

# **RĪGAS VALSTS TEHNIKUMS**

## **DATORIKAS NODAĻA**

Izglītības programma: Programmēšana

### **KVALIFIKĀCIJAS DARBS**

#### **Tīmekļa platforma automašīnu izīrēšanai**

Paskaidrojošais raksts 62 lpp.

Audzēknis:

Ivars Levāns

Prakses vadītājs:

Igors Litvjakovs

Nodaļas vadītājs:

Normunds Barbāns

**Rīga 2024**

# ANOTĀCIJA

Kvalifikācijas darbs raksturo procesu “Drive Wise” platformas izstrādē, kas nodrošinās plašu automašīnu katalogu nomas rezervācijām, kā arī padziļinātu platformas datu rediģēšanas funkcionalitāti administratoriem. Platformas izstrādes mērķis ir nodrošināt intuitīvu nomas procesu lietotājiem, marķējumus par sistēmā pieejamajiem automobiļiem, iespēju rezervēt noteiktus laika periodu, saņemt aprēķinu par pakalpojuma izmaksām un pārvaldīt veiktās rezervācijas datus. Frontend izstrādē tika izvēlēts Next.js, kas ļauj efektīvi izstrādāt servera pusē renderētas lietotnes, nodrošinot ātru lapas ielādi un uzlabotu lietotāja pieredzi. Tāpat tika izmantots Tailwind, kas nodrošina ātru un efektīvu stilizāciju. Backend izstrādē tika izmantots TypeScript, kas nodrošina statisko tipizāciju, REST API realizāciju, kopumā uzlabojot koda lasāmību un samazinot kļūdu skaitu izstrādes procesā. Kā datubāze tika izvēlēta PostgreSQL, kas ir pazīstama ar savu uzticamību un spēju apstrādāt lielus datu apjomus. Papildus tika izmantots Prisma ORM, kas atvieglo datu modeļu pārvaldību, kā arī nodrošina efektīvu datu mijiedarbību ar Next.js un PostgreSQL. Visual Studio Code tika izvēlēta kā izstrādes vide dēļ plašā rīku un paplašinājumu klāsta un valodu atbalsta.

Kvalifikācijas darba raksts sastāv no šādām daļām: ievads, uzdevuma nostādne, prasību specifikācija, uzdevuma risināšanas līdzekļu izvēles pamatojums, programmatūras produkta modelēšana un projektēšana, datu struktūru apskats, lietotāja ceļvedis, nobeigums un pielikumi. Ievadā tiek skaidrots, kāpēc ir nepieciešams izveidot minēto programmsistēmu, tās nozīmīgumu un praktisko pielietojumu. Uzdevuma nostādnē ir izklāstīti galvenie kvalifikācijas darba mērķi un specifiskie uzdevumi. Prasību specifikācijā ir definētas sistēmas funkcijas un drošības prasības, kā arī detalizēti aprakstīta informācija, ko sistēma saņem un izsniedz lietotājam. Uzdevuma risināšanas līdzekļu izvēles pamatojumā ir izskaidrots izvēlēto programmēšanas valodu, teksta redaktoru, relāciju datubāžu pārvaldības sistēmu un datu apmaiņas programmu izmantojums sistēmas izveidē. Programmatūras produkta modelēšanas un projektēšanas apskatā tiek pētītas datu plūsmas diagrammas, kas ilustrē sistēmas funkcionalitāti, ER diagrammas datubāzes struktūras vizualizācijai un sistēmas arhitektūras detalizēts apraksts. Datu struktūru apskats ietver visu datu tabulu struktūras aprakstu un attiecīgo tabulu savstarpējo saistību shēmu. Lietotāja ceļvedī ir izklāstīts sistēmas vizuālais noformējums un sniegti norādījumi par tās efektīvu izmantošanu.

Kopumā kvalifikācijas darba apjoms ir 62 lpp., kurā ietilpst 50 attēli, 4 tabulas un 14 pielikumi.

# ANNOTATION

The qualification paper describes the process of developing the Drive Wise platform, which will provide an extensive car rental catalogue for bookings as well as advanced editing functionality for administrators. The aim of the platform development is to provide an intuitive rental process for users, labelling of the cars available in the system, the possibility to book specific time periods, to get an estimate of the cost of the service and to manage the data of the booked reservation. For the frontend development, Next.js was chosen, which allows efficient development of server-side rendered applications, ensuring fast page loading and an improved user experience. Tailwind was also used, which provides fast and efficient styling. TypeScript was used for backend development, providing static typing, REST API implementation, improving overall code readability and reducing errors during development. PostgreSQL was chosen as the database, which is known for its reliability and ability to handle large amounts of data. Additionally, Prisma ORM was used to facilitate the management of data models, as well as to ensure efficient data interaction with Next.js and PostgreSQL. Visual Studio Code was chosen as the development environment due to its wide range of tools and extensions and language support.

The qualification paper consists of the following parts: introduction, problem statement, requirements specification, justification for the choice of the means of solving the problem, modelling and design of the software product, overview of data structures, user guide, conclusion and annexes. The introduction explains why it is necessary to develop the software system, its importance and practical applications. The assignment outlines the main objectives and specific tasks of the qualification. The requirements specification defines the functions and security requirements of the system and describes in detail the information that the system receives and provides to the user. The rationale for the choice of the means of solving the problem explains the use of the chosen programming languages, text editors, relational database management systems and data exchange programs in the development of the system. The software product modelling and design review examines data flow diagrams illustrating the functionality of the system, ER diagrams to visualise the database structure and a detailed description of the system architecture. The overview of data structures includes a description of the structure of all data tables and a diagram of the interrelationships between the relevant tables. A user guide outlines the visual design of the system and provides guidance on its effective use.

The total length of the qualification work is 62 pages, including 50 images, 4 tables and 14 attachments.

# SATURS

IEVADS .....	5
1. UZDEVUMA NOSTĀDNE .....	6
2. PRASĪBU SPECIFIKĀCIJA .....	8
2.1. Ieejas un izejas informācijas apraksts .....	8
2.1.1. Ieejas informācijas apraksts .....	8
2.1.2. Izejas informācijas apraksts .....	9
2.2. Funkcionālās prasības .....	9
2.3. Nefunkcionālās prasības .....	10
3. UZDEVUMU RISINĀŠANAS LĪDZEKĻU IZVĒLES PAMATOJUMS .....	12
4. PROGRAMMATŪRAS PRODUKTA MODELĒŠANA UN PROJEKTĒŠANA .....	13
4.1. Sistēmas struktūras modelis .....	13
4.1.1. Sistēmas arhitektūra .....	13
4.1.2. Sistēmas ER modelis .....	14
4.2. Funkcionālās sistēmas modelis .....	15
4.2.1. Datu plūsmu modelis .....	15
5. DATU STRUKTŪRAS APRAKSTS .....	19
5.1. Tabulu relāciju shēma .....	21
6. LIETOTĀJA CEĻVEDIS .....	22
6.1. Sistēmas prasības aparatūrai un programmatūrai .....	22
6.2. Sistēmas instalācija un palaišana .....	22
6.3. Programmas apraksts .....	23
6.3.1. Sākuma lapa .....	23
6.3.2. Autentifikācija .....	23
6.3.3. Rezervācijas izveides process .....	28
6.3.4. Lietotāju rezervāciju pārskats .....	31
6.3.5. Lietotāju uzstādījumi .....	33
6.3.6. Administrācijas panelis .....	33
6.4. Testa piemērs .....	35
NOBEIGUMS .....	42
INFORMĀCIJAS AVOTI .....	43
PIELIKUMI .....	44

## IEVADS

Mūsdienās ir ļoti svarīgi izmantot tehnoloģijas, lai uzlabotu mūsu dzīves kvalitāti un padarītu ikdienu efektīvāku. Šajā kontekstā vēlos izstrādāt risinājumu - Auto izīrēšanas platforma izmēģinājumu braucienu rezervēšanai.

Pašlaik, bez šīs sistēmas izmantošanas, cilvēkiem, kuriem ir interese iegādāties automašīnu, ir jāsaskaras ar vairākām sarežģītām problēmām. Tirgū pieejamā informācija par automašīnām var būt izkaisīta, nepilnīga un grūti salīdzināma. Lietotājiem trūkst skaidras priekšstata par automobiļu marku un modeļu plašo klāstu, un viņi ir spiesti pavadīt vairāk laika meklējot nepieciešamo informāciju.

Salīdzinot šīs platformas risinājumu ar esošajiem analogiem, ir skaidri novērojamas vairākas priekšrocības:

- luxrent.lv - piedāvā plašu automašīnu piedāvājumu, taču negarantē ērtu modeļu pārskatu un detalizētus aprakstus par modeļiem un to atšķirībām. Mērķis ir radīt platformu, kas nodrošinātu rūpīgi izstrādātu informāciju un skaidrus modeļu aprakstus, lai palīdzētu lietotājiem veikt labi informētu izvēli.
- rentclub.lv - piedāvā mašīnu nomas iespējas, taču mājaslapa ir grūti pielietojama. Mērķis ir radīt platformu, kura nodrošinātu vienkāršu un intuitīvu pieredzi, kas padara automašīnu izvēles procesu daudz saprotamāku.
- rigacars.lv - mājaslapa nepiedāvā pietiekami intuitīvu satura izkārtojumu. Mērķis ir radīt platformu ar skaidru automašīnu katalogu, lai lietotāji varētu viegli veikt salīdzinājumus un atrast vēlamo transportlīdzekli.

Šī auto tirdzniecības platforma ir paredzēta cilvēkiem, kuri vēlas iegādāties automašīnu, taču vēlas to darīt ar pēc iespējas mazākām šķēršļiem darījuma procesā. Vēlos piedāvāt vienkāršu un intuitīvu veidu, kā pārskatīt automobiļu klāstu, konfigurēt nomas datus pēc personīgiem kritērijiem un rezervēt automašīnu izmēģinājuma braucienam.

# 1. UZDEVUMA NOSTĀDNE

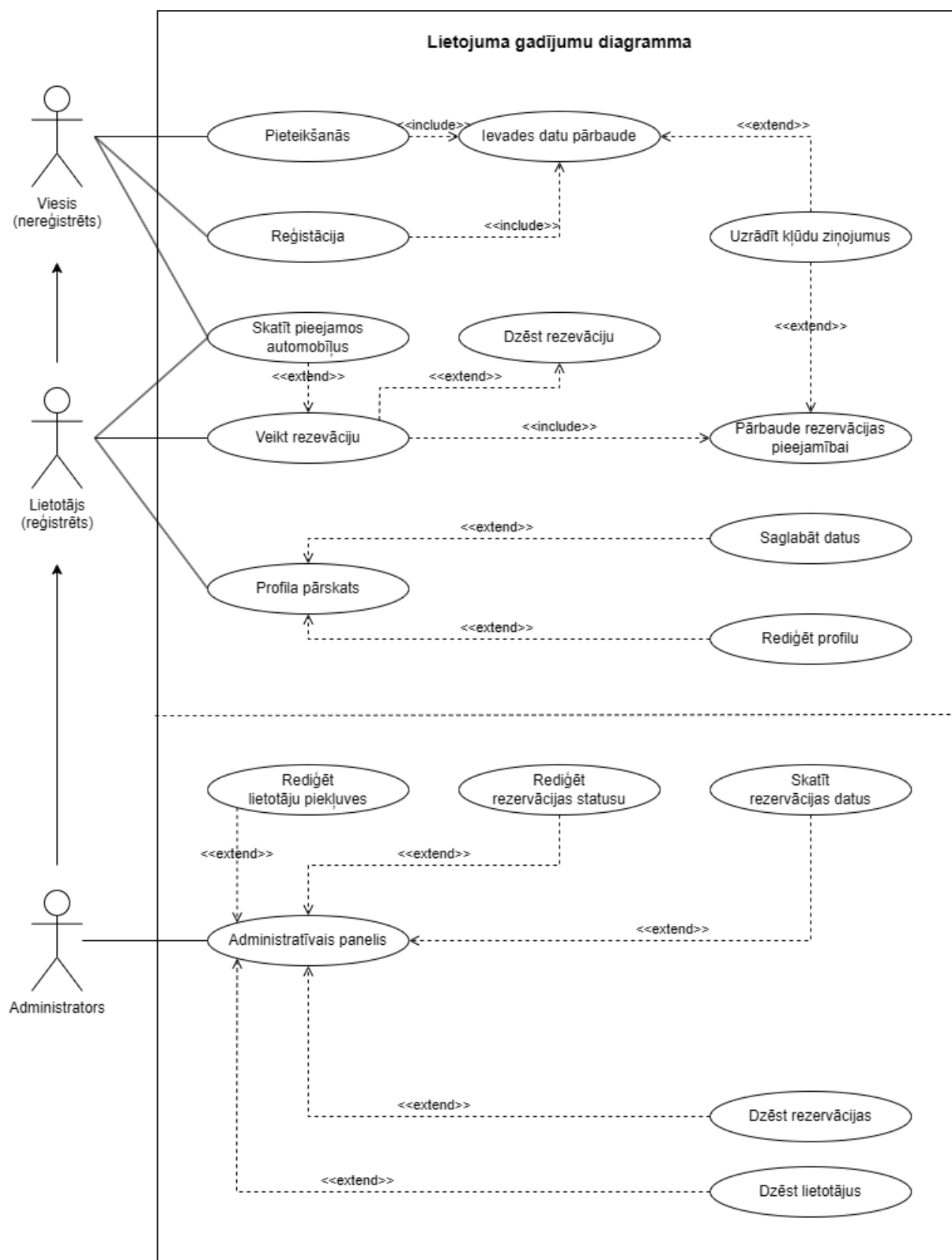
Kvalifikācijas darba uzdevums ir izveidot tīmekļa automašīnu izīrēšanas platformu. Sistēmā nepieciešams realizēt iespēju izvēlēties automašīnu markas un modeļus sasaistot klientu un pārdevēju intereses.

Pakalpojums ir tendēts cilvēkiem ar interesi automobiļa izīrēšanā. Šī platforma ir aktuāla, jo pašreizējais automašīnu īres platformu piedāvājums tirgū nav pietiekami organizēts un intuitīvs. Lielākā daļa esošo platformu nesniedz lietotājiem pietiekami skaidri prezentētu informāciju par automašīnām. Sniegtās informācijas kvalitāte klientiem rada liekas šaubas par izvēli un produktu atšķirībām. Šīs problēmas sekas izraisa nepieciešamību pavadīt vairāk laika meklējot un pārbaudot dažādās iespējas nenonākot pie galējā darījuma veikšanas.

## **Sistēmas pamatdarbības ietver:**

- automobiļu modeļu katalogs;
- datuma un laika rezervācija automašīnai;
- administratīva satura rediģēšana;
- administratīva datu pārskatīšana un dzēšana;
- lietotāju autorizācija platformā;
- lietotāja personīgo datu rediģēšana.

Paredzēts, ka sistēmu lietos trīs lietotāju klases: Administrators, Viesis un Lietotājs (skat. 1. att.).



1.1. att. Lietojumgadījuma diagramma

## 2. PRASĪBU SPECIFIKĀCIJA

### 2.1 Ieejas un izejas informācijas apraksts

#### 2.1.1. Ieejas informācijas apraksts

Sistēmā tiks nodrošināta šādas ieejas informācijas apstrāde.

1. Informācija par **lietotājiem** sastāvēs no šādiem datiem.

- Lietotājvārds – burtu teksts ar izmēru līdz 32 rakstzīmēm.
- E-pasts – burtu un ciparu teksts ar izmēru līdz 64 rakstzīmēm.
- Parole – burtu un ciparu teksts ar izmēru no 8 līdz 64 rakstzīmēm.  
Jāsatur vismaz viens lielais burts un simbols.
- Admin – boolean, izvēle start lietotāju un administratoru

2. Informācija par **automašīnām** sastāvēs no šādiem datiem.

- km/l – float ciparu vērtība kilometriem uz litru.
- Šosejas km/l – float ciparu vērtība kilometriem uz litru.
- Kategorija – burtu teksts ar izmēru līdz 255 rakstzīmēm.
- Pārnesumkārbā – burtu teksts ar izmēru līdz 255 rakstzīmēm.
- Degviela – burtu teksts ar izmēru līdz 255 rakstzīmēm.
- Ražotājs – burtu teksts ar izmēru līdz 255 rakstzīmēm.
- Modelis – burtu teksts ar izmēru līdz 255 rakstzīmēm.
- Gads – cipari ar izmēru līdz 4 rakstzīmēm.
- Ražotājs – burtu teksts ar izmēru līdz 255 rakstzīmēm.
- Nomas faktors – float ciparu vērtība nomas koeficientam.
- Cena – float ciparu vērtība nomas cenai.
- Pieejamība – boolean vērtība modeļa pieejamībai sistēmā.

3. Informācija par **pārdošanas vietu** sastāvēs no šādiem datiem.

- Valsts – burtu teksts ar izmēru līdz 255 rakstzīmēm.
- Pilsēta – burtu un ciparu teksts ar izmēru līdz 255 rakstzīmēm.
- Adrese – burtu un ciparu teksts ar izmēru līdz 255 rakstzīmēm.
- Epasts – burtu un ciparu teksts ar izmēru līdz 255 rakstzīmēm.
- Telefona numurs – cipari ar izmēru līdz 20 rakstzīmēm.

4. Informācija par **īri** sastāvēs no šādiem datiem.

- Sākuma datums – datuma un laika vērtība.
- Beigu datums – datuma un laika vērtība.
- Statuss – boolean vērtība, kas norāda vai noma pašlaik ir procesā.



- Kopējā cena – float vērtība nomas cenas izvadei.

### **2.1.2. Izejas informācijas apraksts**

1. **Automašīnu kataloga skats:** Automašīnu modeļi, pieejamība sistēmā, modeļu cenas.
2. **Rezervācijas skats:** Automašīnu modeļi, pieejamība sistēmā, detalizēta informācija par modeli.
3. **Rezervācijas apkopojuma skats:** Dati ar detalizētu informāciju par rezervēto datumu, laiku, vietu, cenu, personu un automašīnas modeli.
4. **Uzstādījumu skats:** Lietotāja datu izvade un lauki un izmaiņu veikšanai.
5. **Administratīvā paneļa skats:** Izvadīta kopīgā lietotāju uzskaitē, lietotāju rezervācijas datu apkopojumi.

