

**RĪGAS VALSTS TEHNIKUMS**

**DATORIKAS NODAĻA**

Izglītības programma: Programmēšana

**KVALIFIKĀCIJAS DARBS**

**“Kvestu istabu rezervēšanas sistēma”**

Paskaidrojošais raksts 75 lpp.

Audzēknis:

Kirils Ivanovs

Prakses vadītājs:

Ilona Demčenko

Nodaļas vadītājs:

Normunds Barbāns

**Rīga 2024**

# ANOTĀCIJA

Šajā kvalifikācijas darbā tiek pētīts un analizēts tīmekļa sistēmas izstrādes process, kas ir izstrādāta, lai uzlabotu evakuācijas istabu rezervēšanas un pārvaldības procesus. Galvenā uzmanība tiek pievērsta sistēmas izstrādei klientam “Quest Room Management”. Sistēma tika izstrādāta, izmantojot PHP un JavaScript programmēšanas valodas, kā arī HTML/CSS un MySQL datubāzi.

Darba struktūra ietver ievadu, problēmas izklāstu, izvēlēto programmatūras risinājumu pamatojumu, modelēšanu un projektēšanu, datu struktūras aprakstu, lietotāja rokasgrāmatu, secinājumus, atsauces un pielikumus. Ievadā aprakstīta problēma un sistēmas nozīme, savukārt problēmas izklāstā sniegts īss sistēmas galveno uzdevumu un funkciju apraksts. Prasību specifikācija ietver ieejas un izejas datu aprakstus un iedala prasības funkcionālās un nefunkcionālās kategorijās. Datu struktūras aprakstā detalizēti aplūkoti datu bāzes tabulu lauki un to savstarpējās attiecības. Lietotāja rokasgrāmatā sniegta informācija par sistēmas prasībām, uzstādīšanu, programmas aprakstu un testēšanas piemēriem.

Izvēlēto programmatūras risinājumu pamatojumā ir izskaidrotas izvēlētās tehnoloģijas un to priekšrocības, tostarp tehnoloģiju kopuma izmantošana, lai nodrošinātu sistēmas uzticamību un mērogojamību. Modelēšanas un projektēšanas sadaļā ir izklāstīti sistēmas struktūras un funkcionalitātes modeļi. Datu struktūras aprakstā detalizēti aplūkoti datubāzes tabulu lauki un to attiecības. Lietotāja rokasgrāmatā iekļauta informācija par sistēmas prasībām, instalāciju, programmas aprakstu un testēšanas piemēriem. Kopumā šajā darbā sniegts detalizēts pārskats par tīmekļa sistēmas izstrādes procesu, kas atbilst “Quest Room Management” specifiskajām vajadzībām un prasībām, piedāvājot plašu informāciju par sistēmas izveidi un darbību, kā arī skaidru izpratni par izvēlētajām tehnoloģijām un to pielietojumu.

Kvalifikācijas dokuments sastāv no 75 lapām, 35 ilustrācijām, 9 tabulām un 1 pielikumu.

# ANNOTATION

This qualification paper examines and analyzes the process of developing a web system designed to improve the booking and management processes of escape rooms. The primary focus is on developing a system for the client "Quest Room Management." The system was developed using PHP and JavaScript programming languages, as well as HTML/CSS and a MySQL database.

The structure of the paper includes an introduction, problem statement, justification for the chosen software solutions, modeling and design, data structure description, user manual, conclusion, references, and appendices. The introduction describes the problem and the relevance of the system, while the problem statement provides a brief description of the main tasks and functionalities of the system. The requirements specification includes descriptions of input and output data and categorizes requirements into functional and non-functional categories.

The justification for the chosen software solutions explains the selected technologies and their advantages, including the use of a technology stack to ensure the system's reliability and scalability. The modeling and design section outlines the models of the system's structure and functionality. The data structure description provides a detailed examination of the database table fields and their relationships. The user manual includes information on system requirements, installation, program description, and testing examples. Overall, this paper provides a detailed overview of the process of developing a web system that meets the specific needs and requirements of "Quest Room Management," offering extensive information on the creation and operation of the system, as well as a clear understanding of the selected technologies and their application.

The qualification paper consists of 75 pages, 35 illustrations, 9 tables, and 1 appendice.

