($offersInfoID, $detailsID) {

$sql = "SELECT \* FROM car\_colors

INNER JOIN specific\_details ON specific\_details.colorID = car\_colors.colorID

INNER JOIN offersinfo ON specific\_details.offersInfoID = offersinfo.offersInfoID

WHERE offersinfo.offersInfoID = ? AND specific\_details.detailsID = ?";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$offersInfoID, $detailsID]);

$result = $stmt->fetch(PDO::FETCH\_ASSOC);

return $result;

}

public function getOfferDetails($offersInfoID) {

$sql = "SELECT car\_colors.\*, specific\_details.\*, transmission.\*, offersinfo.\*, engine.\*

FROM specific\_details

INNER JOIN car\_colors ON specific\_details.colorID = car\_colors.colorID

INNER JOIN transmission ON specific\_details.transmissionID = transmission.transmissionID

INNER JOIN engine ON specific\_details.engineID = engine.engineID

INNER JOIN offersinfo on offersinfo.offersInfoID = specific\_details.offersInfoID

WHERE specific\_details.offersInfoID = ?";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$offersInfoID]);

$result = $stmt->fetchAll(PDO::FETCH\_ASSOC);

return $result;

}

public function getOfferDetailsByID($offersInfoID) {

$sql = "SELECT specific\_details.\*, car\_colors.\*, transmission.\*, offersinfo.\*, engine.\*

FROM specific\_details

INNER JOIN car\_colors ON specific\_details.colorID = car\_colors.colorID

INNER JOIN transmission ON specific\_details.transmissionID = transmission.transmissionID

INNER JOIN engine ON specific\_details.engineID = engine.engineID

INNER JOIN offersinfo on offersinfo.offersInfoID = specific\_details.offersInfoID

WHERE offersinfo.offersInfoID = ?";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$offersInfoID]);

$result = $stmt->fetchAll(PDO::FETCH\_ASSOC);

return $result;

}

public function getAllOffers() {

$sql = "SELECT offers.\*, offersinfo.\*, specific\_details.\*, car\_colors.\*, transmission.\*, engine.\*

FROM offers

JOIN offersinfo ON offers.offerID = offersinfo.offersID

JOIN specific\_details ON offersinfo.offersInfoID = specific\_details.offersInfoID

JOIN car\_colors ON specific\_details.colorID = car\_colors.colorID

JOIN transmission ON specific\_details.transmissionID = transmission.transmissionID

JOIN engine ON specific\_details.engineID = engine.engineID

ORDER BY manufacturer, type";

$stmt = $this->conn->prepare($sql);

$stmt->execute();

$result = $stmt->fetchAll(PDO::FETCH\_ASSOC);

return $result;

}

public function addOffer($data) {

$imageFileName = $\_FILES['image']['name'];

$imageFilePath = 'img/' . $imageFileName;

move\_uploaded\_file($\_FILES['image']['tmp\_name'], $imageFilePath);

$this->conn->beginTransaction();

try {

$sql = "INSERT INTO offers (type, manufacturer) VALUES (?, ?)";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$data['type'], $data['manufacturer']]);

$offerID = $this->conn->lastInsertId();

$sql = "INSERT INTO car\_colors (color, image, color\_price) VALUES (?, ?, ?)";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$data['color'], $imageFilePath, $data['color\_price']]);

$colorID = $this->conn->lastInsertId();

$sql = "INSERT INTO transmission (transmission\_type, transmission\_price) VALUES (?, ?)";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$data['transmission\_type'], $data['transmission\_price']]);

$transmissionID = $this->conn->lastInsertId();

$sql = "INSERT INTO engine (engine\_type, engine\_price) VALUES (?, ?)";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$data['engine\_type'], $data['engine\_price']]);

$engineID = $this->conn->lastInsertId();

$sql = "INSERT INTO offersinfo (offersID, price, yearOfManufacture, weight, body\_type) VALUES (?, ?, ?, ?, ?)";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$offerID, $data['price'], $data['yearOfManufacture'], $data['weight'], $data['body\_type']]);