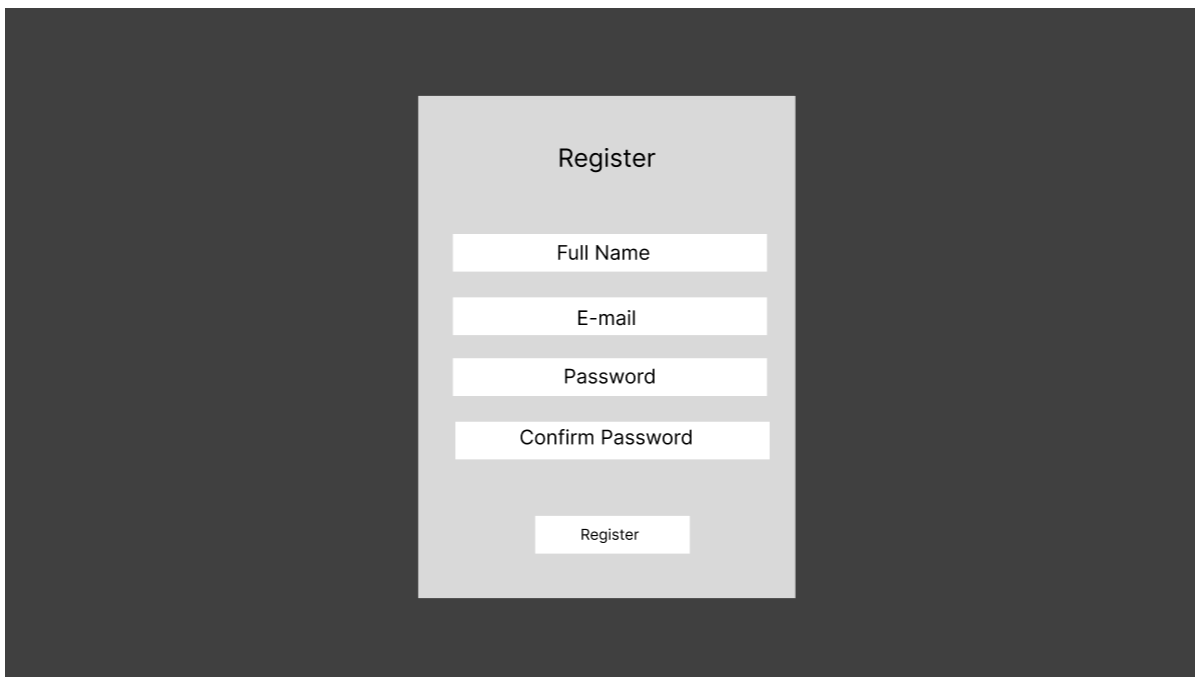
## **2.2 Funkcionālās prasības**

1. Jānodrošina lietotāju reģistrācija.
  - 1.1. Veikt reģistrācijas datu ievadi un pārbaudīt to formatējumu. Izvadīt kļūdu paziņojumus formāta kļūdu gadījumos.
  - 1.2. Attiecīgi informēt par kļūdām gadījumos, kad tiek konstatētas problēmas ar formātu.
  - 1.3. Salīdzināt ievadītos datus ar esošo datu bāzi, lai noskaidrotu, vai lietotājs jau ir reģistrējies.
  - 1.4. Pārbaudīt paroles drošības līmeni. Ja tas neatbilst noteikumiem, sniegt atbilstošu paziņojumu.
2. Jānodrošina lietotāja autorizācija.
  - 2.1. Pārbaudīt ievadītos autorizācijas datus un veikt to formatējuma verifikāciju.
  - 2.2. Izvadīt atbilstošus paziņojumus par kļūdām vai veiksmīgu autorizāciju.
  - 2.3. Pārbaudīt lietotāja statusu un atļaut piekļuvi tikai autorizētiem lietotājiem.
3. Jānodrošina administrācijas paneļa funkcionalitāte.
  - 3.1. Nodrošināt iespēju rediģēt lietotāju piekļuves tiesību. Funkcija administratoriem, lai izveidotu un pārskatītu citus administratorus sistēmā.
  - 3.2. Nodrošināt iespēju administratoriem dzēst rezervācijas, kuras ir nepieciešams atcelt.
  - 3.3. Attēlot visus sistēmas lietotājus un to saistītās rezervācijas.
  - 3.4. Nodrošināt iespēju izmainīt rezervāciju statusa datus.
4. Jānodrošina lietotāja profila pārskata funkcionalitāte paneļa.
  - 4.1. Sniedz iespēju lietotājam rediģēt un atjaunināt personīgo informāciju, piemēram, vārdu, uzvārdu, e-pasta adresi vai citus profilam saistītus datus.
  - 4.2. Piedāvā funkciju saglabāt veiktās profila izmaiņas.

5. Jānodrošina lietotāja profila pārskata funkcionalitāte paneļā.
  - 5.1. Sniedz iespēju lietotājam rediģēt un atjaunināt personīgo informāciju, piemēram, vārdu, e-pasta adresi vai citus profilam saistītus datus.
  - 5.2. Piedāvā funkciju saglabāt veiktās profila izmaiņas.
6. Jānodrošina produktu pārskata funkcionalitāte lietotājiem.
  - 6.1. Nodrošina iespēju lietotājam apskatīt visus pieejamos automobiļus sistēmā.
  - 6.2. Piedāvā informatīvu un viegli pārskatāmu sarakstu ar esošajiem automobiļiem.
  - 6.3. Sniedz detalizētu informāciju par automobiļiem, ieskaitot modeļus, cenas, pieejamību un citas būtiskas īpašības.
7. Jānodrošina rezervācijas izvadīšanas funkcionalitāte lietotājiem.
  - 7.1. Attēlo vēstures pārskatu ar veikto rezervāciju datiem par modeli, vietu un laiku.
  - 7.2. Nodrošina iespēju dzēst rezervācijas, kuras ir nepieciešams atcelt.

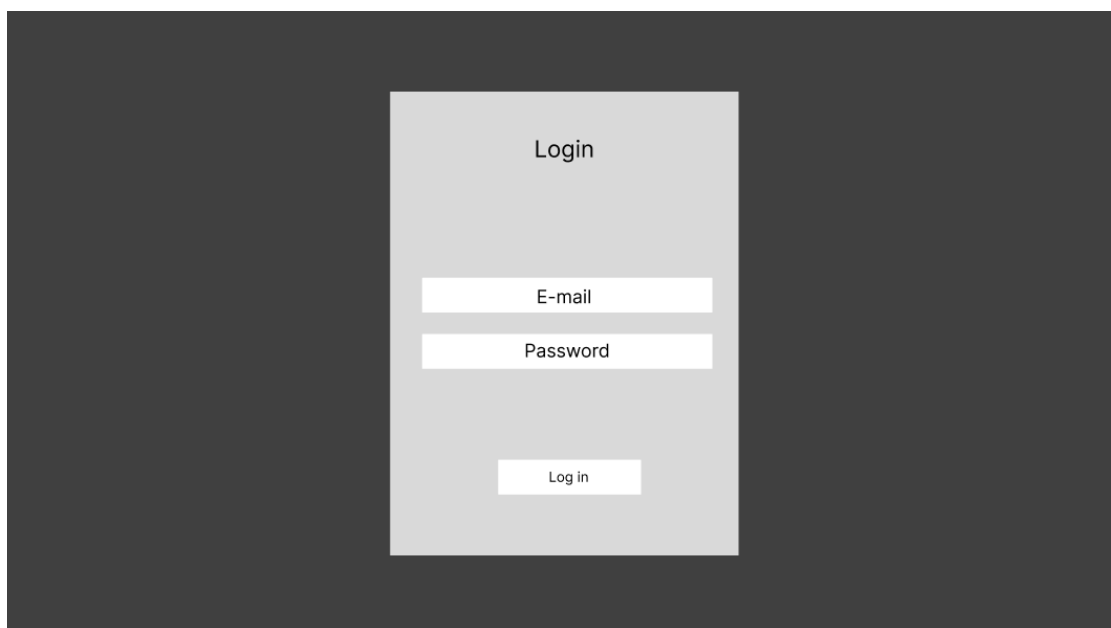
## **2.3 Nefunkcionālās prasības**

1. Sistēmas saskarnes valodai ir jābūt angļu valodā.
2. Jānodrošina tīmekļa lietojumprogrammas pielāgošanas ekrāna izmēriem, kas mūsdienās tiek lietoti, lai to varētu izmantot uz dažādiem monitora izmēriem.
3. Tekstam ir jābūt Helvetica fontā.
4. Jānodrošina vienkāršas sistēmas atjaunināšanas un uzturēšanas iespējas, lai ieviestu jaunas funkcijas vai labotu iespējamās kļūdas.
5. Jānodrošina elastīgas un drošas piekļuves kontroles iespējas, lai ierobežotu piekļuvi noteiktai informācijai un funkcionalitātei atbilstoši lietotāja lomai.

A sketch of a registration form titled "Register". The form is centered on a dark gray background. It contains four input fields stacked vertically, each with a label: "Full Name", "E-mail", "Password", and "Confirm Password". Below these fields is a "Register" button.

2.1. att. Sistēmas reģistrācijas skice

Šī skice demonstrē lapu ērtam un saprotamam interfeisam reģistrācijas procesa nodrošināšanai. Tā piedāvā ievadīt nepieciešamos datus, lai pieteikties sistēmai jaunajiem lietotājiem.

A sketch of a login form titled "Login". The form is centered on a dark gray background. It contains two input fields stacked vertically, each with a label: "E-mail" and "Password". Below these fields is a "Log in" button.

2.2 att. Sistēmas pievienošanās skice

Šī skice demonstrē lapu ērtam un saprotamam interfeisam pievienošanās procesa nodrošināšanai. Tā piedāvā ievadīt nepieciešamos datus, lai pieteikties sistēmai esošajiem lietotājiem.

### 3. UZDEVUMU RISINĀŠANAS LĪDZEKĻU IZVĒLES PAMATOJUMS

#### Frontend:

- Next.js 14 ir React framework, kas paredzēts pilnvērtīgu tīmekļa lietojumprogrammu veidošanai. Izmanto React Components, lai izveidotu lietotāja saskarnes, un Next.js funkcijas un optimizācijas. Next.js automātiski konfigurē React nepieciešamos rīkus, piemēram, komplektēšanu un kompilēšanu.
- Tailwind CSS 3.3.6 izceļas ar savu modularitāti, ātrumu un skaidrajām klasēm. Nodrošina iepriekš definētus stila elementus, atvieglojot izstrādi un nodrošinot atsaucīgu dizainu.

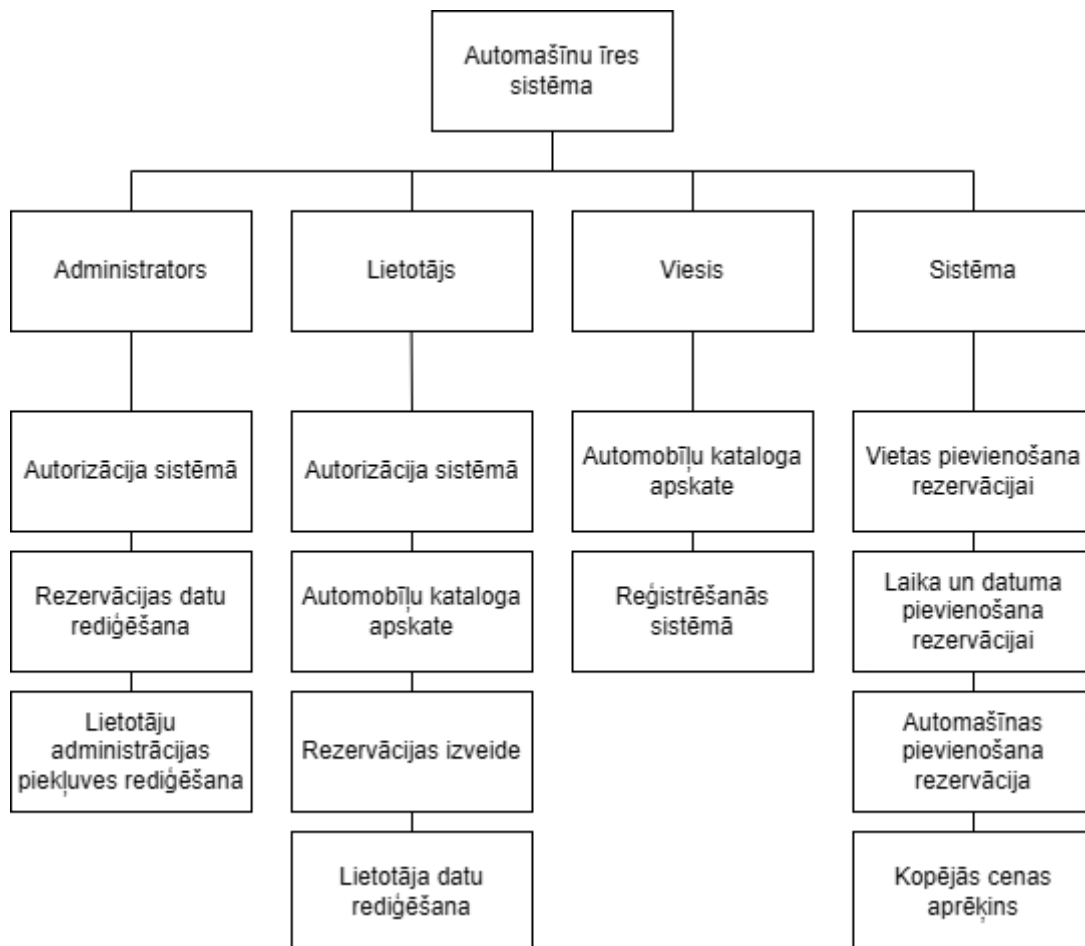
#### Backend:

- Typescript 5.3 piedāvā statisku tipu pārbaudi, kas var palīdzēt novērst kļūdas jau izstrādes laikā. Izmantojot moduļus un importus, veicina kodu atkārtotu pielietošanu un uzturēšanu.
- Prisma 5 ir nākamās paaudzes Node.js un TypeScript ORM. Uzlabo darbu ar datu bāzēm, pateicoties intuitīvajam datu modelim, automātiskajai migrācijai, tipu drošībai un automātiskajai pabeigšanai. Padara datu bāzu darbu vienkāršāku un drošāku, piedāvājot modernus risinājumus un efektīvu izstrādi.
- PostgreSQL 16.1 piedāvā plašu funkciju un paplašinājumu klāstu, kas atbalsta sarežģītus datu pārvaldības uzdevumus. Izmanto uzlabotu SQL valodas implementāciju, kas nodrošina procedurālo valodu, triggeru, pārskatu un citu datu pārvaldības koncepciju atbalstu.

## 4. PROGRAMMATŪRAS PRODUKTA MODELĒŠANA UN PROJEKTĒŠANA

### 4.1 Sistēmas struktūras modelis

#### 4.1.1. Sistēmas arhitektūra



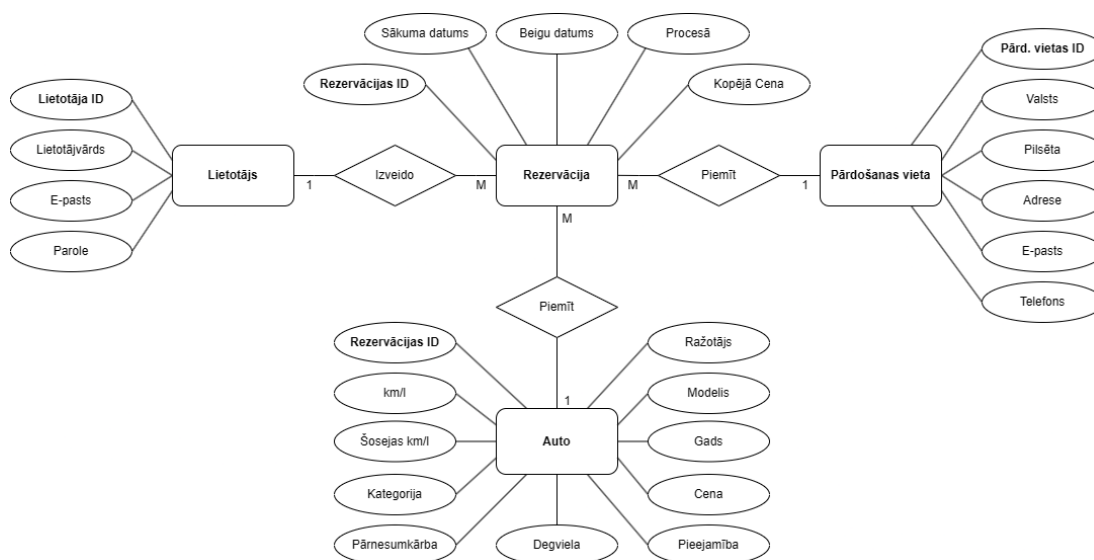
4.1.1.att. Sistēmas arhitektūra

- **Administratoru modulis.** Administratoriem sistēmā ir pieejama datu kontrole sistēmā. Personīgā autorizācija, produkta datu rediģēšana, lietotāju piekļuves pārvaldība un rezervāciju datu rediģēšana. Administrators var autorizēties, lai iegūtu piekļuvi administratīvajam panelim, rediģēt informāciju par lietotājiem un pārdošanas vietām, kā arī pārvaldīt esošās rezervācijas.
- **Lietotāju modulis.** Lietotāji ir sistēmas galvenie lietotāji, kuriem ir pieeja automobiļu katalogam, rezervāciju veidošanai un savu lietotāja datu rediģēšanai. Viņi var pārlūkot pieejamo automobiļu katalogu, pielāgot automobiļu nomas datumu un laiku, izveidot rezervācijas un rediģēt savu personīgo informāciju.

- **Viesu modulis.** Viesis ir nepieautorizēts lietotājs, kuram ir ierobežota pieeja sistēmai. Viņam ir iespēja veikt reģistrāciju, pārlūkot ierobežotu automobiļu katalogu, bet viņš nevar veikt rezervācijas vai rediģēt lietotāja datus.
- **Sistēmas modulis.** Sistēma izpilda tehniskās funkcijas, kas saistītas ar datu apstrādi un iekšējo sistēmas darbību. Pēc rezervācijas izveides no lietotāja puses, sistēma pievieno datus par pārdošanas vietas datiem.

#### 4.1.2. Sistēmas ER modelis

Sistēmas ER-modelis sastāv no 4 entitijām (skat. 2. att.), kas nodrošina pamat informācijas uzglabāšanu un apstrādi. “Rezervācija” raksturo rezervācijas, kas uzglabā lietotāja izvēlēto laiku un datumu, kopējo cenu, kā arī statusu pabeigšanas gadījumos. “Lietotājs” raksturo lietotājvārdu, e-pastu un paroli, kas reģistrējoties tiek saglabāta sistēmā. “Pārdošanas vieta” raksturo tirdzniecības vietas valsti, pilsētu, adresi un telefona numuru. “Auto” raksturo automobiļa km/l, šosejas km/l, kategoriju, pārnēsukārību, degvielu, ražotāju, modeli, gadu, cenu un pieejamību sistēmā.



4.1.2. att. Sistēmas ER-diagramma

ER-modelis demonstrē entītijū savstarpējo saistību:

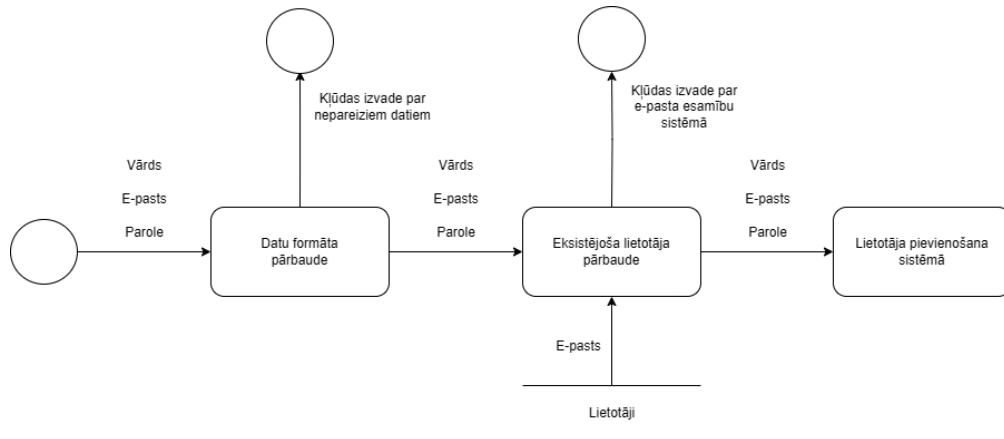
- starp entītijām “Lietotājs” un “Rezervācija” ir attiecība viens pret daudziem, jo viena rezervācija var piederēt tikai vienam lietotājam, bet lietotājs var izveidot vairākas rezervācijas.
- starp entītijām “Pārdošanas Vieta” un “Rezervācija” ir attiecība viens pret daudziem, jo viena rezervācija var piederēt tikai vienai pārdošanas vietai, bet pārdošanas vietas var pievienot vairākām rezervācijām.
- starp entītijām “Auto” un “Rezervācija” ir attiecība viens pret daudzi, jo viens auto var piederēt vairākām rezervācijām.

## 4.2 Funkcionālās sistēmas modelis

### 4.2.1. Datu plūsmu modelis

#### 1. Lietotāja izveidošana.

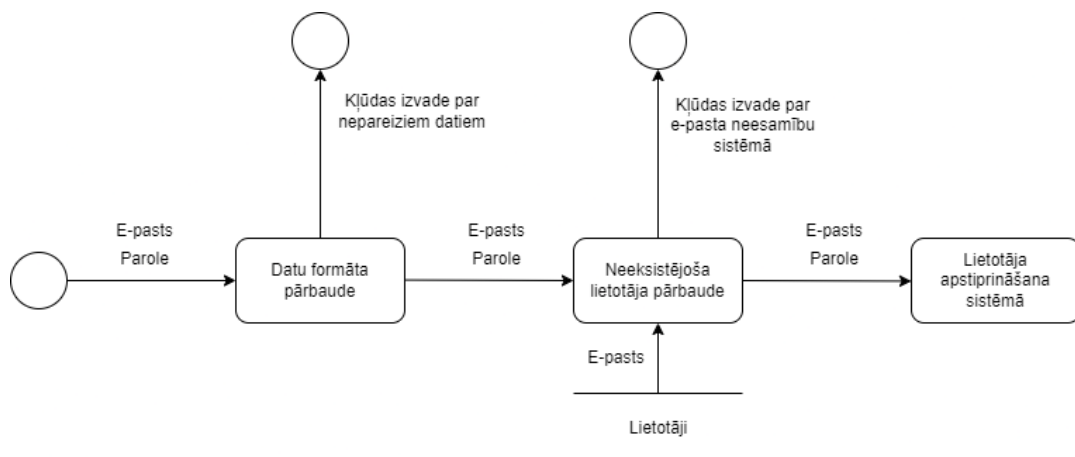
Lietotājiem būs iespēja reģistrēt savu kontu sistēmā. 4.2.1. attēlā ir redzama lietotāja datu ievade, sekojošā formatējuma un drošības nosacījumu pārbaude. Tai seko pārbaude par konta pārbaude par iepriekšējo eksistenci caur e-pasta reģistrāciju datu bāzē. Procesa beigās lietotāja saturs tiek saglabāts, reģistrējot jauno lietotāju sistēmā.