# SATURS

<b>IEVADS</b> .....	<b>5</b>
<b>1. UZDEVUMA NOSTĀDNE</b> .....	<b>6</b>
<b>2. PRASĪBU SPECIFIKĀCIJA</b> .....	<b>8</b>
2.1. Ieejas un izejas informācijas apraksts .....	8
2.1.1. Ieejas informācijas apraksts .....	8
2.1.2. Izejas informācijas apraksts .....	10
2.2. Funkcionālās prasības.....	10
2.3. Nefunkcionālās prasības .....	12
<b>3. UZDEVUMA RISINĀŠANAS LĪDZEKĻU IZVĒLES PAMATOJUMS</b> .....	<b>14</b>
<b>4. PROGRAMMATŪRAS PRODUKTA MODELĒŠANA UN PROJEKTĒŠANA</b>	<b>16</b>
4.1. Sistēmas struktūras modelis.....	16
4.1.1. Sistēmas arhitektūra .....	16
4.1.2. Sistēmas ER modelis.....	17
4.2. Funkcionālais sistēmas modelis .....	20
4.2.1. Datu plūsmu modelis .....	20
<b>5. DATU STRUKTŪRU APRAKSTS</b> .....	<b>24</b>
<b>6. LIETOTĀJA CEĻVEDIS</b> .....	<b>28</b>
6.1. Sistēmas struktūras modelis.....	28
6.2. Sistēmas instalācija un palaišana .....	28
6.3. Programmas apraksts.....	29
6.4. Testa piemērs .....	40
<b>NOBEIGUMS</b> .....	<b>42</b>
<b>INFORMĀCIJAS AVOTI</b> .....	<b>43</b>
<b>PIELIKUMI</b> .....	<b>44</b>
1. pielikums. Programmas pirmkods.....	45

## IEVADS

Šobrīd ir ļoti būtiski izmantot mūsdienīgas tehnoloģijas, kas ne tikai piedāvā ērtu piekļuvi, bet arī padara izklaides kvestu telpu rezervēšanas pieredzi neaizmirstamu. Šajā projektā tiek plānots izmantot modernas tehnoloģijas, kas ļaus klientiem ne tikai viegli atrast interesējošās kvestu istabas, bet arī ātri un ērti tās rezervēt, sniedzot detalizētu informāciju par pieejamību un telpu īpašībām.

Šī tīmekļa vietne ir paredzēta cilvēkiem, kuri meklē unikālas un aizraujošas izklaides piedzīvojumus. Ņemot vērā izklaides nozares attīstību un pieaugošo pieprasījumu pēc kvestu istabām, ir svarīgi nodrošināt vienkāršu un efektīvu rezervēšanas sistēmu.

Projektā plānotajām funkcijām ietilpst iespēja pievienot jaunas kvestu istabas, veikt izmaiņas telpu informācijā un ērti pārvaldīt darbinieku informāciju. Administratīvā līmenī tiks nodrošināta rezervāciju datu pārvaldība, kā arī klientu atsauksmju sistēma, kas ļaus klientiem dalīties ar savu pieredzi.

Lielāka uzmanība tiks pievērsta arī lietotāja pieredzes uzlabošanai, piedāvājot ērtu reģistrāciju un pieteikšanos gan klientiem, gan administratoriem un darbiniekiem. Šī tīmekļa vietne kļūs par centrālo vietu, kur cilvēki varēs atrast un rezervēt izklaides kvestu telpas, padarot šo procesu vienkāršu un pieejamu.

Projekta mērķis ir izveidot tīmekļa vietni, kas atbilst mūsdienīgām prasībām, piedāvājot inovatīvas funkcijas, kas padarīs izklaides kvestu rezervēšanu ērtu un aizraujošu. Šī vietne nesniedz tikai prieku klientiem, bet arī uzņēmumiem, kuri varēs palielināt savu redzamību un efektīvi pārvaldīt kvestu istabu informāciju.