$offersInfoID = $this->conn->lastInsertId();

$sql = "INSERT INTO specific\_details (offersInfoID, colorID, transmissionID, created\_at, engineID) VALUES (?, ?, ?, NOW(), ?)";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$offersInfoID, $colorID, $transmissionID, $engineID]);

$this->conn->commit();

$\_SESSION['offer\_add\_success'] = "Offer added successfully.";

} catch (PDOException $e) {

$this->conn->rollback();

echo "Error: " . $e->getMessage();

}

}

public function updateOfferInformation($offerID, $detailsID, $type, $manufacturer, $body\_type, $transmission\_type, $transmission\_price, $engine\_type, $engine\_price, $color, $color\_price, $price, $weight, $yearOfManufacture, $imageFilePath) {

$this->conn->beginTransaction();

try {

$setValues = [];

$params = [];

if (isset($type)) {

$setValues[] = 'offers.type = ?';

$params[] = $type;

}

if (isset($manufacturer)) {

$setValues[] = 'offers.manufacturer = ?';

$params[] = $manufacturer;

}

if (isset($body\_type)) {

$setValues[] = 'offersinfo.body\_type = ?';

$params[] = $body\_type;

}

if (isset($transmission\_type)) {

$setValues[] = 'transmission.transmission\_type = ?';

$params[] = $transmission\_type;

}

if (isset($transmission\_price) || $transmission\_price === '0') {

$setValues[] = 'transmission.transmission\_price = ?';

$params[] = $transmission\_price;

}

if (isset($engine\_type)) {

$setValues[] = 'engine.engine\_type = ?';

$params[] = $engine\_type;

}

if (isset($engine\_price) || $engine\_price === '0') {

$setValues[] = 'engine.engine\_price = ?';

$params[] = $engine\_price;

}

if (isset($color)) {

$setValues[] = 'car\_colors.color = ?';

$params[] = $color;

}

if (isset($color\_price) || $color\_price === '0') {

$setValues[] = 'car\_colors.color\_price = ?';

$params[] = $color\_price;

}

if (isset($price) || $price === '0') {

$setValues[] = 'offersinfo.price = ?';

$params[] = $price;

}

if (isset($weight) || $weight === '0') {

$setValues[] = 'offersinfo.weight = ?';

$params[] = $weight;

}

if (isset($yearOfManufacture)) {

$setValues[] = 'offersinfo.yearOfManufacture = ?';

$params[] = $yearOfManufacture;

}

if (!empty($imageFilePath)) {

$setValues[] = 'car\_colors.image = ?';

$params[] = $imageFilePath;

}

$setPart = implode(', ', $setValues);

$params[] = $offerID;

$params[] = $detailsID;

$sql = "UPDATE offers

INNER JOIN offersinfo ON offers.offerID = offersinfo.offersID

INNER JOIN specific\_details ON offersinfo.offersInfoID = specific\_details.offersInfoID

INNER JOIN car\_colors ON specific\_details.colorID = car\_colors.colorID

INNER JOIN transmission ON specific\_details.transmissionID = transmission.transmissionID

INNER JOIN engine ON specific\_details.engineID = engine.engineID

SET $setPart

WHERE offers.offerID = ? AND specific\_details.detailsID = ?";

$stmt = $this->conn->prepare($sql);

$stmt->execute($params);

$this->conn->commit();

return $stmt->rowCount() > 0;

} catch (PDOException $e) {

$this->conn->rollback();

echo "Error: " . $e->getMessage();

return false;

}

}

public function addConfiguration($detailsID, $offersInfoID, $color, $image, $color\_price, $transmission\_type, $transmission\_price, $engine\_type, $engine\_price) {

$sql = "INSERT INTO car\_colors (color, image, color\_price) VALUES (?, ?, ?)";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$color, $image, $color\_price]);

$colorID = $this->conn->lastInsertId();