4.2.1. att. Lietotāju izveidošanas datu plūsmas diagramma

#### 2. Lietotāja pievienošanās sistēmā.

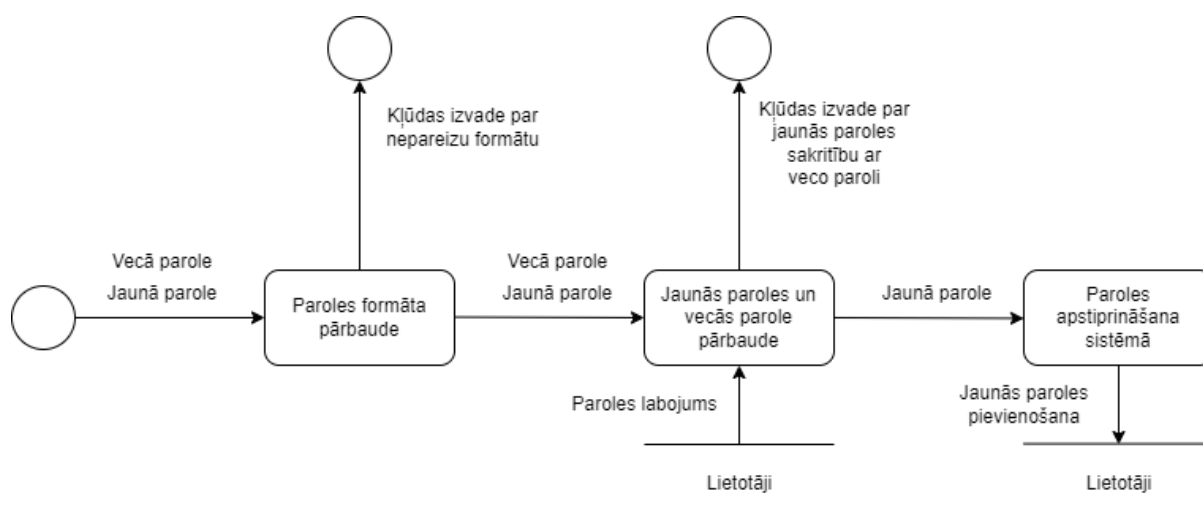
Lietotājiem būs iespēja ar izveidoto kontu pievienoties sistēmā. 4.2.2. attēlā ir redzama lietotāja datu ievade, seko formatējuma pārbaude. Tai seko pārbaude par konta neeksistēšanas gadījumu caur e-pastu datu bāzē. Procesa beigās lietotāja saturs tiek saglabāts, pievienojot lietotāju sistēmā.



4.2.2.att. Lietotāju pievienošanas datu plūsmas diagramma

#### 3. Lietotāja paroles atjaunošana.

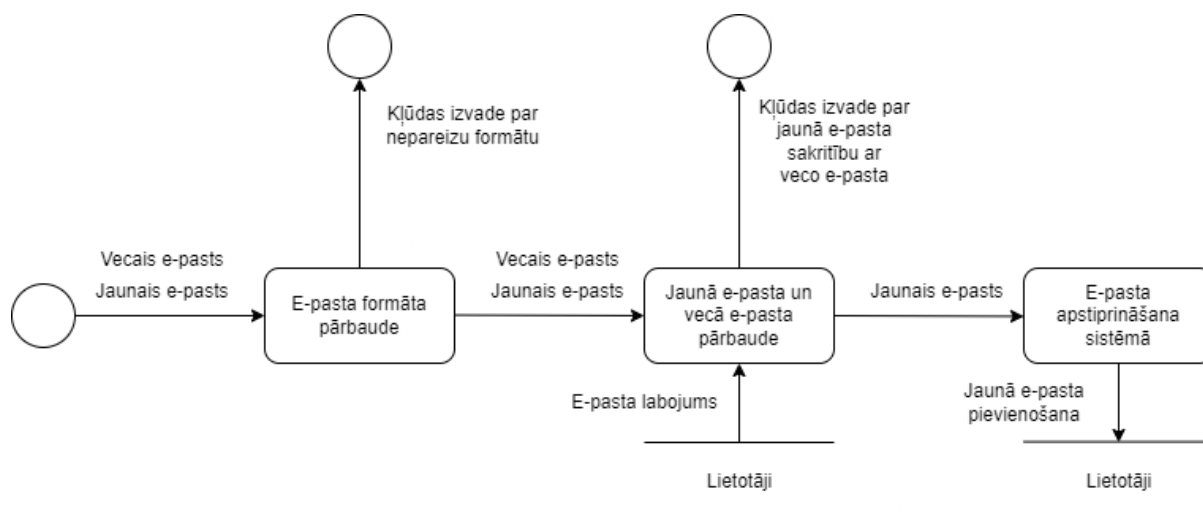
Lietotājiem būs iespēja ar atjaunot savu paroli sistēmā. 4.2.3. attēlā ir redzama lietotāja paroles ievade, seko formatējuma pārbaude. Tai seko pārbaude par paroles sakritību ar iepriekšējo. Procesa beigās lietotāja parole tiek saglabāta sistēmā.



4.2.3.att. Lietotāju paroles atjaunošanas datu plūsmas diagramma

#### 4. Lietotāja e-pasta atjaunošana.

Lietotājiem būs iespēja ar atjaunot savu e-pastu sistēmā. 4.2.4. attēlā ir redzama lietotāja e-pasta ievade, seko formatējuma pārbaude. Tai seko pārbaude par e-pastu sakritību ar iepriekšējo. Procesa beigās lietotāja e-pasts tiek saglabāts sistēmā.

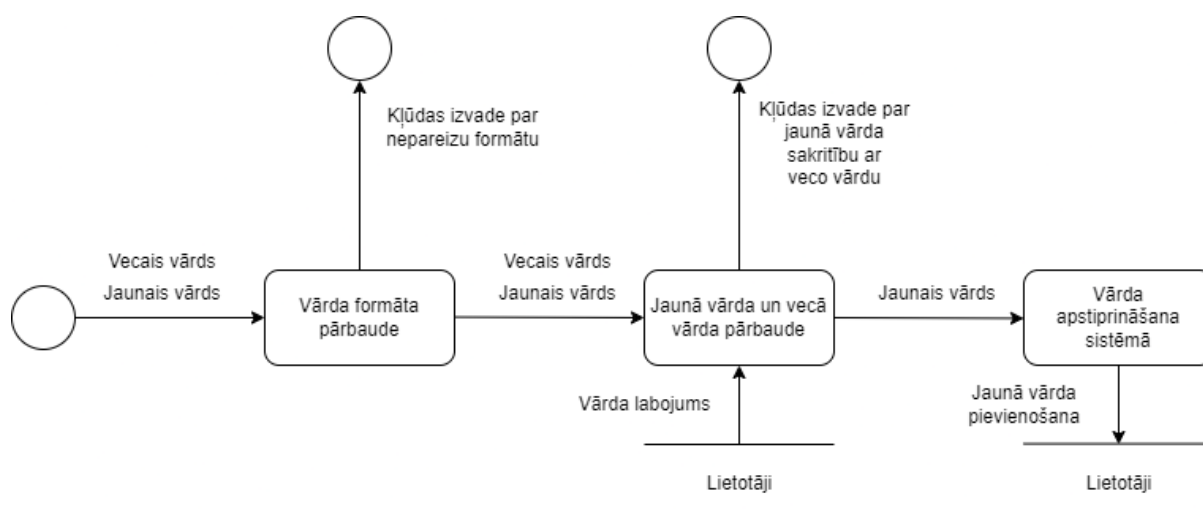


4.2.4.att. Lietotāju e-pasta atjaunošanas datu plūsmas diagramma

#### 5. Lietotāja vārda atjaunošana.

Lietotājiem būs iespēja ar atjaunot savu vārdu sistēmā. 4.2.5. attēlā ir redzama lietotāja vārda ievade, seko formatējuma pārbaude. Tai seko pārbaude par vārda sakritību ar iepriekšējo. Procesa beigās lietotāja vārds tiek saglabāts sistēmā.

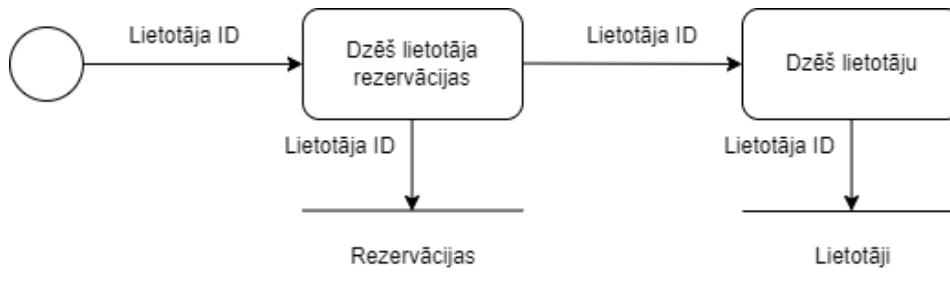




4.2.5.att. Lietotāju vārda atjaunošanas datu plūsmas diagramma

## 6. Lietotāja profila dzēšana, administratīva lietotāja dzēšana.

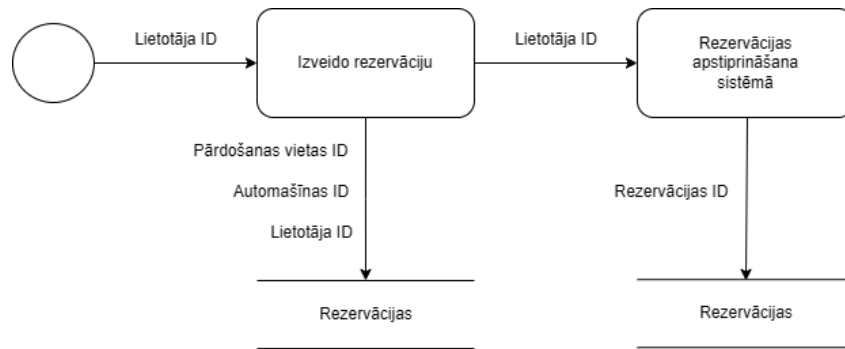
Lietotājiem būs iespēja ar dzēst savu profilu, administratoriem būs iespēja dzēst savu un citu lietotāju profilus sistēmā. 4.2.6. attēlā ir redzama ar lietotājiem saistīto rezervāciju dzēšana, kurai seko lietotāja datu izdzēšana no sistēmas. Procesa beigās lietotāja tiek izņemts no sistēmas.



4.2.6.att. Lietotāju profila dzēšanas datu plūsmas diagramma

## 7. Lietotāja rezervācijas izveidošana.

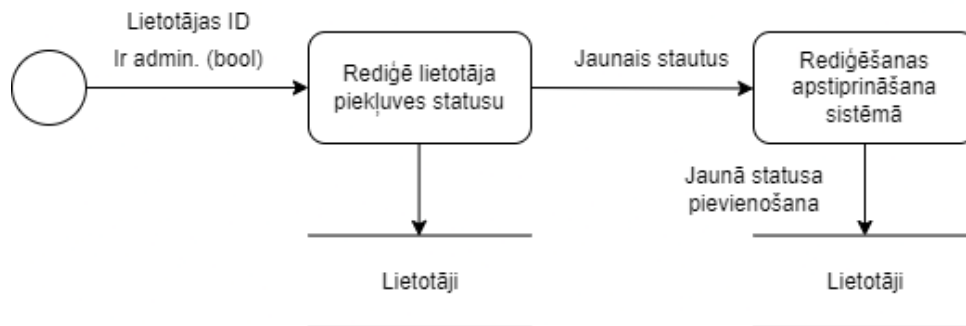
Lietotājiem būs iespēja izveidot rezervācijas sistēmā. 4.2.7. attēlā ir redzama lietotāja rezervācijas izveide un lietotāja, pārdošanas vietas, pārdevēja un automašīnu datu pievienošana. Seko rezervācijas izveide sistēmā ar unikālo identifikatoru. Procesa beigās rezervācija tiek pievienota sistēmai.



4.2.7.att. Lietotāju rezervācijas izveidošanas datu plūsmas diagramma

#### 8. Administratīvā lietotāju piekļuves datu rediģēšana.

Administratoriem būs iespēja rediģēt lietotāju statusu sistēmā. 4.2.8. attēlā ir redzama “Ir admin” statusa rediģēšana, kurai seko jaunā statusa pievienošana, kritēriju padarot patiesu. Procesa beigās jaunie piekļuves statusa dati tiek pievienota sistēmai.



4.2.8. att. Administratīvās lietotāju piekļuves datu rediģēšanas datu plūsmas diagramma

## 5. DATU STRUKTŪRAS APRAKSTS

Datubāze sastāv no 4 tabulām, kas satur informāciju par īri, saņemšanas un nodošanas vietu, automašīnu un lietotāju.

- Tabula “**Rental**” glabā informāciju par īri.
- Tabula “**Dealership**” glabā informāciju par saņemšanas un nodošanas vietu.
- Tabula “**Auto**” glabā informāciju automobīli.
- Tabula “**User**” glabā informāciju par reģistrētajiem lietotājiem.

Tabula “user” ir saistīta ar tabulu “rental”

1. tabula

Tabulas “user” struktūra

Nr.	Nosaukums	Tips	Izmērs	Apraksts
1.	id	varchar	25	Primārā atslēga, unikāls identifikators, pēc noklusējuma cuid()
2.	name	varchar	255	Lietotāja vārds
3.	email	varchar	255	Lietotāja e-pasts, unikāls
4.	password	varchar	255	Lietotāja parole
5.	emailVerified	timestamp	-	Datums un laiks, kad e-pasts tika verificēts, var būt null
6.	image	varchar	255	Lietotāja profila attēls, var būt null
7.	createdAt	timestamp	-	Lietotāja izveides laiks, pēc noklusējuma tagad()
8.	updatedAt	timestamp	-	Datums un laiks, kad lietotājs pēdējo reizi tika atjaunināts, tiek atjaunināts automātiski
9.	admin	boolean	-	Booleans karogs, lai norādītu, vai lietotājs ir administrators, pēc noklusējuma false

Tabula “car” ir saistīta ar tabulu “rental”

2. tabula

Tabulas “car” struktūra

Nr.	Nosaukums	Tips	Izmērs	Apraksts
1.	id	varchar	25	Primārā atslēga, unikāls identifikators, pēc noklusējuma cuid()
2.	kmpl	float	-	Kilometri uz litru (kmpl) automašīnai
3.	highway kmpl	float	-	Kilometri uz litru (kmpl) uz šosejas automašīnai
4.	category	varchar	255	Automobiļa kategorija
5.	transmission	varchar	255	Automobiļa pārnēsma veids
6.	fuel type	varchar	255	Automobiļa degvielas veids
7.	manufacturer	varchar	255	Automobiļa ražotājs
8.	model	varchar	255	Automobiļa modelis
9.	year	int	-	Automobiļa ražošanas gads
10.	slug	varchar	255	URL draudzīgs identifikators automobiļiem
11.	rental factor	float	-	Nomāšanas koeficients cenas aprēķināšanai
12.	rental price	float	-	Pamata nomas cena automašīnai
13.	available	boolean	-	Automobiļa pieejamības statuss, pēc noklusējuma true
14.	createdAt	timestamp	-	Automobiļa izveides laiks, pēc noklusējuma tagad()
15.	updatedAt	timestamp	-	Datums un laiks, kad automašīna pēdējo reizi tika atjaunināta, tiek atjaunināts automātiski

Tabula “dealership” ir saistīta ar tabulu “rental”.

3. tabula

Tabulas “dealership” struktūra

Nr.	Nosaukums	Tips	Izmērs	Apraksts
1.	id	varchar	25	Primārā atslēga, unikāls identifikators, pēc noklusējuma cuid()
2.	country	varchar	255	Valsts, kurā atrodas dīleris
3.	city	varchar	255	Pilsēta, kurā atrodas dīleris
4.	address	varchar	255	Dīlera adrese
5.	email	varchar	255	Dīlera e-pasts
6.	phone	varchar	20	Dīlera tālruņa numurs

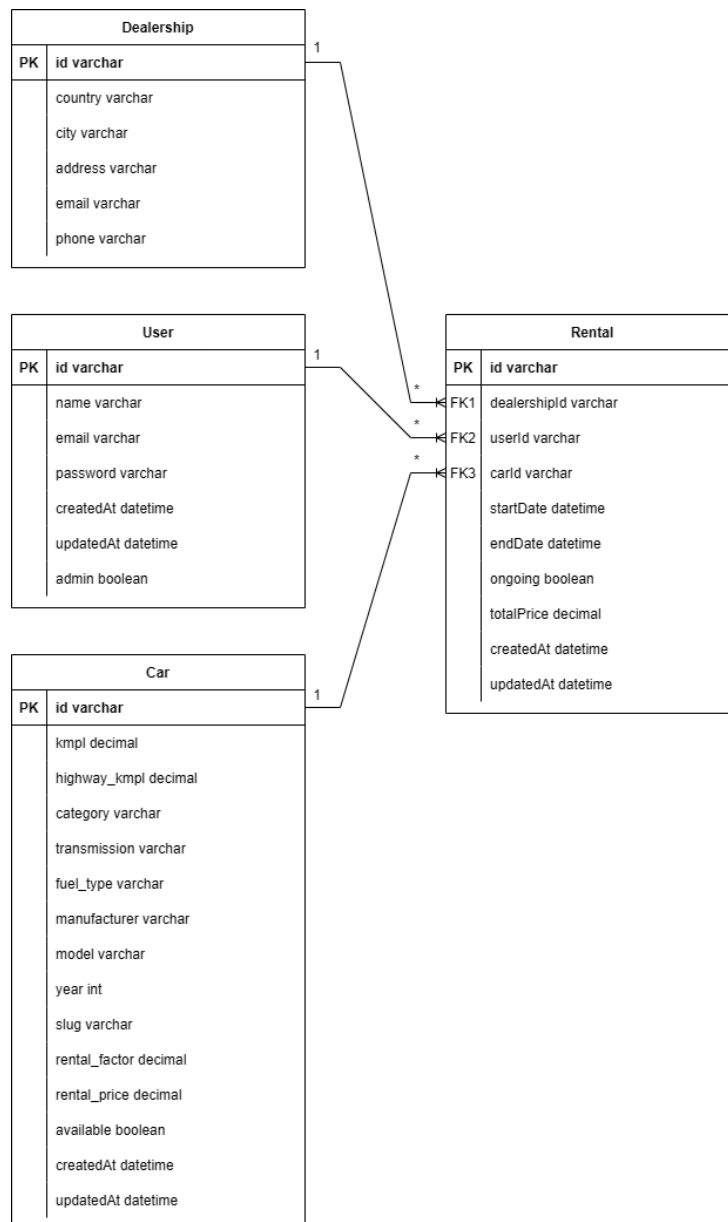
Tabula “rental” ir saistīta ar tabulām “user”, “car”, “dealership”.

4. tabula

Tabulas “rental” struktūra

Nr.	Nosaukums	Tips	Izmērs	Apraksts
1.	id	varchar	25	Primārā atslēga, unikāls identifikators, pēc noklusējuma cuid()
2.	userId	float	25	Ārējā atslēga, norāda uz User(id)
3.	carId	float	25	Ārējā atslēga, norāda uz Car(id)
4.	dealershipId	varchar	-	Ārējā atslēga, norāda uz Dealership(id)
5.	startDate	varchar	-	Nomāšanas sākuma datums
6.	endDate	varchar	-	Nomāšanas beigu datums
7.	ongoing	boolean	-	Nomāšanas statuss, vai tas notiek
8.	totalPrice	float	-	Kopējā nomas cena
9.	createdAt	timestamp	-	Nomāšanas izveides laiks, pēc noklusējuma tagad()
10.	updatedAt	timestamp	-	Datums un laiks, kad nomāšana pēdējo reizi tika atjaunināta, tiek atjaunināts automātiski

## 5.1 Tabulu relāciju shēma



5.1.1 att. Tabulu relāciju shēma

## 6. LIETOTĀJA CEĻVEDIS

### 6.1 Sistēmas prasības aparatūrai un programmatūrai

Sistēmas darbības nodrošināšanai ir nepieciešama tīmekļa pārlūkprogramma un interneta savienojums. Sistēma tika izstrādāta Windows 11 operētājsistēmā, izmantojot Brave tīmekļa pārlūkprogrammu, taču tiek nodrošināts atbalsts arī citām tīmekļa pārlūkprogrammām, tostarp:

- Google Chrome;
- Microsoft Edge;
- Mozilla Firefox;
- Opera;
- Safari.