$sql = "INSERT INTO transmission (transmission\_type, transmission\_price) VALUES (?, ?)";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$transmission\_type, $transmission\_price]);

$transmissionID = $this->conn->lastInsertId();

$sql = "INSERT INTO engine (engine\_type, engine\_price) VALUES (?, ?)";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$engine\_type, $engine\_price]);

$engineID = $this->conn->lastInsertId();

$sql = "INSERT INTO specific\_details (offersInfoID, colorID, transmissionID, created\_at, engineID)

VALUES (?, ?, ?, NOW(), ?)";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$offersInfoID, $colorID, $transmissionID, $engineID]);

return true;

}

public function checkOrderConnection($detailsID) {

$sql = "SELECT COUNT(\*) AS totalOrders FROM `order` WHERE orderDetailsID = :detailsID";

$stmt = $this->conn->prepare($sql);

$stmt->bindParam(':detailsID', $detailsID, PDO::PARAM\_INT);

$stmt->execute();

$result = $stmt->fetch(PDO::FETCH\_ASSOC);

return $result['totalOrders'];

}

public function deleteConfiguration($detailsID) {

try {

$totalOrders = $this->checkOrderConnection($detailsID);

if ($totalOrders == 0) {

$sql = "SELECT colorID, transmissionID, engineID FROM specific\_details WHERE detailsID = ?";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$detailsID]);

$configInfo = $stmt->fetch(PDO::FETCH\_ASSOC);

$sql = "DELETE FROM specific\_details WHERE detailsID = ?";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$detailsID]);

$sql = "DELETE FROM car\_colors WHERE colorID = ?";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$configInfo['colorID']]);

$sql = "DELETE FROM transmission WHERE transmissionID = ?";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$configInfo['transmissionID']]);

$sql = "DELETE FROM engine WHERE engineID = ?";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$configInfo['engineID']]);

header("Location: index.php");

exit();

return true;

} else {

return false;

}

} catch (PDOException $e) {

echo "Error: " . $e->getMessage();

return false;

}

}

public function deactivateOffer($detailsID) {

$sql = "UPDATE specific\_details SET active\_status = 1 WHERE detailsID = ?";

$stmt = $this->conn->prepare($sql);

return $stmt->execute([$detailsID]);

}

public function activateOffer($detailsID) {

$sql = "UPDATE specific\_details SET active\_status = 0 WHERE detailsID = ?";

$stmt = $this->conn->prepare($sql);

return $stmt->execute([$detailsID]);

}

public function isOfferActive($detailsID) {

$sql = "SELECT active\_status FROM specific\_details WHERE detailsID = ?";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$detailsID]);

$result = $stmt->fetch(PDO::FETCH\_ASSOC);

return $result['active\_status'] ?? false;

}

}

?>

**//Pasūtījumu veikšana**

if (isset($\_SESSION['success'])) {

$order = new Order($offerID, $\_SESSION['userID']);

$hasActiveOrders = $order->checkOrdersStatus();

}

if (isset($\_POST['submit\_order'])) {

$name = $\_POST['name'];

$surname = $\_POST['surname'];

$telephone = $\_POST['telephone'];

$detailsID = $\_POST['detailsID'];

echo "<script>event.preventDefault();</script>";

if (isset($\_SESSION['userID'])) {

$order = new Order();

$order->createOrder($\_POST['name'], $\_POST['surname'], $\_POST['telephone'], $\_POST['offerID'], $\_POST['detailsID']);

} else {

}

}

<script>

function showConfirmationModal() {

var name = document.getElementById("name").value;

var surname = document.getElementById("surname").value;

var telephone = document.getElementById("telephone").value;

var termsChecked = document.getElementById("check").checked;

if (name.length >= 3 && surname.length >= 3 && telephone.length >= 5 && termsChecked) {

document.getElementById("confirmation\_modal").style.display = "block";

document.getElementById("confirmation\_name").innerText = name;

document.getElementById("confirmation\_surname").innerText = surname;

document.getElementById("confirmation\_telephone").innerText = telephone;