Atbalsts tiek nodrošināts operētājsistēmām, kas atbalsta iepriekšminētās tīmekļa pārlūkprogrammas, tostarp

- Windows 11;
- MacOS;
- Linux;

Veiksmīgai darbībai ir nepieciešams 1 gigabais (GB) brīvas sistēmas atmiņas.

### 6.2 Sistēmas instalācija un palaišana

Sistēmu var palaist, sekojot šīm instrukcijām. Jāinstalē PostgreSQL un Visual Studio Code. Pēc PostgreSQL datu bāzes izveidošanas un palaišanas ir jāveic projekta konfigurācija.

Vispirms ir nepieciešams aizpildīt .env faila datus VisualStudio Code.

- DATABASE\_URL=postgres://YourUserName:YourPassword@YourHostname:5432/YourDatabaseName
- NEXTAUTH\_URL=http://localhost:3000
- NEXTAUTH\_SECRET=string

Pēc .env faila aizpildīšanas, ir jāatver terminālis projekta mapē un jāizpilda komanda `npm install`, lai instalētu nepieciešamos npm moduļus. Šī komanda nodrošinās, ka visas projekta atkarības tiek lejupielādētas un instalētas lokāli.

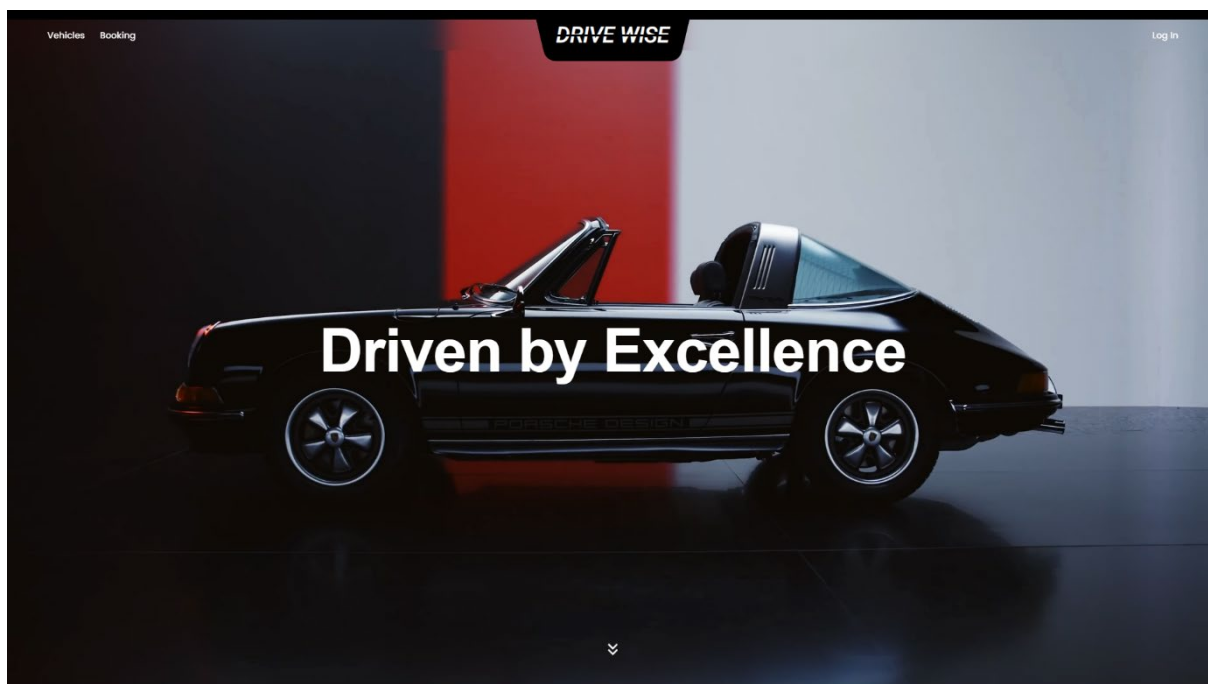
Kad npm moduļi ir instalēti, nākamais solis ir inicializēt datubāzi. To var izdarīt, izpildot komandu `"npx prisma migrate dev --name init"`. Šī komanda veiks datubāzes migrāciju un izveidos nepieciešamās tabulas, pamatojoties uz definēto datubāzes shēmu.

Pēc datubāzes inicializācijas un aizpildīšanas, lai palaistu programmu lokālajā vidē, ir jāizpilda komanda “npm run dev”. Šī komanda sāks izstrādes serveri, un sistēma būs pieejama tīmekļa pārlūkprogrammā, ievadot adresi “http://localhost:3000”.

Kad visi šie soļi ir izpildīti, sistēma būs gatava lietošanai un pieejama tīmekļa pārlūkprogrammā.

## 6.3 Programmas apraksts

### 6.3.1. Sākuma lapa



6.1. att. Sākuma lapas skats

Pēc sistēmas palaišanas tīmeklī tiek uzrādīta sākuma lapa, pilnas funkcionalitātes iespējām ir nepieciešama autorizācija. Navigācijas josla pieļauj apskatīt pieejamo automobiļu katalogu un autentifikāciju. Neautorizētiem lietotājiem rezervācijas lapa ved uz autentifikāciju.

### 6.3.2. Autentifikācija

Neautorizētiem lietotājiem autentifikācijas lapa ļauj ieiet sistēmā ar esošiem konta datiem. Lapa satur laukus e-pasta adresei un parolei, pēc kuru ievades tiek izvadīti paziņojumi par potenciālajām datu kļūmēm. Lietotājs gūst piekļuvi sistēmā pēc pogas “Sign in” nospiešanas. Jaunu kontu ir iespējams reģistrēt spiežot pogu “Register”.

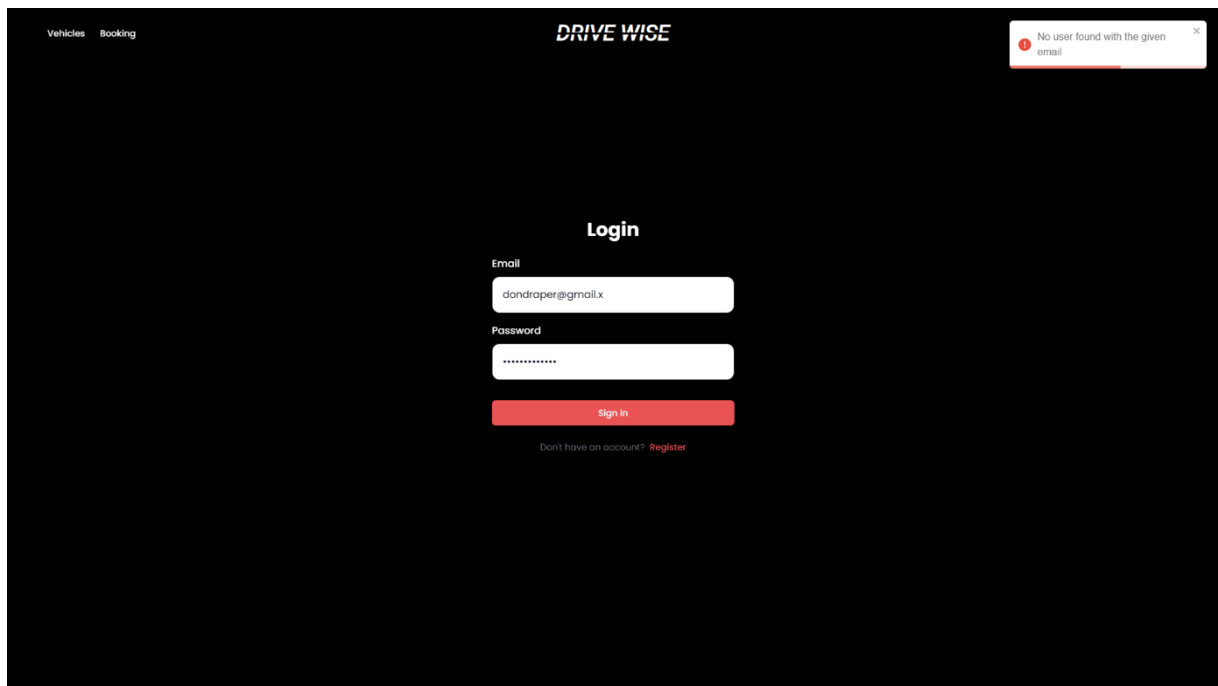
The screenshot shows the 'DRIVE WISE' login page. At the top left are links for 'Vehicles' and 'Booking'. At the top right is a 'Log In' link. The main heading is 'Login'. Below it are two input fields: 'Email' with the placeholder 'mail@example.com' and 'Password' with the placeholder 'your password'. A red 'Sign in' button is positioned below the password field. At the bottom, there is a link that says 'Don't have an account? Register'.

6.2. att. Autentifikācijas skats

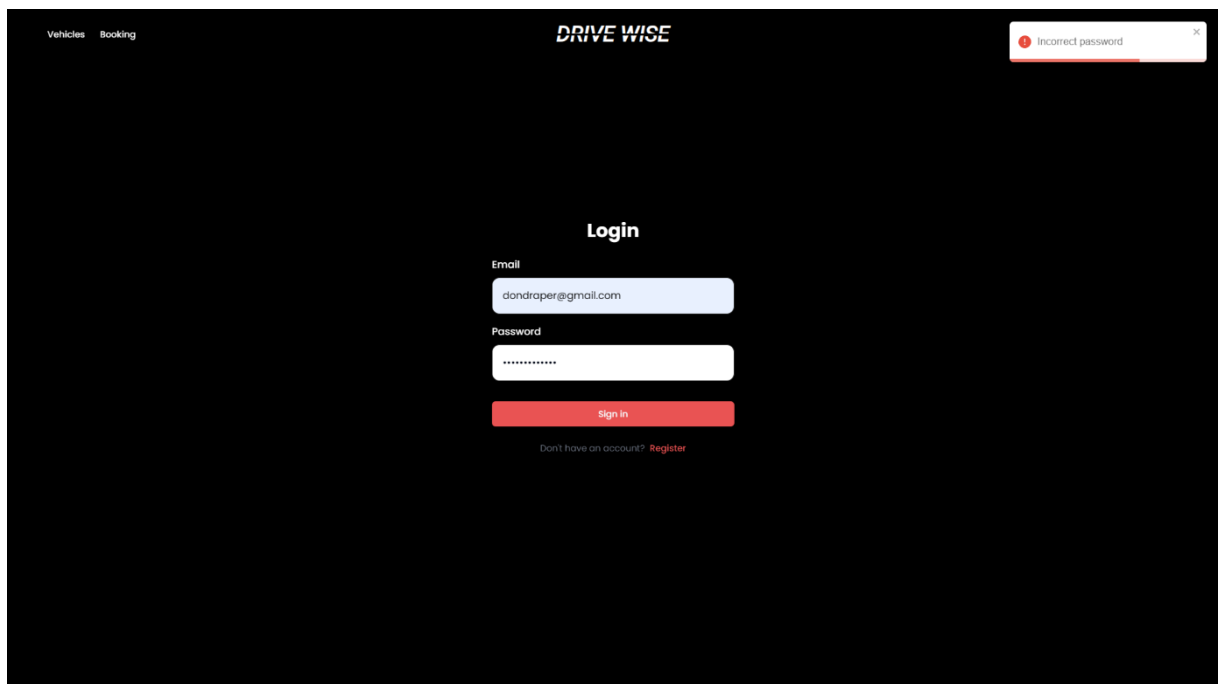
This screenshot shows the same 'DRIVE WISE' login page, but with an error message displayed. The 'Email' input field now contains 'dondrapergmailcom'. A red error message box appears below the email field, stating: 'Please include an '@' in the email address. 'dondrapergmailcom' is missing an '@'. The 'Password' field is masked with dots. The 'Sign in' button and the 'Don't have an account? Register' link remain visible at the bottom.

6.3. att. Autentifikācijas lapas, e-pasta formatējuma kļūdas izvades skats



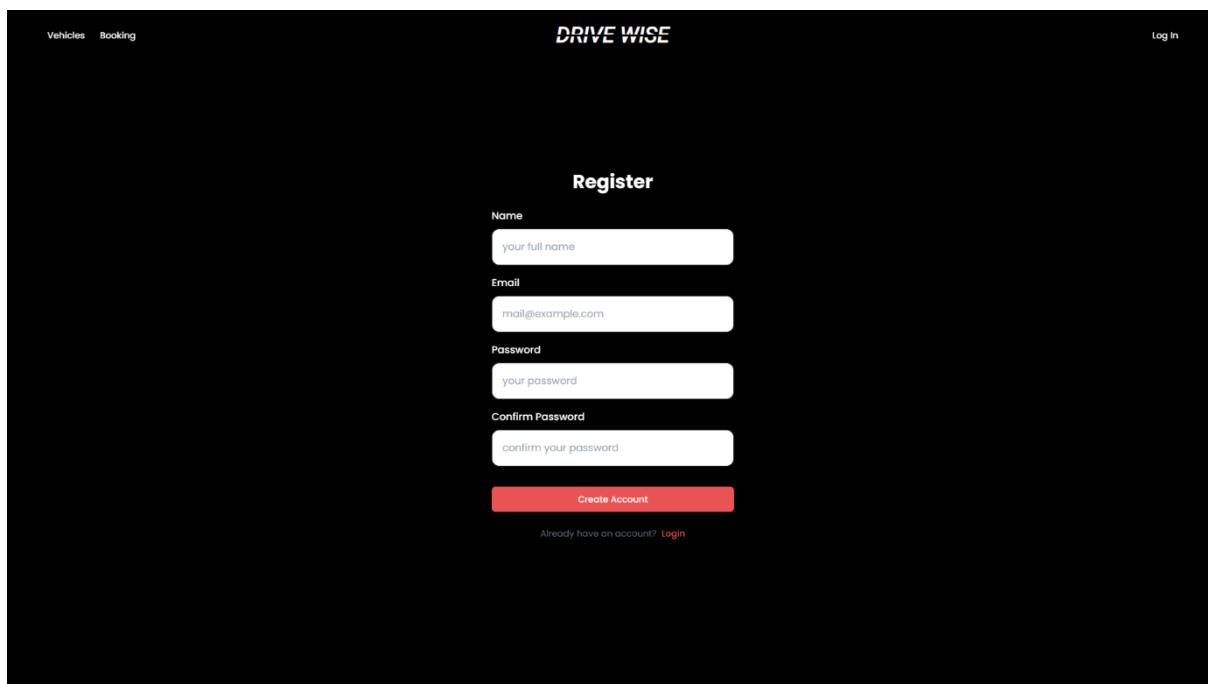


6.4. att. Autentifikācijas lapas, lietotāja e-pasta kļūdas izvades skats

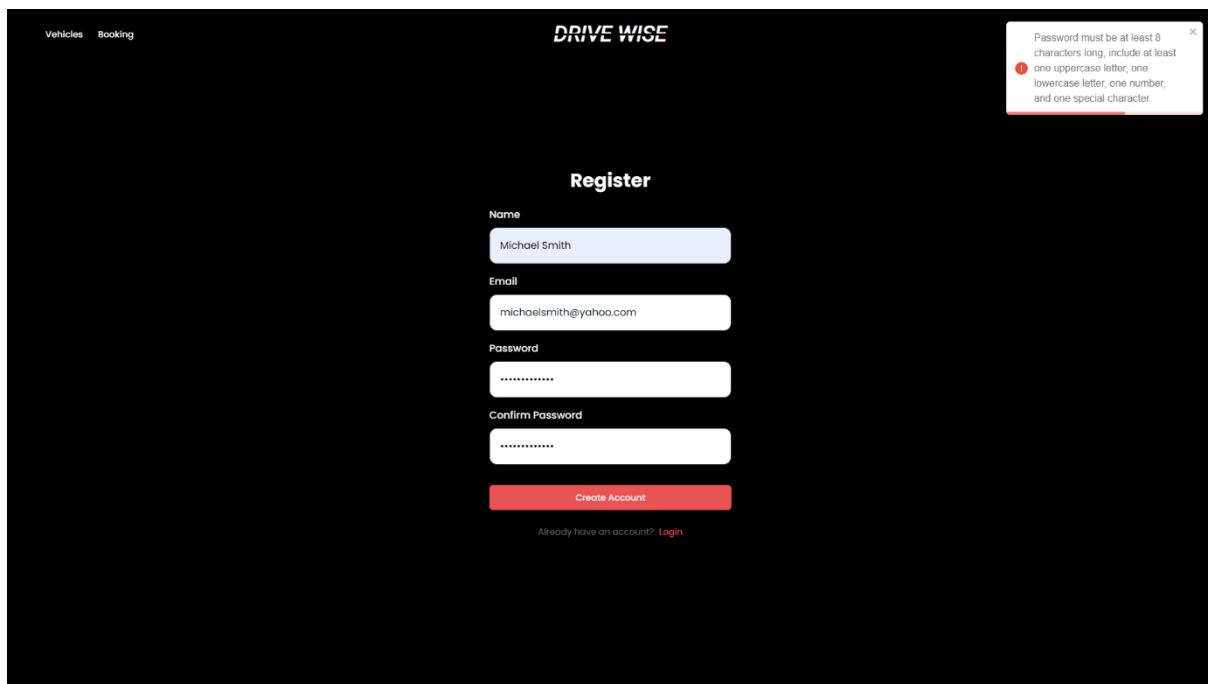


6.5. att. Autentifikācijas lapas, lietotāja paroles kļūdas izvades skats

Reģistrācijas lapā ir ievadlauki vārdam, e-pasta adresei, parolei un paroles apstiprināšanai. Kļūdu gadījumā lietotājs tiek informēts par formatējuma prasībām. Lietotājs tiek reģistrēts pēc “Create Account” pogas nospiešanas.



6.6. att. Lietotāja reģistrācijas skats



6.7. att. Lietotāja reģistrācijas skats, paroles prasību kļūdas izvades skats

Vehicles Booking

DRIVE WISE

Passwords do not match.

### Register

Name

Michael Smith

Email

michaelsmith@yahoo.com

Password

\*\*\*\*\*

Confirm Password

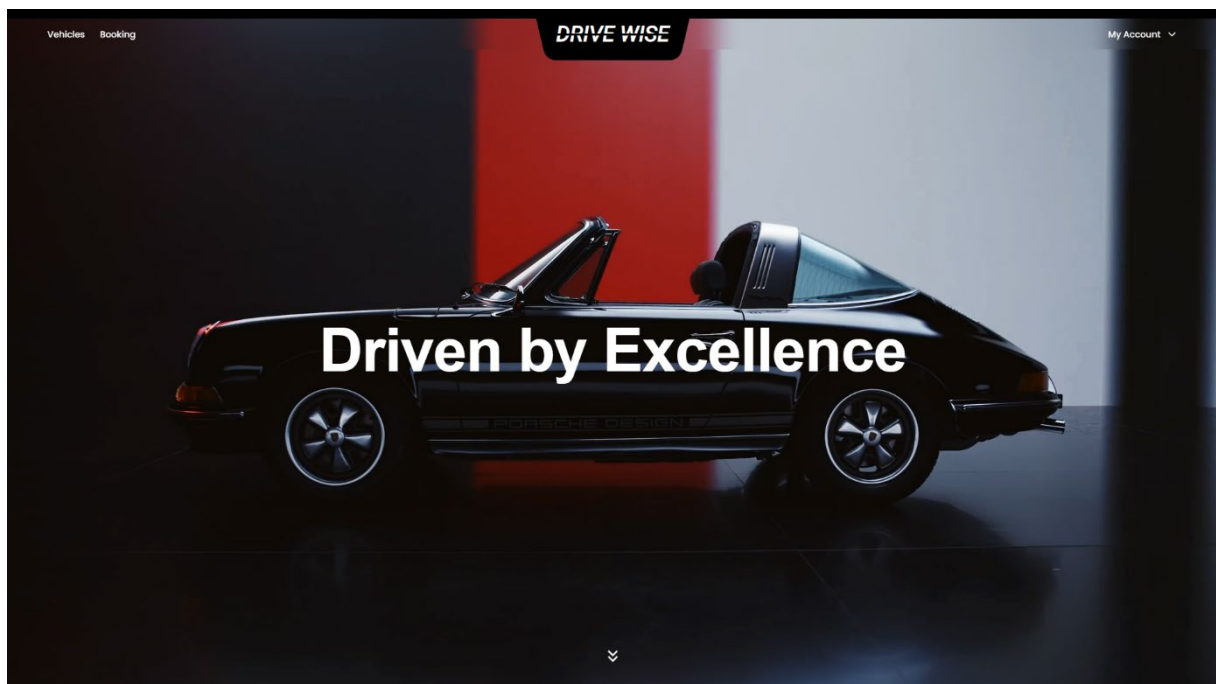
\*\*\*\*\*

Create Account

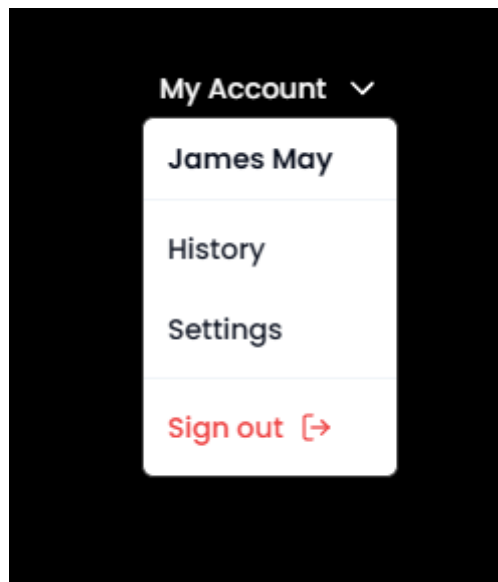
Already have an account? [Login](#)

6.8. att. Lietotāja reģistrācijas skats, paroli nesakritības kļūdas izvades skats

Pēc konta izveidošanas tiek atvērta sākuma lapa ar papildinātām konta navigācijas iespējām, pieeju rezervāciju izveidei.



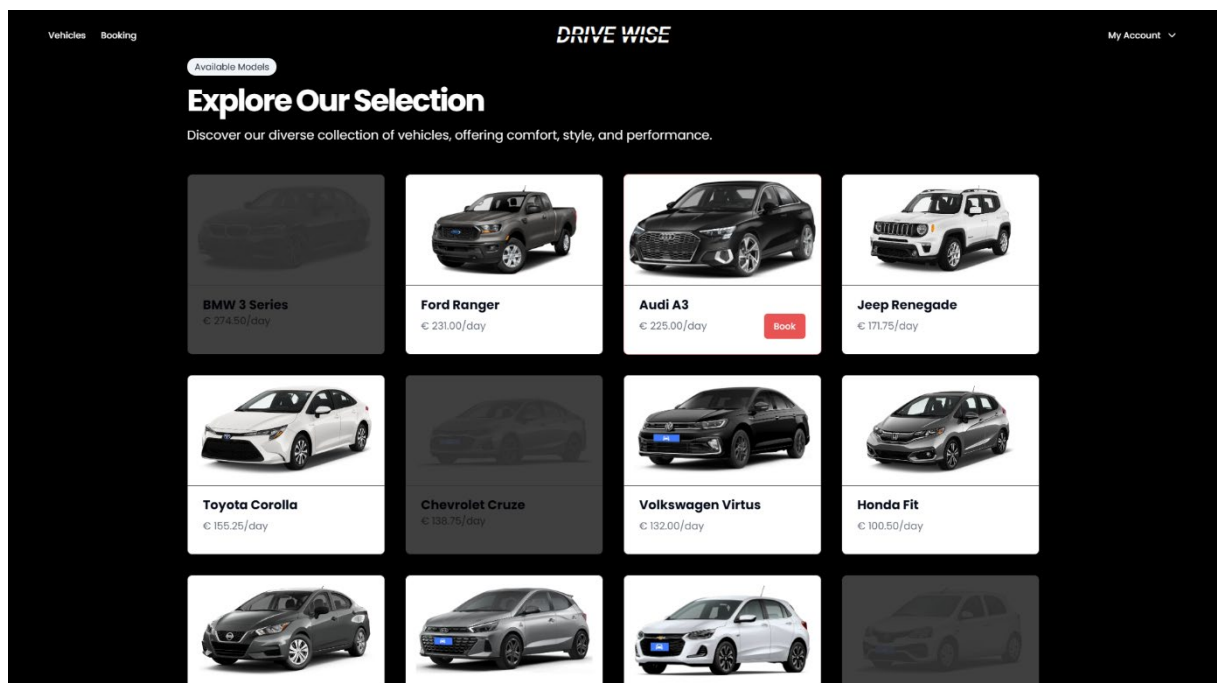
6.9. att. Sākuma lapas skats pēc lietotāja autorizācijas sistēmā



6.10. att. Lietotāja konta navigācijas skats

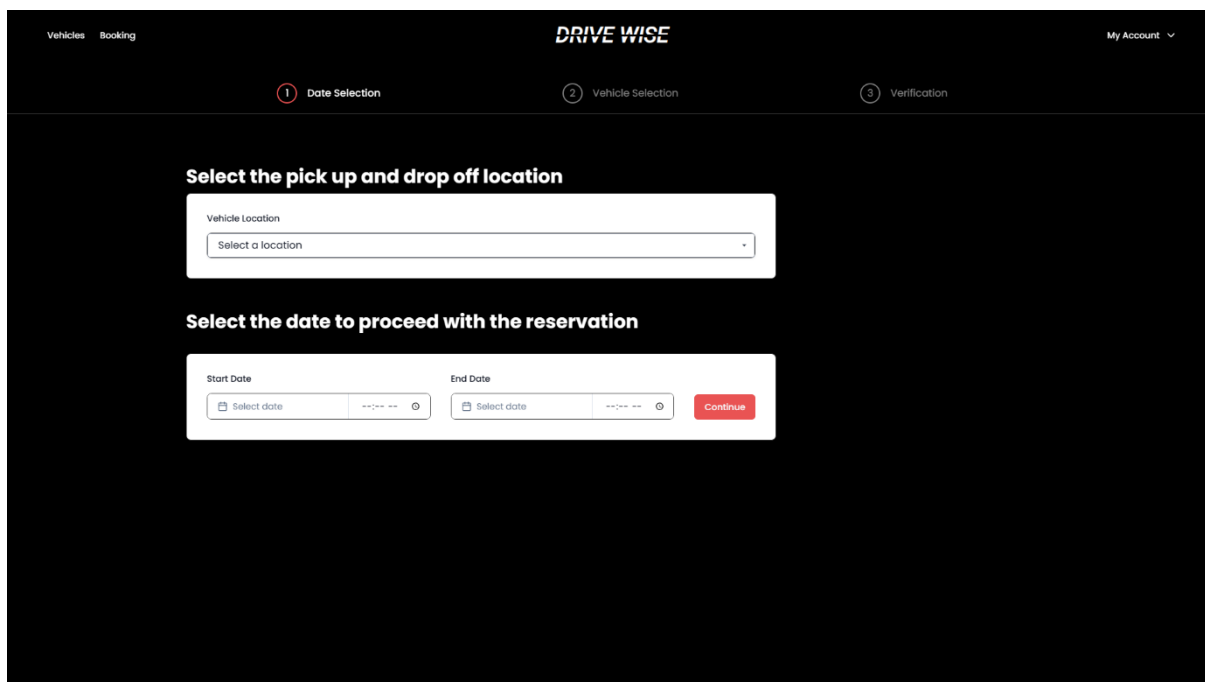
### 6.3.3. Rezervācijas izveides process

Lietotājam nospiežot pogu “Book” kādā no automašīnām katalogā vai “Booking” navigācijas panelī tiek sākts rezervācijas process. Ja lietotājs izvēlās sākt procesu caur “Booking”, tad automašīna izvēle tiek veikta otrajā rezervācijas solī.

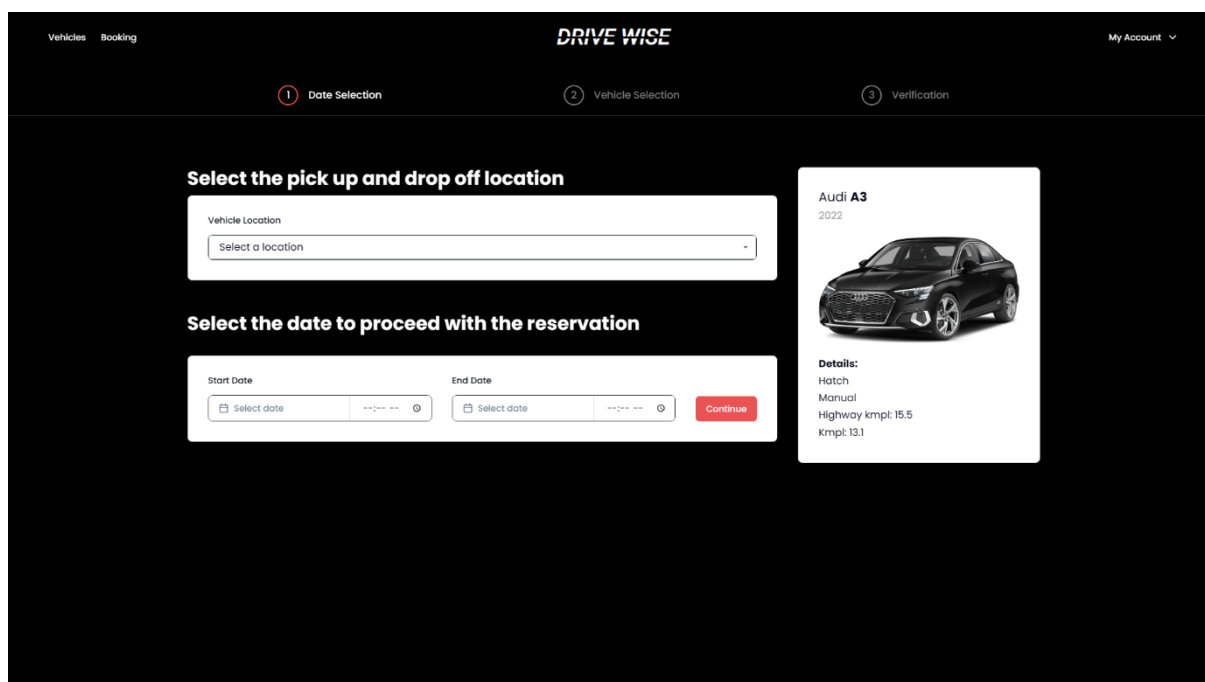


6.11. att. Automobiļu kataloga skats

Pirmajā rezervācijas solī ir ievadlauki vietai, laikam un datumam. Lietotājs atzīmē automašīna savākšanas un nodošanas vietu, laiku un datumu. Poga “Continue” lietotāju ved uz otro rezervācijas soli.

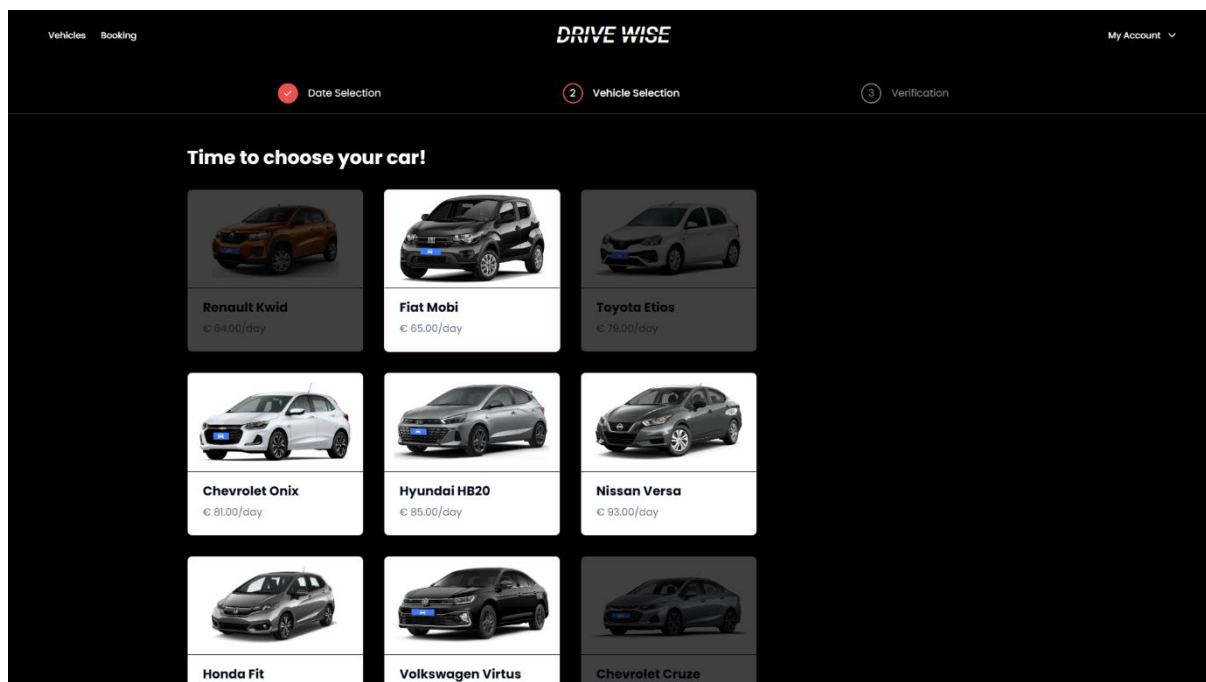


6.12. att. Pirmā rezervācijas soļa skats bez kataloga automašīnas izvēles

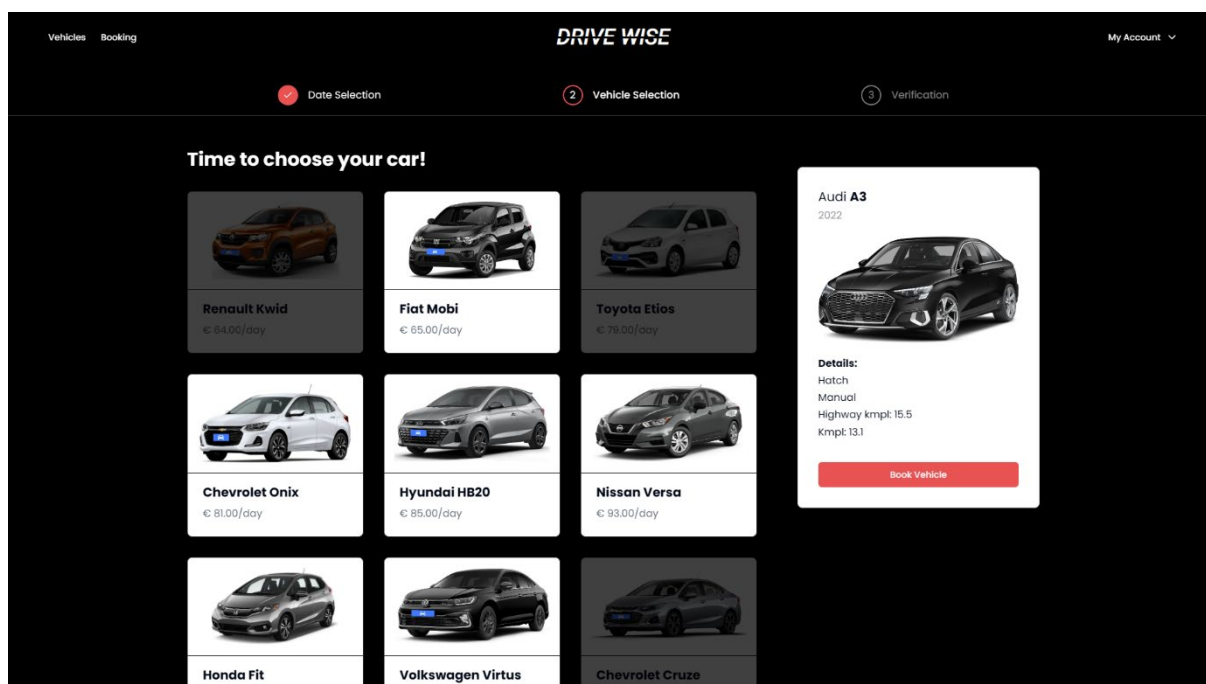


6.13. att. Pirmā rezervācijas soļa skats ar kataloga automašīnas izvēli

Otrajā rezervācijas solī ir iespēja apstiprināt automašīnas izvēli, veikt izmaiņas un salīdzināt datus. Ja lietotājs nav izvēlējis automobili katalogā, tad šeit to ir iespējams izdarīt. Poga “Book Vehicle” lietotāju ved uz trešo rezervācijas soli.

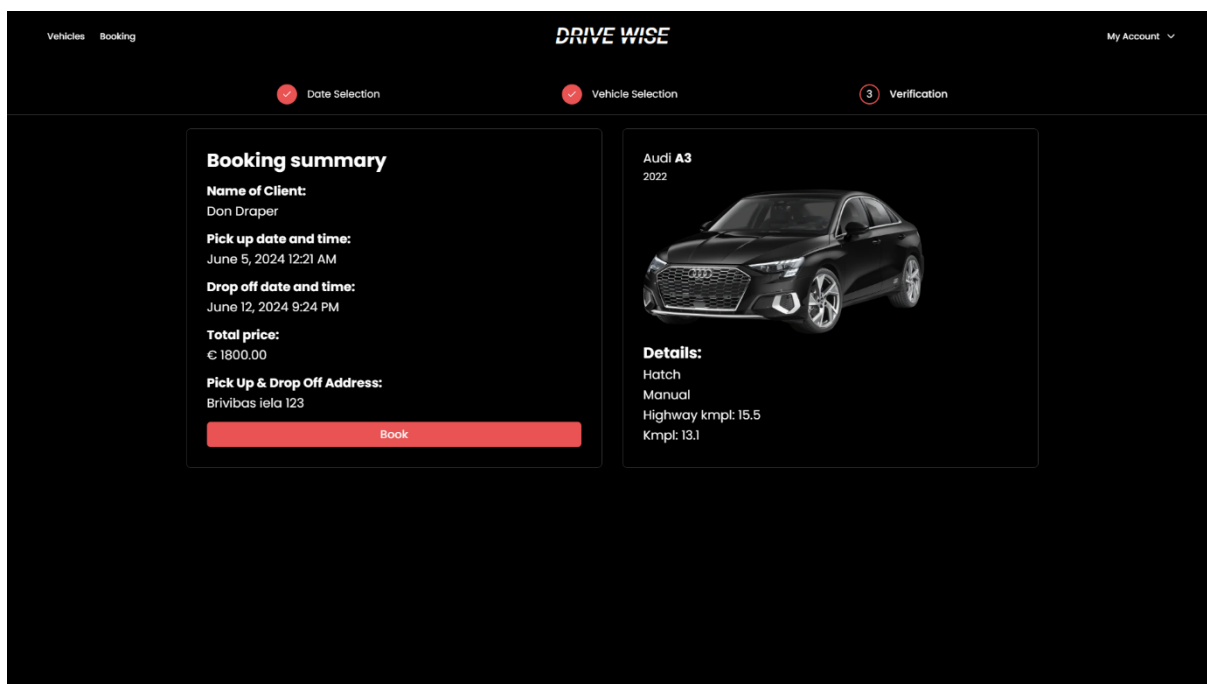


6.14. att. Otrā rezervācijas soļa skats bez kataloga automašīnas izvēli



6.15. att. Otrā rezervācijas soļa skats ar kataloga automašīnas izvēli

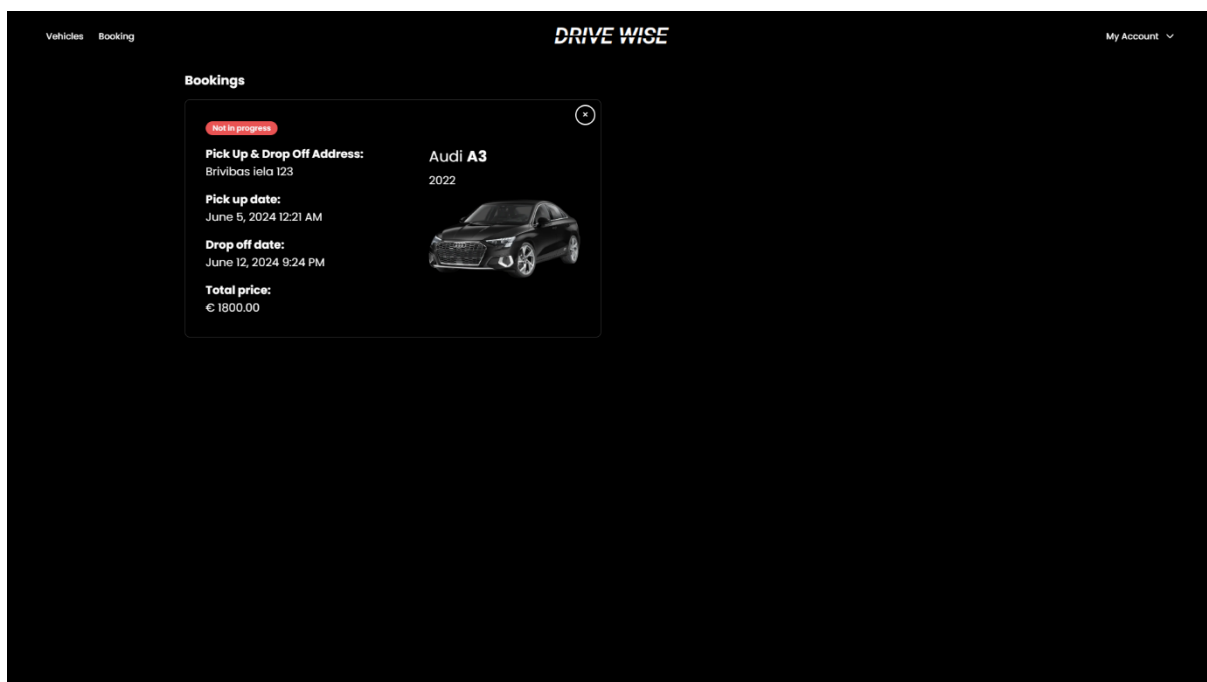
Trešajā rezervācijas solī lietotājs saņem apkopotos datus par rezervāciju. Poga “Book” apstiprina un reģistrē rezervāciju, lietotājs tiek aizvests uz rezervācijas vēstures pārskatu.