} else {

var errorText = "Please ensure all fields meet the minimum length requirements and accept the terms and conditions:\n\n"

+ "Name: at least 3 characters\n"

+ "Surname: at least 3 characters\n"

+ "Telephone: at least 5 characters\n"

+ "Acceptance of terms and conditions";

swal("Error", errorText, "error");

}

}

function closeConfirmationModal() {

document.getElementById("confirmation\_modal").style.display = "none";

}

function scrollDown() {

var modalContent = document.querySelector('.confirmation-modal-content');

var scrollAmount = modalContent.scrollHeight - modalContent.scrollTop - modalContent.clientHeight;

modalContent.scrollBy({

top: scrollAmount,

behavior: 'smooth'

});

var isScrolledToBottom = modalContent.scrollHeight - modalContent.scrollTop === modalContent.clientHeight;

var scrollDownBtn = document.querySelector('.scroll-down-btn');

if (isScrolledToBottom) {

scrollDownBtn.classList.add('hidden');

} else {

scrollDownBtn.classList.remove('hidden');

}

}

document.querySelector('.confirmation-modal-content').addEventListener('scroll', function() {

var modalContent = this;

var isScrolledToBottom = modalContent.scrollHeight - modalContent.scrollTop === modalContent.clientHeight;

var scrollDownBtn = document.querySelector('.scroll-down-btn');

if (isScrolledToBottom) {

scrollDownBtn.classList.add('hidden');

} else {

scrollDownBtn.classList.remove('hidden');

}

});

</script>

**//Pasūtījumu izpildes statusa maiņa**

<?php

session\_start();

require\_once 'connection.php';

require\_once 'Order.php';

require\_once 'User.php';

$user = new UserMain($\_SESSION['userID']);

$userInfo = $user->getUserInfo();

if ($\_SESSION['roleID'] !== 1) {

header('Location: index.php');

exit();

}

$order = new Order();

$orders = $order->getAllOrderInfo();

if (isset($\_POST['submit'])) {

if (empty($status)) {

foreach ($\_POST['status'] as $orderID => $status) {

$order = new Order();

$order->updateStatus($status, $orderID);

$orders = $order->getAllOrderInfo();

}

}

}

$newOrders = array();

$inProgressOrders = array();

$doneOrders = array();

foreach($orders as $order){

if($order['status'] == 'New'){

array\_push($newOrders, $order);

}

elseif($order['status'] == 'In progress'){

array\_push($inProgressOrders, $order);

}

elseif($order['status'] == 'Done'){

array\_push($doneOrders, $order);

}

}

if (isset($\_POST['deleteOrder'])) {

$order = new Order();

$order->deleteOrder($\_POST['orderID']);

}

?>

<script>

function showTab(tabName, event) {

event.preventDefault();

// Hide all tab contents

const tabs = document.querySelectorAll('.tabContent');

tabs.forEach(tab => {

tab.style.display = 'none';

});

const tabButtons = document.querySelectorAll('.setTabButton');

tabButtons.forEach(button => {

button.classList.remove('activeSetTab');

});

document.getElementById(tabName + 'Table').style.display = 'block';

event.currentTarget.classList.add('activeSetTab');

}

document.addEventListener('DOMContentLoaded', () => {

document.getElementById('newTab').click();

});

</script>

**//Pasūtījumu klase**

<?php

require\_once 'connection.php';

require\_once 'Offer.php';

require\_once 'User.php';

class Order {

private $conn;

private $orderID;

private $orderDate;

private $name;

private $surname;

private $telephone;

private $status;

private $orderUserID;

private $orderOfferID;

public function \_\_construct() {

$this->conn = (new Database())->connect();

}

public function getStatus() {

return $this->status;

}

public function updateStatus($status, $orderID) {

if (!empty($status)) {

$this->orderID = $orderID;

$this->status = $status;