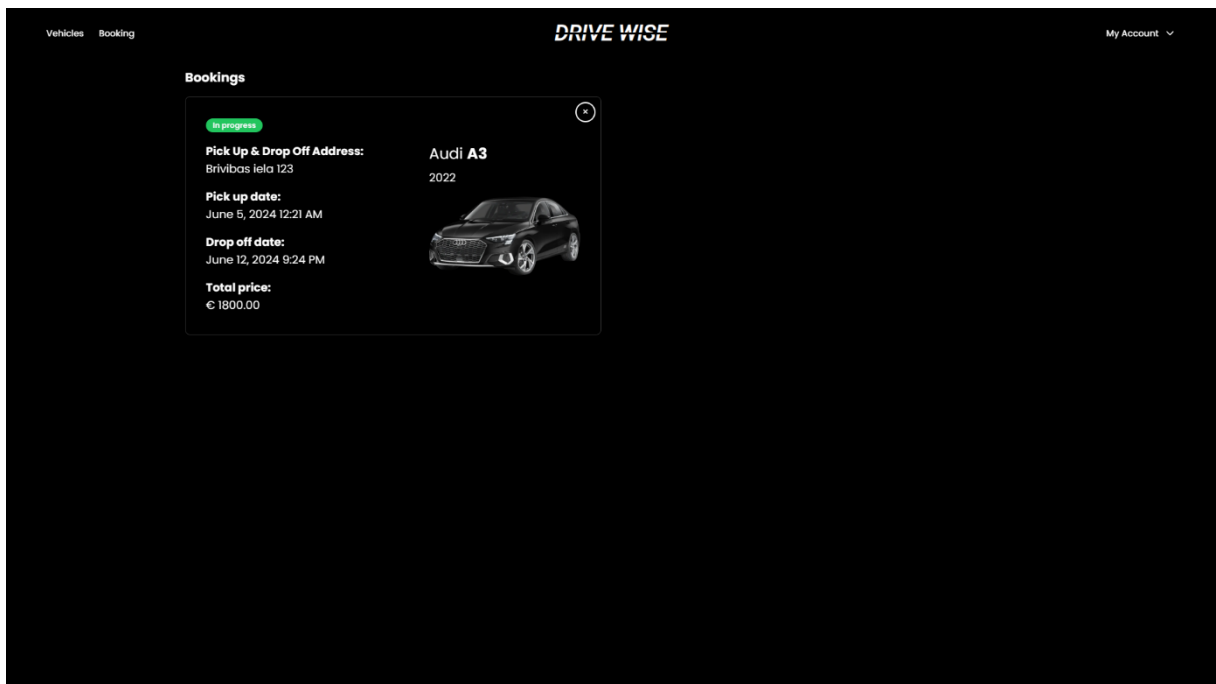
6.16. att. Trešā rezervācijas soļa skats

#### 6.3.4. Lietotāju rezervāciju pārskats

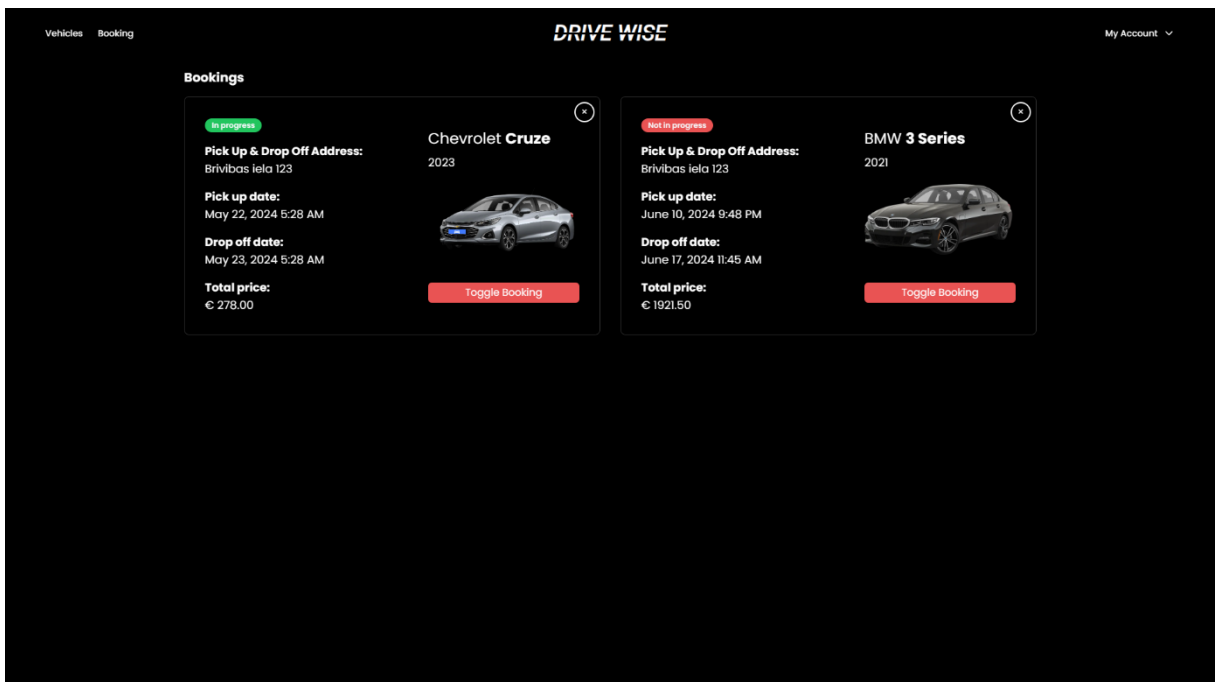
Vēstures pārskatā lietotājam ir iespēja redzēt savas rezervācijas statusu, kuru administrators izmaina caur kontroles paneli. No vēstures lapas administrators var izmainīt arī savas rezervācijas statusu neejot uz kontroles paneli. Ja statuss norāde, ka rezervācija nav procesā, tad rezervāciju var atcelt nospiežot krustiņa ikonu.



6.17. att. Vēstures pārskata skats ar nemainītu statusu



6.18. att. Vēstures pārskata skats ar mainītu statusu

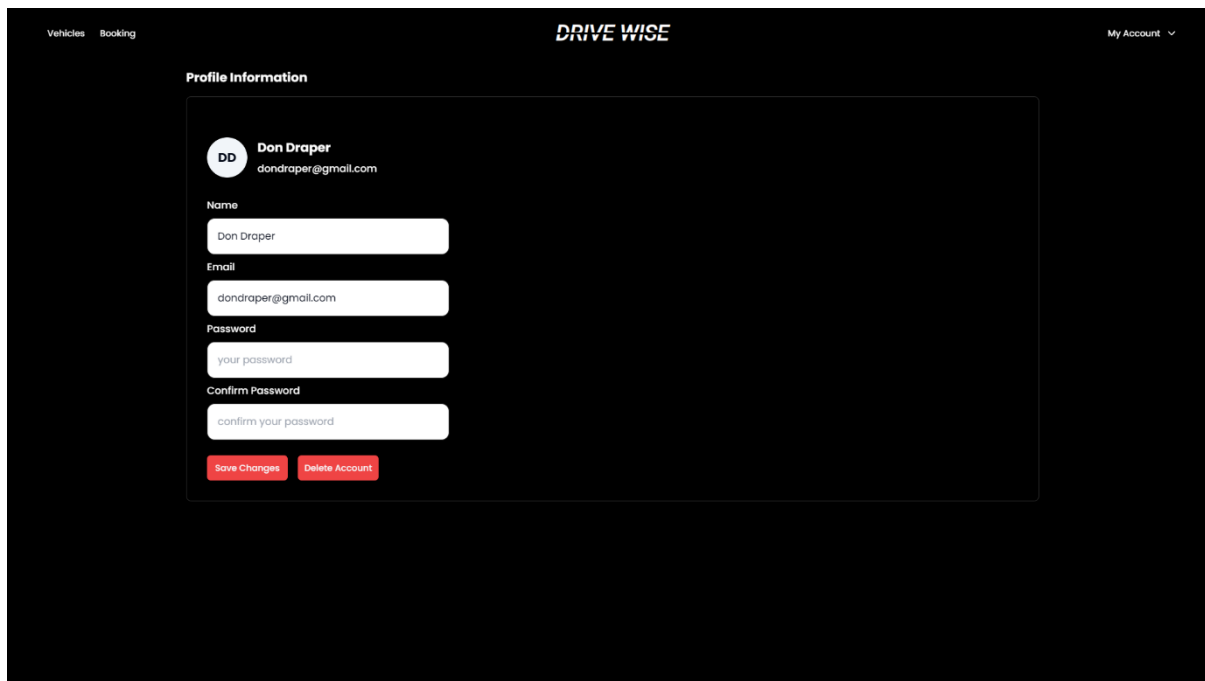


6.19. att. Vēstures pārskata skats ar administratīvajām privilēģijām



### 6.3.5 Lietotāju uzstādījumi

Lietotājam ir iespējams izmainīt sava profila vārdu, e-pastu un paroli. Lapā ir ievadlauki vārdam, e-pasta adresei, parolei un paroles apstiprināšanai. Kļūdu gadījumā lietotājs tiek informēts par formatējuma prasībām. Dati tiek saglabāti pēc “Save Changes” pogas nospiešanas. Poga “Delete Account” ļauj lietotājiem savu kontu izdzēst.

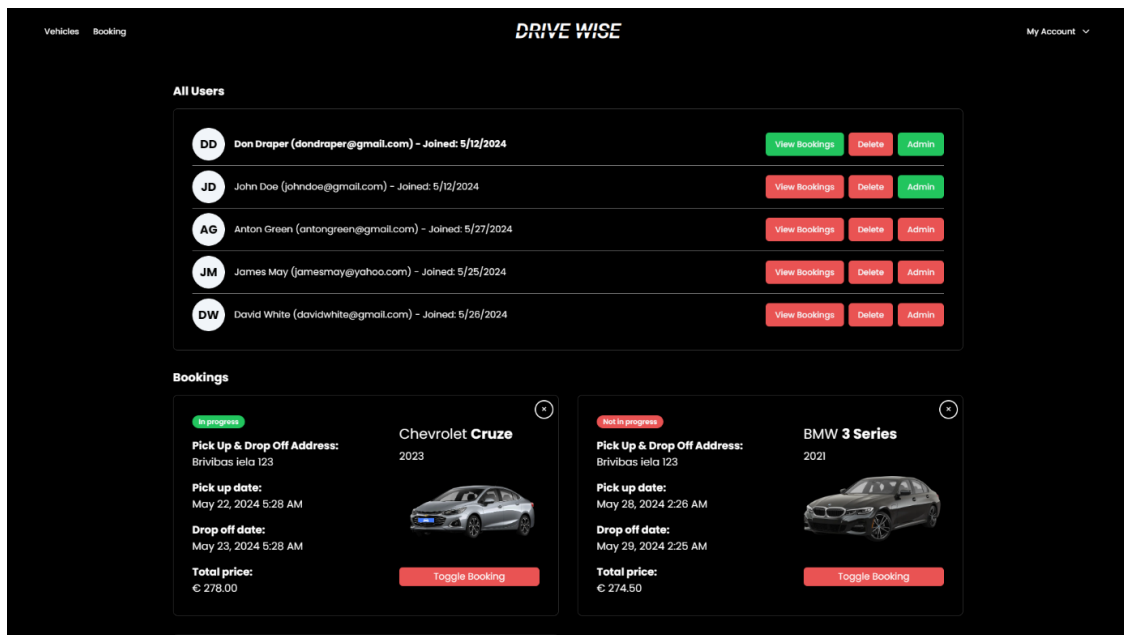


The screenshot shows a web interface for 'DRIVE WISE'. At the top, there are navigation links 'Vehicles' and 'Booking', and a 'My Account' dropdown menu. The main content area is titled 'Profile Information'. It features a user profile card for 'Don Draper' with the email 'dondraper@gmail.com'. Below the card, there are four input fields: 'Name' (containing 'Don Draper'), 'Email' (containing 'dondraper@gmail.com'), 'Password' (containing 'your password'), and 'Confirm Password' (containing 'confirm your password'). At the bottom of the form are two buttons: 'Save Changes' and 'Delete Account'.

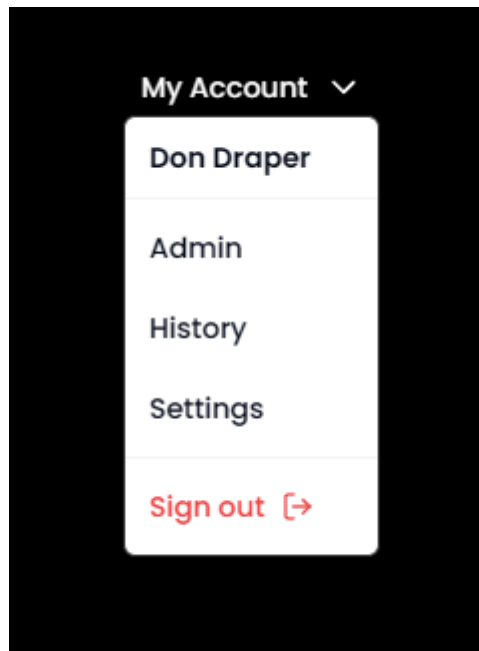
6.20. att. Uzstādījumu skats

### 6.3.6. Administrācijas panelis

Administrācijas panelī administratoriem ir iespējas skatīt visu lietotāju rezervācijas un pārvaldīt to statusus ar pogu “Toggle Booking”. Ja statuss norāde, ka rezervācija nav procesā, tad rezervāciju var atcelt nospiežot krustiņa ikonu. Izmantojot pogu “Delete” ir iespējams lietotāju izdzēst. Poga “Admin” ļauj lietotājiem ieslēgt piekļuvi panelim caur konta navigāciju.



6.21. att. Administrācijas paneļa skats

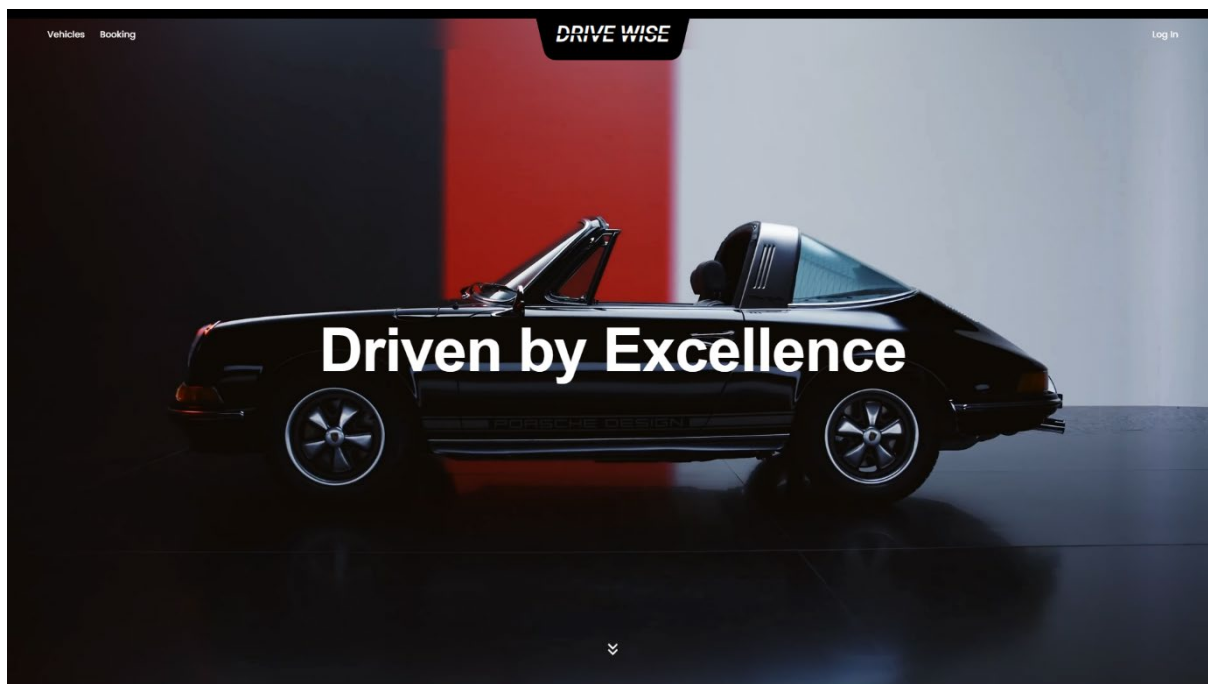


6.22. att. Administratora konta navigācijas skats

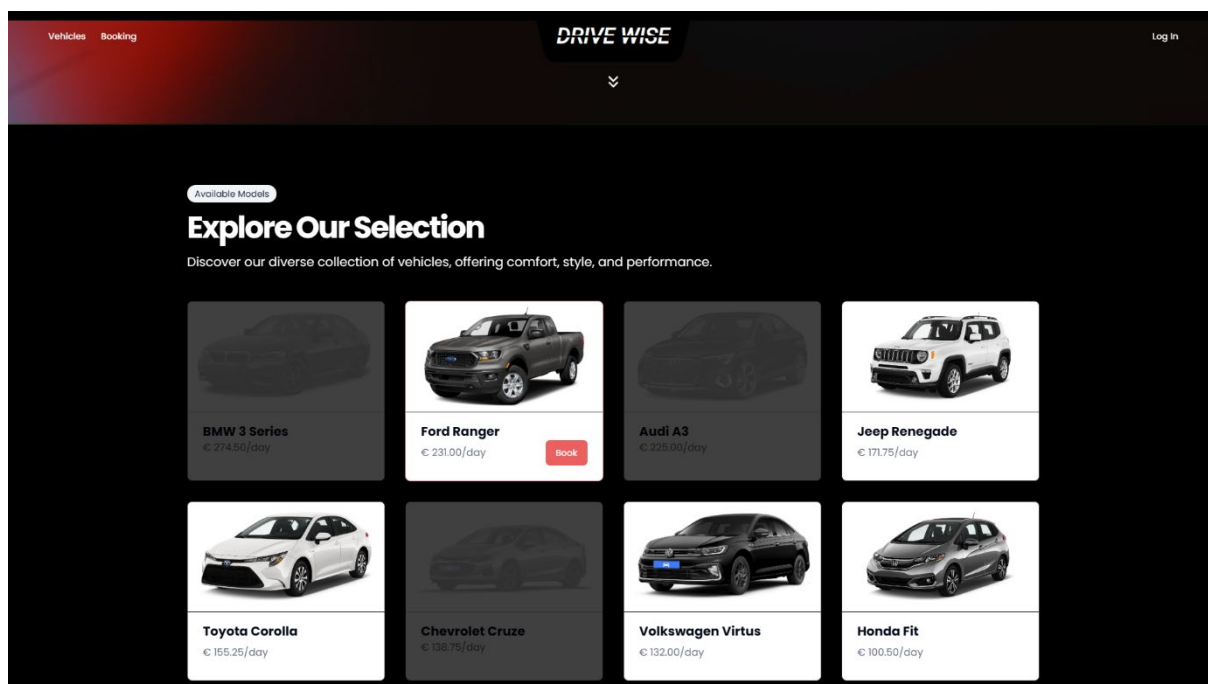
## 6.4 Testa piemērs

Testa piemērā tiks demonstrēts jauna lietotāja izveides process un automobiļa rezervācijas izveidošana.