$sql = "UPDATE `order` SET `status` = :status WHERE `orderID` = :orderID";

$stmt = $this->conn->prepare($sql);

$stmt->bindValue(':status', $this->status);

$stmt->bindValue(':orderID', $this->orderID, PDO::PARAM\_INT);

if ($stmt->execute()) {

$\_SESSION['order\_status\_success'] = "Order status changed successfully.";

} else {

echo "Error updating order status: " . $stmt->errorInfo()[2];

}

}

}

public function createOrder($name, $surname, $telephone, $offerID, $detailsID) {

if (strlen($name) > 20 || strlen($surname) > 20 || strlen($telephone) > 20) {

return false;

}

$this->status = 'New';

date\_default\_timezone\_set('Europe/Riga');

$this->orderDate = date("Y-m-d H:i:s");

$this->name = $name;

$this->surname = $surname;

$this->telephone = $telephone;

$this->orderOfferID = $offerID;

$this->orderUserID = $\_SESSION['userID'];

if(isset($\_SESSION["userID"])) {

$sql = "INSERT INTO `order` (`orderDate`, `status`, `orderOfferID`, `orderUserID`, `name`, `surname`, `telephone`, `orderDetailsID`)

VALUES (:orderDate, :status, :orderOfferID, :orderUserID, :name, :surname, :telephone, :detailsID)";

$stmt = $this->conn->prepare($sql);

$stmt->bindValue(':orderDate', $this->orderDate);

$stmt->bindValue(':status', $this->status);

$stmt->bindValue(':orderOfferID', $this->orderOfferID);

$stmt->bindValue(':orderUserID', $this->orderUserID);

$stmt->bindValue(':name', $this->name);

$stmt->bindValue(':surname', $this->surname);

$stmt->bindValue(':telephone', $this->telephone);

$stmt->bindValue(':detailsID', $detailsID);

if ($stmt->execute()) {

$\_SESSION['order\_success'] = "Your order has been sent successfully.";

$userMain = new UserMain($this->orderUserID);

$topicName = 'Orders';

$topicID = $userMain->getNotificationTopicIDByName($topicName);

$notificationText = "Your order has been successfully completed! Please wait while our staff contacts You. You can check your order <a href='profile.php'>here</a> in the 'My Orders' tab.";

$userMain->addNotification($topicID, $notificationText);

header("Location: index.php");

} else {

echo "Error: " . $sql . "<br>" . $conn->error;

}

} else {

header('location: loginPage.php');

}

}

public function deleteOrder($orderID) {

$sql = "DELETE FROM `order` WHERE `orderID` = :orderID";

$stmt = $this->conn->prepare($sql);

$stmt->bindValue(':orderID', $orderID, PDO::PARAM\_INT);

if ($stmt->execute()) {

$\_SESSION['order\_delete\_success'] = "Order deleted successfully.";

header("Location: ordersPage.php");

} else {

echo "Error deleting order: " . $stmt->errorInfo()[2];

}

}

public function getAllOrderInfo() {

$sql = "SELECT o.\*, u.\*, off.\*, car\_colors.\*, offInf.\*, specific\_details.\*, transmission.\*, engine.\*

FROM `order` o

INNER JOIN `user` u ON o.orderUserID = u.userID

INNER JOIN `offers` off ON o.orderOfferID = off.offerID

INNER JOIN `offersinfo` offInf ON off.offerID = offInf.offersID

INNER JOIN `specific\_details` ON o.orderDetailsID = specific\_details.detailsID

INNER JOIN `transmission` ON transmission.transmissionID = specific\_details.transmissionID

INNER JOIN `engine` ON engine.engineID = specific\_details.engineID

INNER JOIN `car\_colors` ON specific\_details.colorID = car\_colors.colorID

ORDER BY `orderID` DESC";

$stmt = $this->conn->prepare($sql);

$stmt->execute();

$orders = $stmt->fetchAll(PDO::FETCH\_ASSOC);

return $orders;

}

public function getOrderInfo($userID) {

$this->orderUserID = $userID;

$sql = "SELECT o.\*, u.\*, off.\*, car\_colors.\*, offInf.\*, specific\_details.\*, transmission.\*, engine.\*

FROM `order` o

INNER JOIN `user` u ON o.orderUserID = u.userID

INNER JOIN `offers` off ON o.orderOfferID = off.offerID

INNER JOIN `offersinfo` offInf ON off.offerID = offInf.offersID

INNER JOIN `specific\_details` ON o.orderDetailsID = specific\_details.detailsID

INNER JOIN `transmission` ON transmission.transmissionID = specific\_details.transmissionID

INNER JOIN `engine` ON engine.engineID = specific\_details.engineID

INNER JOIN `car\_colors` ON specific\_details.colorID = car\_colors.colorID

WHERE o.orderUserID = ?

ORDER BY o.orderDate DESC";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$this->orderUserID]);

$orders = $stmt->fetchAll(PDO::FETCH\_ASSOC);

return $orders;

}

public function getOrderSum($userID) {

$this->orderUserID = $userID;

$sql = "SELECT SUM(offInf.price) + SUM(car\_colors.color\_price) + SUM(transmission.transmission\_price) + SUM(engine.engine\_price) as totalPrice

FROM `order` o

LEFT JOIN `user` u ON o.orderUserID = u.userID

LEFT JOIN `offers` off ON o.orderOfferID = off.offerID

LEFT JOIN `offersinfo` offInf ON off.offerID = offInf.offersID

INNER JOIN `specific\_details` ON o.orderDetailsID = specific\_details.detailsID

INNER JOIN `transmission` ON transmission.transmissionID = specific\_details.transmissionID

INNER JOIN `engine` ON engine.engineID = specific\_details.engineID

INNER JOIN `car\_colors` ON specific\_details.colorID = car\_colors.colorID

WHERE o.orderUserID = ?";

$stmt = $this->conn->prepare($sql);

$stmt->execute([$this->orderUserID]);

$sum = $stmt->fetch(PDO::FETCH\_ASSOC);

return $sum['totalPrice'];

}

public function checkOrdersStatus() {

$this->orderUserID = $\_SESSION['userID'];

$query = "SELECT COUNT(\*) as count FROM `order` WHERE orderUserID=:userID AND (status='New' OR status='In progress')";

$stmt = $this->conn->prepare($query);

$stmt->bindParam(':userID', $this->orderUserID);

$stmt->execute();

$result = $stmt->fetch(PDO::FETCH\_ASSOC);

return $result['count'];

}

public function setUserIDFromSession($userID) {

$this->orderUserID = $userID;

}

}

?>

**//Pasūtījumu tabulas izvadīšana Excel failā**

<?php

require '../vendor/autoload.php'; // Путь к autoload.php из PhpSpreadsheet

use PhpOffice\PhpSpreadsheet\Spreadsheet;

use PhpOffice\PhpSpreadsheet\Writer\Xlsx;

if (isset($\_POST['export']) && $\_POST['export'] === 'true') {

if (isset($\_POST['orders\_data'])) {

$orders = unserialize($\_POST['orders\_data']);

} else {

echo "Error: Orders data is missing.";

exit;

}

$spreadsheet = new Spreadsheet();

$sheet = $spreadsheet->getActiveSheet();

$styleArray = [

'font' => [

'bold' => true,

'color' => ['rgb' => '000000'],

],

'fill' => [

'fillType' => \PhpOffice\PhpSpreadsheet\Style\Fill::FILL\_SOLID,

'startColor' => ['rgb' => 'FFFF00'],

],

];

$sheet->getStyle('A1:T1')->applyFromArray($styleArray);

$sheet->setCellValue('A1', 'Order ID');

$sheet->setCellValue('B1', 'Order Date');

$sheet->setCellValue('C1', 'Name');

$sheet->setCellValue('D1', 'Surname');

$sheet->setCellValue('E1', 'Telephone');