Atverot sistēmu tīmeklī tiek atvērta sākumlapa, kurā atrodams automobiļu katalogs. Sākumā tiks demonstrēta pārbaude viesu lietotājam, mēģinot izveidot rezervāciju. Lai katalogu atvērtu varam nospiegt pogu “Vehicles” navigācijas joslā, vai tinot sākuma lapu uz leju.



6.23. att. Sākuma lapas skats



6.24. att. Automašīnu kataloga skats

Spiežot pogu ‘Book’ notiks mēģinājums rezervācijas izveidei. Lietotājs varēs iepazīties ar sākotnējo rezervācijas izveides procesu un salīdzināt automobiļu datus. Process tiks apturēts otrajā solī ar pogas “Book Vehicle” pogas nospiešanu, kad lietotājs tiek novirzīts uz ieiešanas lapu.

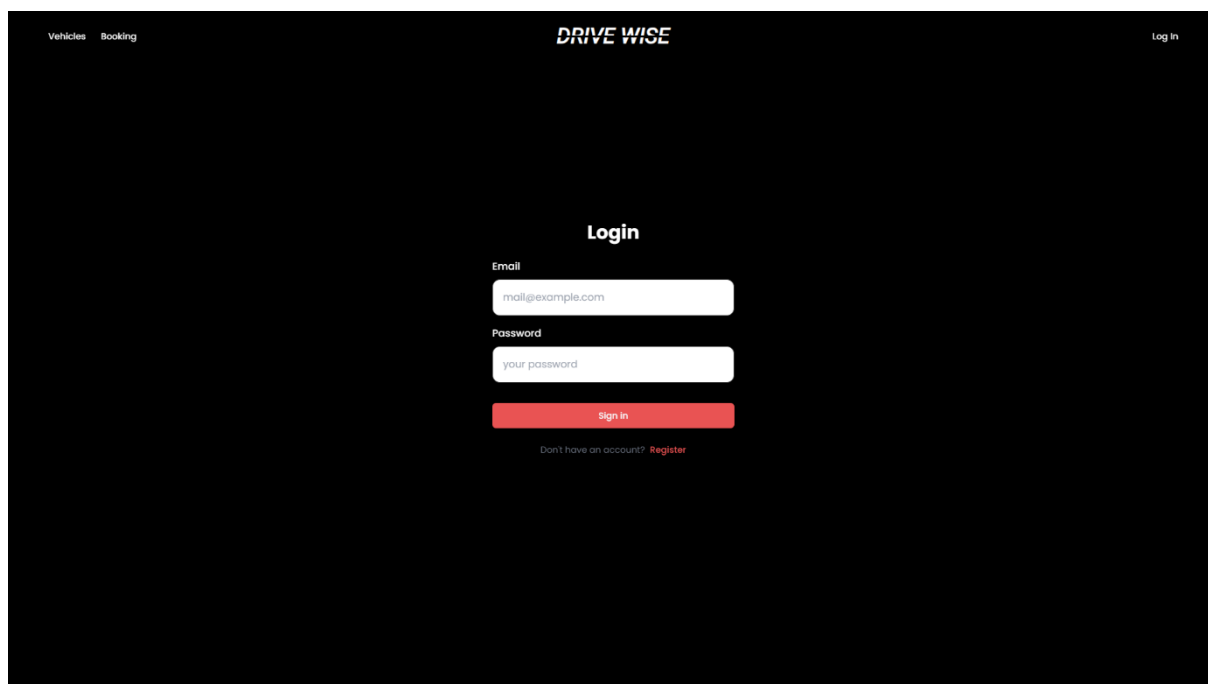
The screenshot shows the 'DRIVE WISE' website interface. At the top, there are links for 'Vehicles' and 'Booking', and a 'Log In' button. Below the navigation bar, there are three steps: 1. Date Selection (highlighted with a red circle), 2. Vehicle Selection, and 3. Verification. The main content area is titled 'Select the pick up and drop off location'. It features a 'Vehicle Location' dropdown menu with 'Latvia - Riga, Brīvības iela 123' selected. Below this, there is a section titled 'Select the date to proceed with the reservation'. It includes 'Start Date' and 'End Date' fields, both set to '06/26/2024'. The 'Start Date' field also shows '01:43 AM' and the 'End Date' field shows '01:44 AM'. A red 'Continue' button is located to the right of the date fields. On the right side of the page, there is a card for a 'Ford Ranger 2023' with a pickup truck image and details: 'Pickup', 'Automatic', 'Highway kmpt: 10.4', and 'Kmpt: 8.7'.

6.25. att. Rezervācijas pirmā soļa skats

The screenshot shows the 'DRIVE WISE' website interface. At the top, there are links for 'Vehicles' and 'Booking', and a 'Log In' button. Below the navigation bar, there are three steps: 1. Date Selection, 2. Vehicle Selection (highlighted with a red circle), and 3. Verification. The main content area is titled 'Time to choose your car!'. It displays a grid of car options. Each option includes a car image, the car model, and the daily rental price. The cars shown are: Renault Kwid (€ 64.00/day), Fiat Mobi (€ 65.00/day), Toyota Etios (€ 76.00/day), Chevrolet Onix (€ 81.00/day), Hyundai HB20 (€ 85.00/day), Nissan Versa (€ 93.00/day), Honda Fit, Volkswagen Virtus, and Chevrolet Cruze. On the right side of the page, there is a card for a 'Ford Ranger 2023' with a pickup truck image and details: 'Pickup', 'Automatic', 'Highway kmpt: 10.4', and 'Kmpt: 8.7'. A red 'Book Vehicle' button is located at the bottom of this card.

6.26. att. Rezervācijas otrā soļa skats

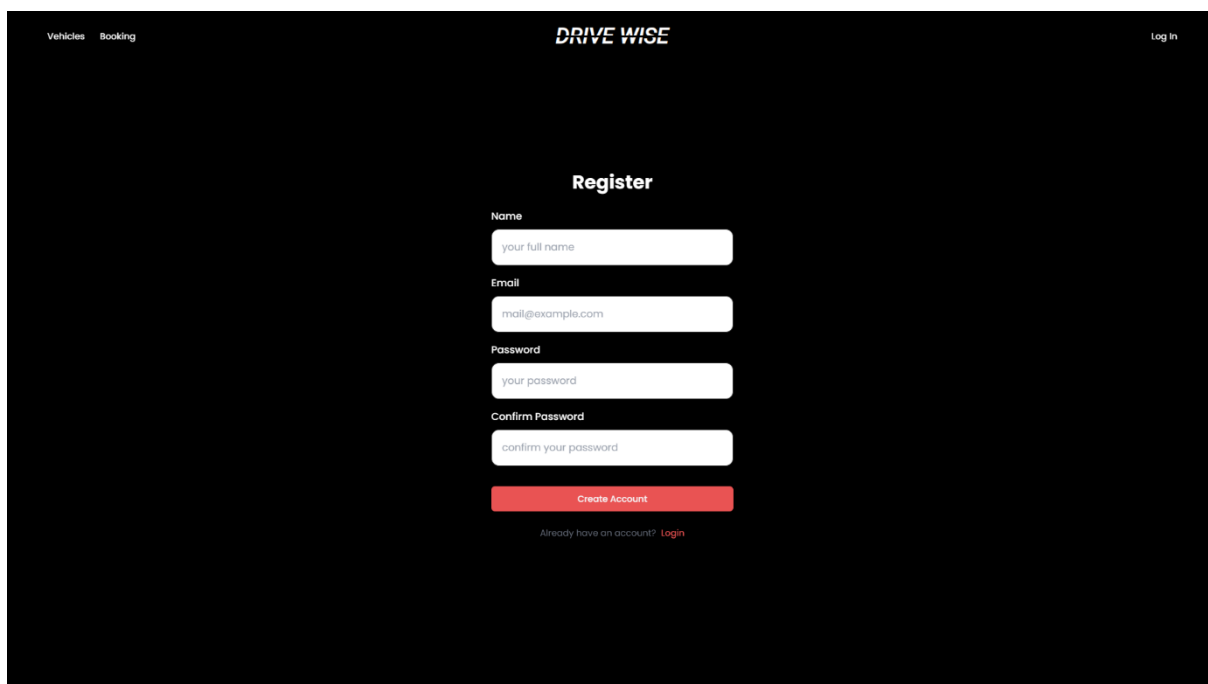
Jauna konta izveidošanai lietotājam ir jānospiež “Register” poga, pēc kā viņš tiks virzīts uz reģistrācijas lapu.



The screenshot shows the 'Login' page of the 'DRIVE WISE' application. The page has a dark background. At the top, there are links for 'Vehicles' and 'Booking' on the left, the 'DRIVE WISE' logo in the center, and a 'Log In' link on the right. The main heading is 'Login'. Below it, there are two input fields: 'Email' with the placeholder 'mail@example.com' and 'Password' with the placeholder 'your password'. A red 'Sign In' button is positioned below the password field. At the bottom, there is a link that says 'Don't have an account? Register'.

6.27. att. Autentifikācijas skats

Reģistrācijas lapā ir ievadlauki vārdam, e-pasta adresei, parolei un paroles apstiprināšanai. Kļūdu gadījumā lietotājs tiek informēts par formatējuma prasībām. Lietotājs tiek reģistrēts pēc “Create Account” pogas nospiešanas. Ievadītie dati tiek pārbaudīti un tiek izvadīts atbilstošs paziņojums par neatbilstībām.



The screenshot shows the 'Register' page of the 'DRIVE WISE' application. The page has a dark background. At the top, there are links for 'Vehicles' and 'Booking' on the left, the 'DRIVE WISE' logo in the center, and a 'Log In' link on the right. The main heading is 'Register'. Below it, there are four input fields: 'Name' with the placeholder 'your full name', 'Email' with the placeholder 'mail@example.com', 'Password' with the placeholder 'your password', and 'Confirm Password' with the placeholder 'confirm your password'. A red 'Create Account' button is positioned below the 'Confirm Password' field. At the bottom, there is a link that says 'Already have an account? Login'.

6.28. att. Lietotāja reģistrācijas skats

Vehicles Booking

DRIVE WISE

Register

Name  
Michael Smith

Email  
michaelsmith@yahoo.com

Password  
\*\*\*\*\*

Confirm Password  
\*\*\*\*\*

Create Account

Already have an account? [Login](#)

Password must be at least 8 characters long, include at least one uppercase letter, one lowercase letter, one number, and one special character.

6.29. att. Lietotāja reģistrācijas skats ar paroles prasību paziņojumu

Vehicles Booking

DRIVE WISE

Register

Name  
Michael Smith

Email  
michaelsmith@yahoo.com

Password  
\*\*\*\*\*

Confirm Password  
\*\*\*\*\*

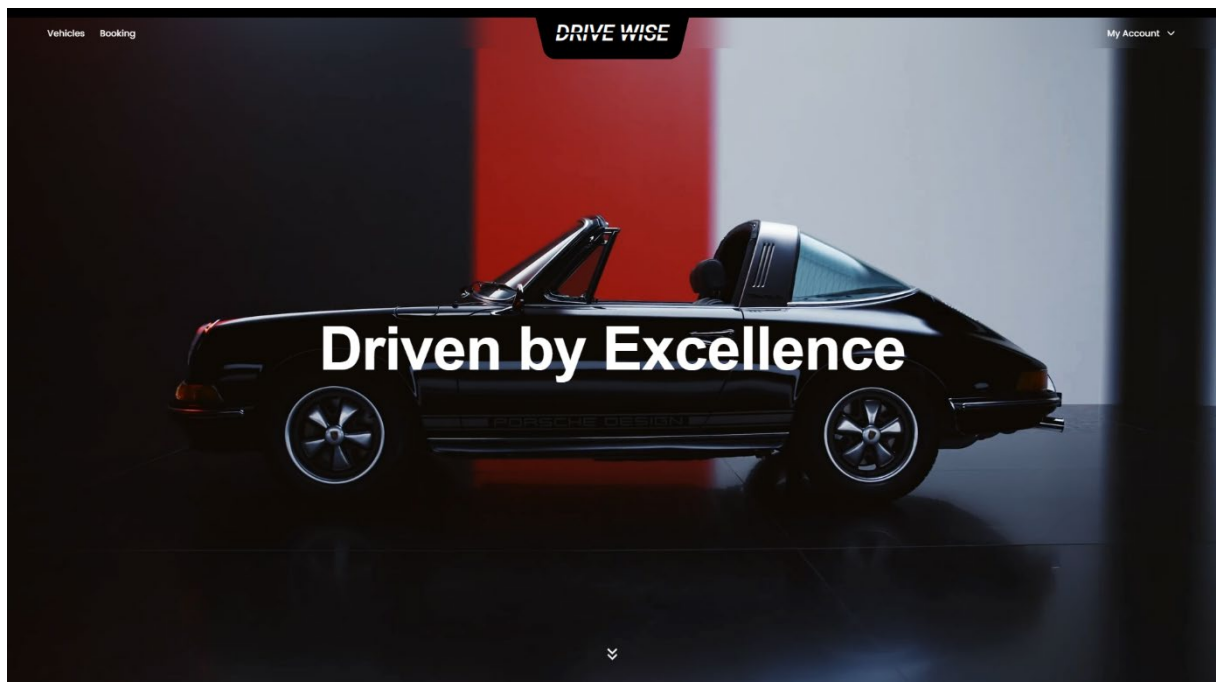
Create Account

Already have an account? [Login](#)

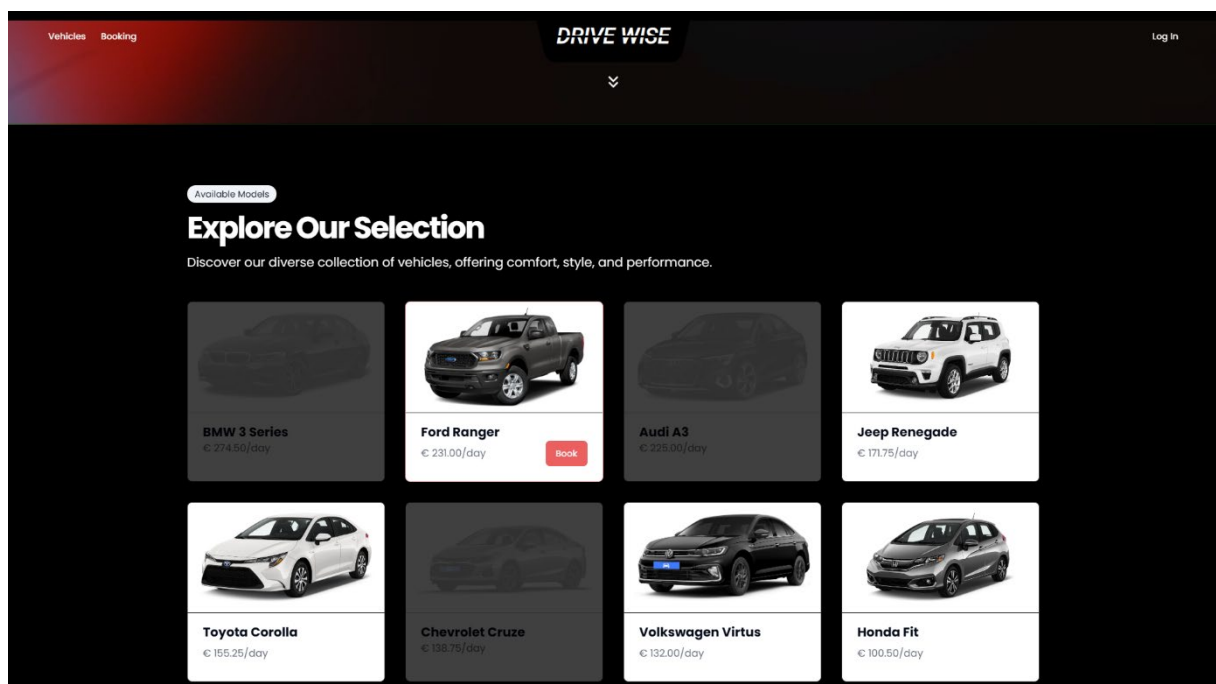
Passwords do not match.

6.30. att. Lietotāja reģistrācijas skats ar paroļu nesakritības paziņojumu

Pēc veiksmīgas reģistrācijas lietotājs tiek pārvietots uz sākumlapu, kur viņš var iesākt rezervācijas procesu.



6.31. att. Sākuma lapas skats



6.32. att. Automobiļu kataloga skats

Pogas “Book” nospiešanai seko pirmais rezervācijas soli, tiek aizpildīta informācijas par vietu, kur automobīlis tiks saņemts un nodots, laiks un datums.

**DRIVE WISE**

Vehicles Booking Log In

1 Date Selection 2 Vehicle Selection 3 Verification

**Select the pick up and drop off location**

Vehicle Location  
Latvia - Riga, Brīvības iela 123

**Select the date to proceed with the reservation**

Start Date  
06/25/2024 01:43 AM

End Date  
06/26/2024 01:44 AM

Continue

**Ford Ranger**  
2023

**Details:**  
Pickup  
Automatic  
Highway kmpt: 10.4  
Kmpt: 8.7

6.33. att. Pirmā rezervācijas soļa skats

Otrajā rezervācijas solī turpinām rezervāciju ar izvēlēto automobīli un nonākam rezervācijas apkopojumā ar pogu “Book Vehicle”.