$sheet->setCellValue('F1', 'Username');

$sheet->setCellValue('G1', 'Email');

$sheet->setCellValue('H1', 'Manufacturer');

$sheet->setCellValue('I1', 'Type');

$sheet->setCellValue('J1', 'Year of Manufacture');

$sheet->setCellValue('K1', 'Body Type');

$sheet->setCellValue('L1', 'Transmission');

$sheet->setCellValue('M1', 'Engine Type');

$sheet->setCellValue('N1', 'Color');

$sheet->setCellValue('O1', 'Car Price');

$sheet->setCellValue('P1', 'Transmission Price');

$sheet->setCellValue('Q1', 'Engine Price');

$sheet->setCellValue('R1', 'Color Price');

$sheet->setCellValue('S1', 'Final Price');

$sheet->setCellValue('T1', 'Status');

$row = 2;

foreach ($orders as $order) {

$sheet->setCellValue('A' . $row, $order['orderID']);

$sheet->setCellValue('B' . $row, $order['orderDate']);

$sheet->setCellValue('C' . $row, $order['name']);

$sheet->setCellValue('D' . $row, $order['surname']);

$sheet->setCellValue('E' . $row, $order['telephone']);

$sheet->setCellValue('F' . $row, $order['username']);

$sheet->setCellValue('G' . $row, $order['email']);

$sheet->setCellValue('H' . $row, $order['manufacturer']);

$sheet->setCellValue('I' . $row, $order['type']);

$sheet->setCellValue('J' . $row, $order['yearOfManufacture']);

$sheet->setCellValue('K' . $row, $order['body\_type']);

$sheet->setCellValue('L' . $row, $order['transmission\_type']);

$sheet->setCellValue('M' . $row, $order['engine\_type']);

$sheet->setCellValue('N' . $row, $order['color']);

$sheet->setCellValue('O' . $row, $order['price'] . ' €');

$sheet->setCellValue('P' . $row, $order['transmission\_price'] . ' €');

$sheet->setCellValue('Q' . $row, $order['engine\_price'] . ' €');

$sheet->setCellValue('R' . $row, $order['color\_price'] . ' €');

$sheet->setCellValue('S' . $row, $order['price'] + $order['color\_price'] + $order['transmission\_price'] + $order['engine\_price'] . ' €');

$sheet->setCellValue('T' . $row, $order['status']);

$row++;

}

$writer = new \PhpOffice\PhpSpreadsheet\Writer\Xlsx($spreadsheet);

header('Content-Type: application/vnd.openxmlformats-officedocument.spreadsheetml.sheet');

header('Content-Disposition: attachment;filename="Orders.xlsx"');

header('Cache-Control: max-age=0');

$writer->save('php://output');

exit;

}

// ...

?>

<?php

require '../vendor/autoload.php';

require\_once '../Order.php';

use PhpOffice\PhpSpreadsheet\Spreadsheet;

use PhpOffice\PhpSpreadsheet\Writer\Xlsx;

if ($\_SERVER['REQUEST\_METHOD'] == 'POST' && isset($\_POST['export'])) {

if ($\_SERVER['REQUEST\_METHOD'] == 'POST' && isset($\_POST['export'])) {

$userID = $\_POST['userID'];

$totalSum = $\_POST['totalSum'];

}

$spreadsheet = new \PhpOffice\PhpSpreadsheet\Spreadsheet();

$sheet = $spreadsheet->getActiveSheet();

$styleArray = [

'font' => [

'bold' => true,

'color' => ['rgb' => '000000'],

],

'fill' => [

'fillType' => \PhpOffice\PhpSpreadsheet\Style\Fill::FILL\_SOLID,

'startColor' => ['rgb' => 'FFFF00'],

],

];

$sheet->getStyle('A1:U1')->applyFromArray($styleArray);

$sheet->setCellValue('A1', 'Order ID');

$sheet->setCellValue('B1', 'Order Date');

$sheet->setCellValue('C1', 'Name');

$sheet->setCellValue('D1', 'Surname');

$sheet->setCellValue('E1', 'Telephone');

$sheet->setCellValue('F1', 'Status');

$sheet->setCellValue('G1', 'Username');

$sheet->setCellValue('H1', 'Email');

$sheet->setCellValue('I1', 'Manufacturer');

$sheet->setCellValue('J1', 'Type');

$sheet->setCellValue('K1', 'Year of Manufacture');

$sheet->setCellValue('L1', 'Body Type');

$sheet->setCellValue('M1', 'Transmission');

$sheet->setCellValue('N1', 'Engine Type');

$sheet->setCellValue('O1', 'Color');

$sheet->setCellValue('P1', 'Car Price');

$sheet->setCellValue('Q1', 'Transmission Price');

$sheet->setCellValue('R1', 'Engine Price');

$sheet->setCellValue('S1', 'Color Price');

$sheet->setCellValue('T1', 'Final Price');

$sheet->setCellValue('U1', 'Total Price');

$order = new Order();

$orders = $order->getOrderInfo($userID);

$row = 2;

foreach ($orders as $order) {

$sheet->setCellValue('A' . $row, $order['orderID']);

$sheet->setCellValue('B' . $row, $order['orderDate']);

$sheet->setCellValue('C' . $row, $order['name']);

$sheet->setCellValue('D' . $row, $order['surname']);

$sheet->setCellValue('E' . $row, $order['telephone']);

$sheet->setCellValue('F' . $row, $order['status']);

$sheet->setCellValue('G' . $row, $order['username']);

$sheet->setCellValue('H' . $row, $order['email']);

$sheet->setCellValue('I' . $row, $order['manufacturer']);

$sheet->setCellValue('J' . $row, $order['type']);

$sheet->setCellValue('K' . $row, $order['yearOfManufacture']);

$sheet->setCellValue('L' . $row, $order['body\_type']);

$sheet->setCellValue('M' . $row, $order['transmission\_type']);

$sheet->setCellValue('N' . $row, $order['engine\_type']);

$sheet->setCellValue('O' . $row, $order['color']);

$sheet->setCellValue('P' . $row, $order['price'] . ' €');

$sheet->setCellValue('Q' . $row, $order['transmission\_price'] . ' €');

$sheet->setCellValue('R' . $row, $order['engine\_price'] . ' €');

$sheet->setCellValue('S' . $row, $order['color\_price'] . ' €');

$sheet->setCellValue('T' . $row, $order['price'] + $order['color\_price'] + $order['transmission\_price'] + $order['engine\_price'] . ' €');

$row++;

}

$sheet->setCellValue('U' . $row, $totalSum . ' €');

$writer = new \PhpOffice\PhpSpreadsheet\Writer\Xlsx($spreadsheet);

header('Content-Type: application/vnd.openxmlformats-officedocument.spreadsheetml.sheet');

header('Content-Disposition: attachment; filename="' . $order['username'] . '\_orders.xlsx"');

header('Cache-Control: max-age=0');

$writer->save('php://output');

exit;

}

?>

**//Lapas pāradresācija**

function RedToProfile() {

window.location.href="../autosalons/profile.php";

}

function RedToComments() {

window.location.href="../autosalons/index.php";

}

function RedToRegistration() {

window.location.href="../autosalons/registrationPage.php";

}

function RedToLogin() {

window.location.href="../autosalons/loginPage.php";

}

function RedToForum() {

window.location.href="../autosalons/forum.php";

}

function RedToHomepage() {

window.location.href="../autosalons/index.php";

}

function RedToOffer() {

window.location.href="../autosalons/offerPage.php";

}

function RedToInfo() {

window.location.href="../autosalons/infoPage.php";

}

function RedToContacts() {

window.location.href="../autosalons/contacts.php";

}

**//Iziešana no konta**

<?php

session\_start();

session\_unset();

session\_destroy();

header('Location: index.php?activity=log-out\_successful');

?>