**DRIVE WISE**

Vehicles Booking Log In

1 Date Selection 2 Vehicle Selection 3 Verification

**Time to choose your car!**

 <b>Renault Kwid</b> € 64.00/day	 <b>Fiat Mobi</b> € 65.00/day	 <b>Toyota Etios</b> € 76.00/day
 <b>Chevrolet Onix</b> € 81.00/day	 <b>Hyundai HB20</b> € 85.00/day	 <b>Nissan Versa</b> € 93.00/day
 <b>Honda Fit</b>	 <b>Volkswagen Virtus</b>	 <b>Chevrolet Cruze</b>

**Ford Ranger**  
2023

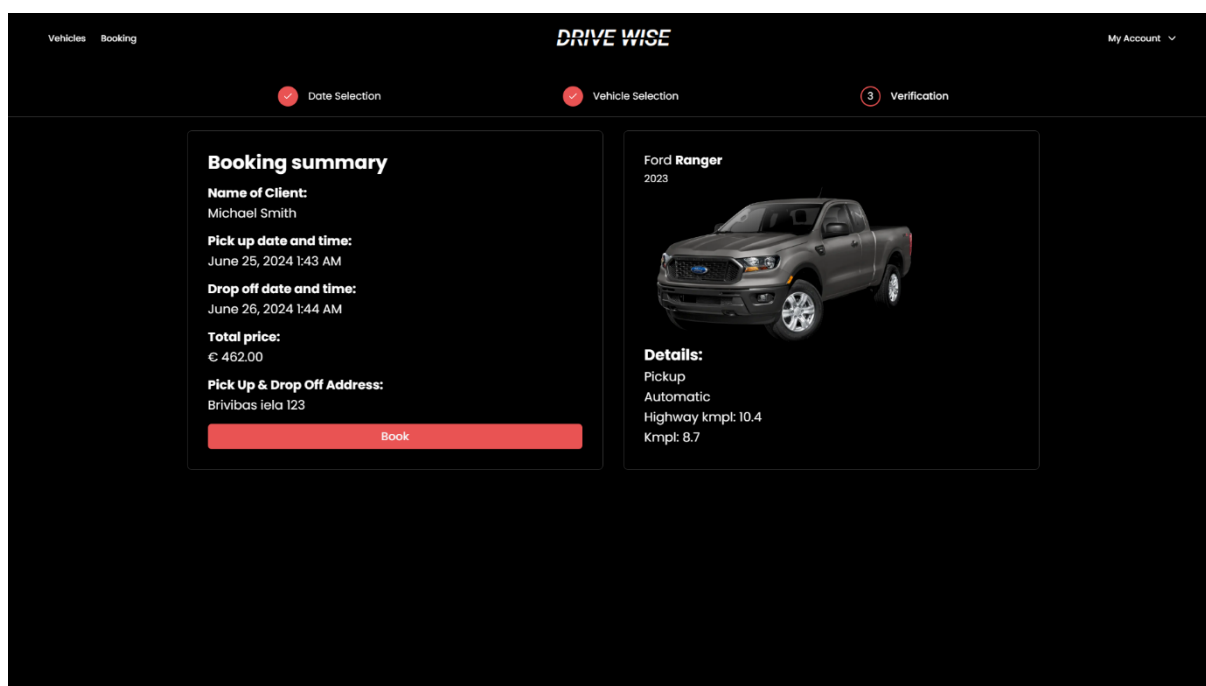
**Details:**  
Pickup  
Automatic  
Highway kmpt: 10.4  
Kmpt: 8.7

Book Vehicle

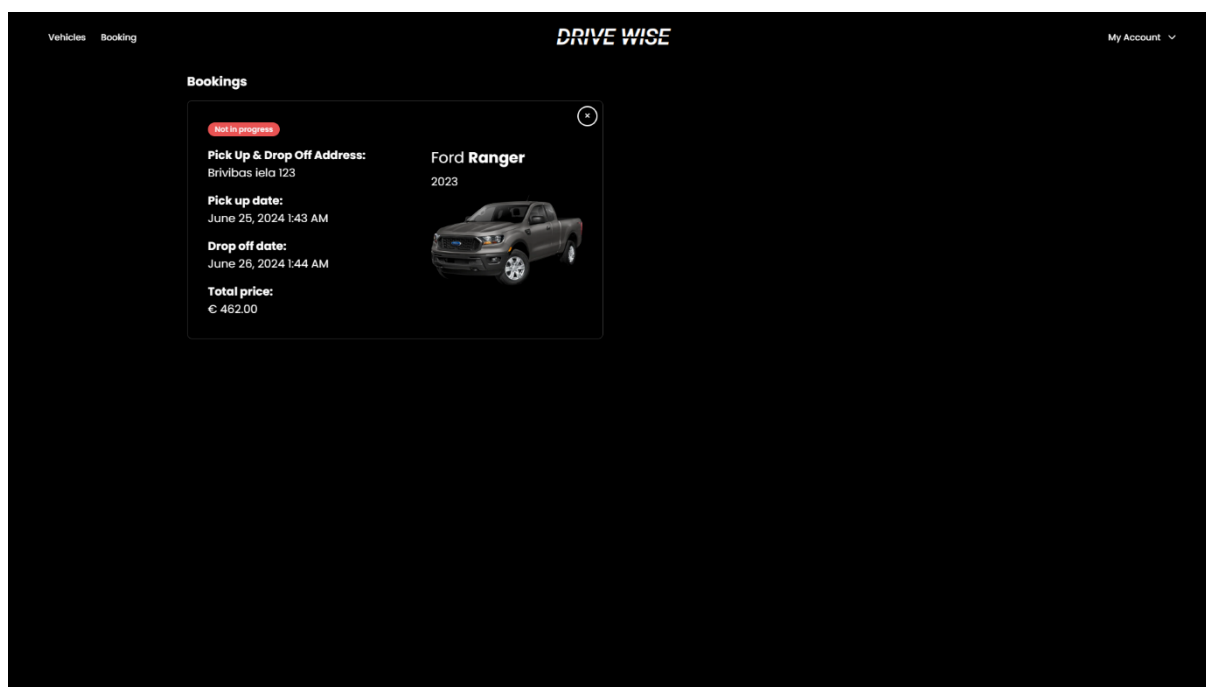
6.34. att. Otrā rezervācijas soļa skats



Trešajā solī saņemam datu apkopojumu un pabeidzam rezervācijas procesu ar pogu “Book”. Pēc pogas nospiešanas lietotājs tiek pārvietots uz vēsture pārskatu.



6.35. att. Trešā rezervācijas soļa skats



6.36. att. Rezervāciju vēstures skats

## NOBEIGUMS

Kvalifikācijas darba izvirzītais mērķis izstrādāt automašīnu izīrēšanas platformu ar intuitīvu pielietojumu un administratīvu pārvaldi ir sasniegts.

Sistēma nodrošina neregistrētiem lietotājiem automobiļu pārskatu, autorizāciju. Reģistrētiem lietotājiem tiek piedāvāta izveidoto kontu datu rediģēšana, rezervāciju izveidošana un dzēšana, vēstures uzrādīšana. Administratoriem tiek piedāvāti dati par visiem sistēmas lietotājiem un rezervācijām, tiek nodrošināta statusa kontrole rezervācijām un lietotāju dzēšanas un privilēģiju maiņas kontroles.

Izstrādes laikā tika pilnveidotas zināšanas darbā ar Next.js un TypeScript, kā arī apgūta darbība ar Prisma ORM. Tika attīstītas zināšanas API strukturēšanu, satura stilizāciju un tīmekļa vietņu izveidi.

Potenciāli sistēmas uzlabojumi ietver plašāku automobiļu piedāvājumu, vairāk saņemšanas un nodošanas vietu, kā arī tām piesaistīta kartes integrācija. Šī funkcionalitāte padarītu sistēmas lietošanu ērtāku lietotājiem, tādejādi uzlabojot kopējo lietošanas pieredzi.

## INFORMĀCIJAS AVOTI

1. ER diagramma - <https://www.lucidchart.com/pages/er-diagrams>
2. UML diagramma - <https://www.lucidchart.com/pages/uml-use-case-diagram>
3. Datu plūsmas diagramma - <https://www.lucidchart.com/pages/data-flow-diagram>
4. Next.js dokumentācija - <https://nextjs.org/docs>
5. NextAuth.js dokumentācija - <https://next-auth.js.org/getting-started/introduction>
6. Typescript dokumentācija - <https://www.typescriptlang.org/>
7. React-toastify dokumentācija - <https://fkhadra.github.io/react-toastify/introduction>
8. Lucide React dokumentācija - <https://lucide.dev/guide/packages/lucide-react>
9. React DayPicker dokumentācija - <https://react-day-picker.js.org/>
10. Radix dokumentācija - <https://www.radix-ui.com/themes/docs/overview/getting-started>
11. Material UI dokumentācija - <https://mui.com/material-ui/getting-started/>
12. Tailwind CSS dokumentācija - <https://tailwindcss.com/>
13. Prisma dokumentācija - <https://www.prisma.io/>
14. PostgreSQL dokumentācija - <https://www.postgresql.org/>

# PIELIKUMI

## 1. pielikums

### Reģistrēšanās metode

```
export async function POST(req: Request) {
  try {
    const body = await req.json();

    const { name, email, password } = body;

    const existingUserByEmail = await prisma.user.findUnique({
      where: { email },
    });
    if (existingUserByEmail) {
      return NextResponse.json(
        { user: null, message: "This e-mail is already in use" },
        { status: 409 }
      );
    }

    const hashPassword = await hash(password, 10);
    const newUser = await prisma.user.create({
      data: {
        name,
        email,
        password: hashPassword,
      },
    });
    const { password: newUserPassword, ...rest } = newUser;

    return NextResponse.json(
      { user: newUser, message: "User created successfully" },
      { status: 201 }
    );
  } catch (error: any) {
    return NextResponse.json({ message: error.message }, { status: 500 });
  }
}
```

**Lietotāja datu maiņas metode**

```
export async function PATCH(req: Request) {
  try {
    const body = await req.json();
    const { email, name, newEmail, password } = body;

    const existingUser = await prisma.user.findUnique({
      where: { email },
    });

    if (!existingUser) {
      return NextResponse.json(
        { message: "User not found with provided email" },
        { status: 404 }
      );
    }

    const updateData: { name?: string; email?: string; password?: string } = {};
    if (name) updateData.name = name;
    if (newEmail) updateData.email = newEmail;
    if (password) updateData.password = await hash(password, 10);

    const updatedUser = await prisma.user.update({
      where: { email },
      data: updateData,
    });

    return NextResponse.json(
      { user: updatedUser, message: "User updated successfully" },
      { status: 200 }
    );
  } catch (error: any) {
    return NextResponse.json({ message: error.message }, { status: 500 });
  }
}
```

**Lietotāja dzēšanas metode**

```
export async function DELETE(req: Request) {
  try {
    const body = await req.json();
    const { email } = body;

    const existingUser = await prisma.user.findUnique({
      where: { email },
    });

    if (!existingUser) {
      return NextResponse.json(
        { message: "User not found with provided email" },
        { status: 404 }
      );
    }

    await prisma.user.delete({
      where: { email },
    });

    return NextResponse.json(
      { message: "User account deleted successfully" },
      { status: 200 }
    );
  } catch (error: any) {
    return NextResponse.json({ message: error.message }, { status: 500 });
  }
}
```

**Automašīnu izvadīšanas metode**

```
export async function GET(req: NextRequest) {
  if (req.method === "GET") {
    const vehicles = await prisma.car.findMany({
      orderBy: [{ rental_price: "asc" }],
    });

    const vehiclesWithPrice = vehicles.map((vehicle) => {
      return {
        ...vehicle,
        rental_price: Math.ceil(75 * vehicle.rental_factor),
      };
    });

    return NextResponse.json({ vehicles: vehiclesWithPrice }, { status: 200 });
  } else {
    return NextResponse.json(
      { message: "Method not allowed", method: req.method },
      { status: 200 }
    );
  }
}
```

**Īres izveidošanas metode**

```

export async function POST(req: Request) {
  try {
    const body = await req.json();
    const { dealership, car, userEmail, startDate, finishDate } = body;

    const totalPrice =
      car.rental_price * (dayjs(finishDate).diff(startDate, "day") + 1);

    const newRental = await prisma.rental.create({
      data: {
        dealership: { connect: { id: dealership.id } },
        car: { connect: { id: car.id } },
        user: { connect: { email: userEmail } },
        startDate,
        endDate: finishDate,
        totalPrice,
        ongoing: false,
      },
      include: {
        user: { select: { id: true, name: true, email: true } },
        car: true,
      },
    });

    await prisma.car.update({
      where: { id: car.id },
      data: { available: false },
    });

    return NextResponse.json({ newRental });
  } catch (error: any) {
    return NextResponse.json({ message: error.message }, { status: 500 });
  }
}

```



**Īres dzēšanas metode**

```
export async function DELETE(req: Request) {  
  try {  
    const body = await req.json();  
    const { rentalId } = body;  
  
    const rental = await prisma.rental.findUnique({  
      where: { id: rentalId },  
      include: { car: true },  
    });  
  
    if (!rental) {  
      return NextResponse.json({ message: "Rental not found" }, { status: 404 });  
    }  
  
    await prisma.rental.delete({  
      where: { id: rentalId },  
    });  
  
    await prisma.car.update({  
      where: { id: rental.car.id },  
      data: { available: true },  
    });  
  
    return NextResponse.json({ message: "Rental deleted successfully" }, { status: 200 });  
  } catch (error: any) {  
    return NextResponse.json({ message: error.message }, { status: 500 });  
  }  
}
```

**Īres izvadišanas metode**

```
export async function GET() {  
  try {  
    const rentals = await prisma.rental.findMany({  
      include: {  
        dealership: true,  
        car: true,  
        user: { select: { id: true, name: true, email: true } },  
      },  
    });  
  
    return NextResponse.json({ rentals }, { status: 200 });  
  } catch (error: any) {  
    return NextResponse.json({ message: error.message }, { status: 500 });  
  }  
}
```

**Īres statusa maiņas metode**

```
export async function PATCH(req: Request) {
  try {
    const body = await req.json();
    const { rentalId } = body;

    const rental = await prisma.rental.findUnique({
      where: { id: rentalId },
    });

    if (!rental) {
      return NextResponse.json(
        { message: "Rental not found" },
        { status: 404 }
      );
    }

    const updatedRental = await prisma.rental.update({
      where: { id: rentalId },
      data: {
        ongoing: !rental.ongoing,
      },
    });

    return NextResponse.json(
      { rental: updatedRental, message: "Rental status updated successfully" },
      { status: 200 }
    );
  } catch (error: any) {
    return NextResponse.json({ message: error.message }, { status: 500 });
  }
}
```

**Vietas izvadišanas metode**

```
export async function GET(req: NextRequest) {  
  if (req.method === "GET") {  
    try {  
      const dealerships = await prisma.dealership.findMany(); // Fetch all dealerships from the database  
  
      return NextResponse.json({ dealerships }, { status: 200 }); // Return dealerships  
    } catch (error: any) {  
      return NextResponse.json(  
        { message: "Failed to fetch dealerships", error: error.message },  
        { status: 500 }  
      );  
    }  
  } else {  
    return NextResponse.json(  
      { message: "Method not allowed", method: req.method },  
      { status: 405 }  
    );  
  }  
}
```

**Administratora privilēģiju metode**

```

export async function POST(req: Request) {
  try {
    const { email, isAdmin } = await req.json();

    // Find the user by email
    const existingUser = await prisma.user.findUnique({
      where: { email },
    });

    // If user doesn't exist, return an error
    if (!existingUser) {
      return NextResponse.json(
        { message: "User not found with provided email" },
        { status: 404 }
      );
    }

    // Toggle the admin status
    const updatedUser = await prisma.user.update({
      where: { email },
      data: {
        admin: !existingUser.admin, // Toggle the admin status
      },
    });

    return NextResponse.json(
      { user: updatedUser, message: "User admin status updated successfully" },
      { status: 200 }
    );
  } catch (error: any) {
    return NextResponse.json({ message: error.message }, { status: 500 });
  }
}

```

**Lietotāju saraksta izvadīšanas metode**

```
export async function GET(req: NextRequest) {
  if (req.method === "GET") {
    try {
      const users = await prisma.user.findMany({
        select: {
          id: true,
          name: true,
          email: true,
          createdAt: true,
          updatedAt: true,
          admin: true,
        }
      });
      return NextResponse.json({ users }, { status: 200 });
    } catch (error: any) {
      return NextResponse.json({ message: "Failed to fetch users", error: error.message }, { status: 500 });
    }
  } else {
    return NextResponse.json(
      { message: "Method not allowed", method: req.method },
      { status: 405 }
    );
  }
}
```

**Lietotāja konta un saistītās īres dzēšanas procesa metode**

```

export async function DELETE(req: Request) {
  try {
    const body = await req.json();
    const { email } = body;
    const user = await prisma.user.findUnique({
      where: {
        email,
      },
      include: {
        rentals: true,
      },
    });
    if (!user) {
      return NextResponse.json({ message: "User not found" }, { status: 404 });
    }
    await Promise.all(
      user.rentals.map(async (rental) => {
        await prisma.rental.delete({
          where: {
            id: rental.id,
          },
        });
      })
    );
    await prisma.user.delete({
      where: {
        email,
      },
    });
    return NextResponse.json({ message: "User and associated rentals deleted successfully" }, { status:
200 });
  } catch (error: any) {
    return NextResponse.json({ message: error.message }, { status: 500 });
  }
}

```

**Prisma modeļu shēma**

```

model User {
  id      String  @id @default(cuid())
  name    String
  email   String  @unique
  password String?
  emailVerified DateTime?
  image   String?
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt
  rentals Rental[]
  admin   Boolean  @default(false)
}

```

```

model Car {
  id      String  @id @default(cuid())
  kmpl    Float
  highway_kmpl Float
  category String
  transmission String
  fuel_type String
  manufacturer String
  model   String
  year    Int
  slug    String
  rental_factor Float
  rental_price Float
  available Boolean @default(true)
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt
  rentals Rental[]
}

```

```

model Rental {
  id      String  @id @default(cuid())
  user    User    @relation(fields: [userId], references: [id])
  userId  String

```



```

car      Car      @relation(fields: [carId], references: [id])
carId    String
startDate DateTime
endDate  DateTime
ongoing  Boolean
totalPrice Float
createdAt DateTime @default(now())
updatedAt DateTime @updatedAt
dealership Dealership @relation(fields: [dealershipId], references: [id])
dealershipId String
}

```

```

model Dealership {
  id      String @id @default(cuid())
  country String
  city    String
  address String
  email   String
  phone   String
  rentals Rental[]
}

```

**Datu bāzes sākotnējie inicializācijas dati**

```
const dealerships = [  
  {  
    country: "Latvia",  
    city: "Riga",  
    address: "Brivibas iela 123",  
    email: "email1@example.com",  
    phone: "+371 12345678"  
  },  
  {  
    country: "Latvia",  
    city: "Riga",  
    address: "Lacplesa iela 456",  
    email: "email2@example.com",  
    phone: "+371 24567890"  
  },  
];  
const cars = [  
  {  
    kmpl: 13.5,  
    highway_kmpl: 16.2,  
    category: "subcompact",  
    transmission: "manual",  
    fuel_type: "flex",  
    manufacturer: "Fiat",  
    model: "Mobi",  
    year: 2023,  
    rental_factor: 0.86,  
    slug: "fiat-mobi-2023",  
  },  
  {  
    kmpl: 12.9,  
    highway_kmpl: 15.7,  
    category: "compact",  
    transmission: "manual",  
    fuel_type: "flex",  
    manufacturer: "Chevrolet",
```

```
model: "Onix",
year: 2023,
rental_factor: 1.07,
slug: "chevrolet-onix-2023",
},
{
  kmpl: 14.2,
  highway_kmpl: 16.9,
  category: "subcompact",
  transmission: "manual",
  fuel_type: "flex",
  manufacturer: "Renault",
  model: "Kwid",
  year: 2022,
  rental_factor: 0.849,
  slug: "renault-kwid-2022",
},
{
  kmpl: 12.7,
  highway_kmpl: 15.0,
  category: "compact",
  transmission: "manual",
  fuel_type: "flex",
  manufacturer: "Toyota",
  model: "Etios",
  year: 2020,
  rental_factor: 1.041,
  slug: "toyota-etios-2020",
},
{
  kmpl: 12.3,
  highway_kmpl: 14.5,
  category: "compact",
  transmission: "manual",
  fuel_type: "flex",
  manufacturer: "Hyundai",
  model: "HB20",
  year: 2022,
```

```
rental_factor: 1.12,  
slug: "hyundai-hb20-2022",  
},  
{  
  kmpl: 12.6,  
  highway_kmpl: 15.2,  
  category: "sedan",  
  transmission: "manual",  
  fuel_type: "flex",  
  manufacturer: "Nissan",  
  model: "Versa",  
  year: 2023,  
  rental_factor: 1.23,  
  slug: "nissan-versa-2023",  
},  
{  
  kmpl: 12.8,  
  highway_kmpl: 15.4,  
  category: "compact",  
  transmission: "manual",  
  fuel_type: "flex",  
  manufacturer: "Honda",  
  model: "Fit",  
  year: 2022,  
  rental_factor: 1.34,  
  slug: "honda-fit-2022",  
},  
{  
  kmpl: 12.1,  
  highway_kmpl: 14.2,  
  category: "sedan",  
  transmission: "automatic",  
  fuel_type: "flex",  
  manufacturer: "Chevrolet",  
  model: "Cruze",  
  year: 2023,  
  rental_factor: 1.85,  
  slug: "chevrolet-cruze-2023",
```

```

    },
    {
      kmpl: 11.9,
      highway_kmpl: 14.1,
      category: "sedan",
      transmission: "automatic",
      fuel_type: "flex",
      manufacturer: "Volkswagen",
      model: "Virtus",
      year: 2023,
      rental_factor: 1.76,
      slug: "volkswagen-virtus-2023",
    },
    {
      kmpl: 12.3,
      highway_kmpl: 14.6,
      category: "sedan",
      transmission: "automatic",
      fuel_type: "flex",
      manufacturer: "Toyota",
      model: "Corolla",
      year: 2023,
      rental_factor: 2.07,
      slug: "toyota-corolla-2023",
    },
    {
      kmpl: 8.7,
      highway_kmpl: 10.4,
      category: "pickup",
      transmission: "automatic",
      fuel_type: "diesel",
      manufacturer: "Ford",
      model: "Ranger",
      year: 2023,
      rental_factor: 3.08,
      slug: "ford-ranger-2023",
    },
    {

```

```

    kmpl: 10.2,
    highway_kmpl: 12.1,
    category: "SUV",
    transmission: "automatic",
    fuel_type: "flex",
    manufacturer: "Jeep",
    model: "Renegade",
    year: 2021,
    rental_factor: 2.29,
    slug: "jeep-renegade-2021",
  },
  {
    kmpl: 13.1,
    highway_kmpl: 15.5,
    category: "hatch",
    transmission: "manual",
    fuel_type: "gas",
    manufacturer: "Audi",
    model: "A3",
    year: 2022,
    rental_factor: 3.0,
    slug: "audi-a3-2022",
  },
  {
    kmpl: 11.8,
    highway_kmpl: 14.0,
    category: "sedan",
    transmission: "automatic",
    fuel_type: "gas",
    manufacturer: "BMW",
    model: "3 Series",
    year: 2021,
    rental_factor: 3.66,
    slug: "bmw-3-series-2021",
  },
];

```