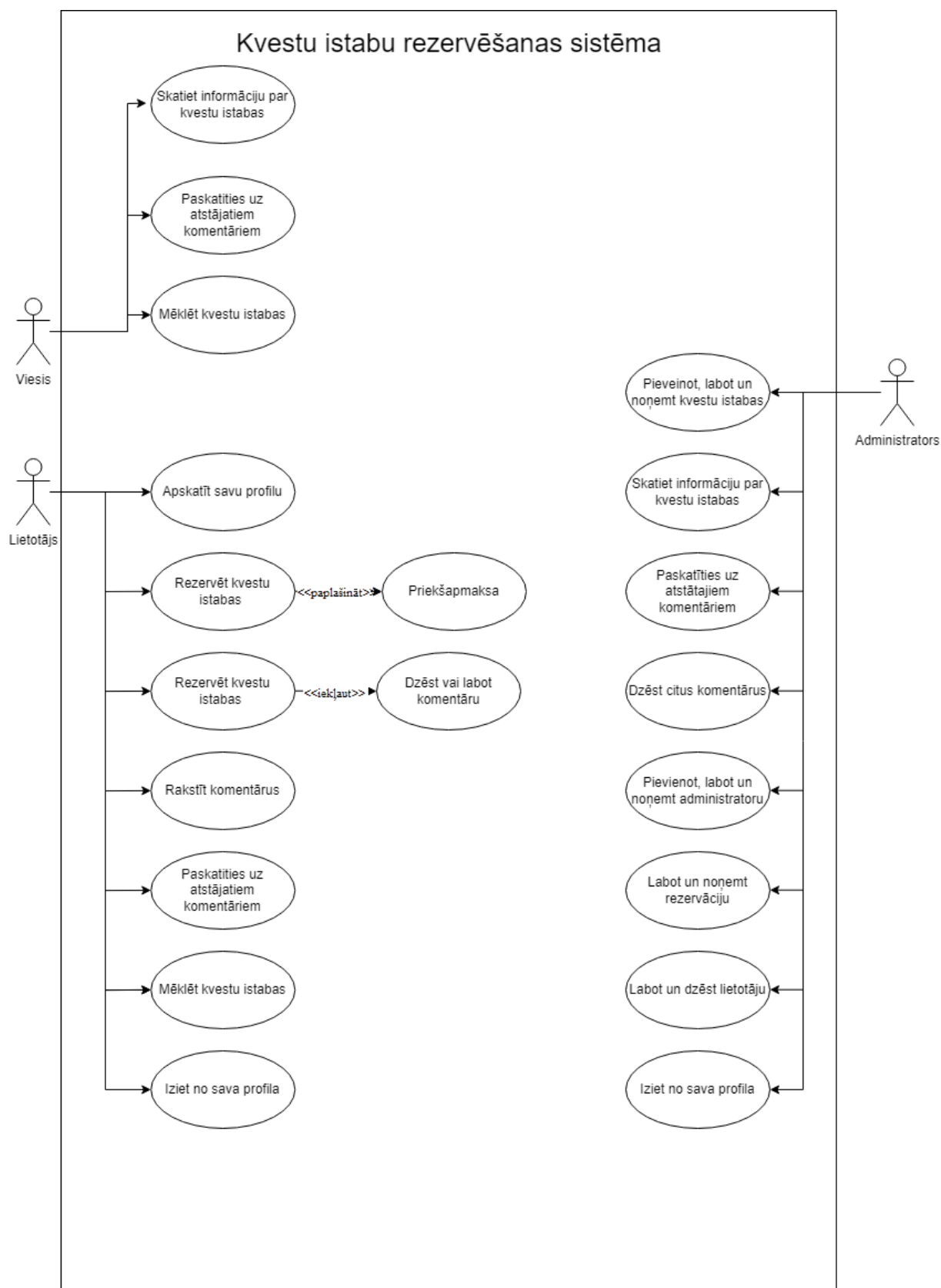
# 1. UZDEVUMA NOSTĀDNE

Kvalifikācijas darba uzdevums ir izveidot tīmekļa vietni, kas sniedz izklaides kvesta telpu rezervēšanas pakalpojumus. Sistēmā nepieciešams realizēt iespēju nodrošināt lietotājam iespēju rezervēt kvestu telpas, kā arī sniegt informāciju par pieejamajām kvestu telpām un to īpašībām.

Līdz ar izklaides industrijas attīstību un kvestu istabu kā aktīvas atpūtas veida popularitāti dažādu vecuma grupu vidū, ir radies pieprasījums pēc tiešsaistes platformas, kas atvieglo šo izklaides pasākumu rezervēšanu un piekļuvi tiem. Mūsu auditorija ir cilvēki, kas meklē unikālas un aizraujošas izklaides. Pašlaik Rīgā ir tikai viena konkurētspējīga centralizētas tiešsaistes platformas kvestu istabu rezervēšanai ar neērtu funkcionalitāti. Citādāk cilvēkiem ir jāmeklē informācija par kvestu istabām un jāsazinās ar tām tieši, kas var būt neērti un laiktīlpīgi(sk. 1.1. att.).

Ir plānotas vairākas funkcijas:

- iespēja sistēmā pievienot jaunas kvestu istabas, esošo kvestu istabu dzēšana un izmaiņas kvestu istabas informācijā (pieejama tikai administratoram vai darbiniekam);
- darbinieku informācijas pievienošana/pārvietošana/mainīšana (pieejama tikai administratoram);
- klientu rezervācijas datu dzēšana un modificēšana (pieejama tikai administratoram vai darbiniekam);
- iespēja klientiem rezervēt kvestu telpas;
- kvestu istabu meklēšana un filtrēšana pēc dažādiem kritērijiem;
- reģistrācija jauniem klientiem, pieteikšanās administratoriem, darbiniekiem un klientiem;
- iespēja klientiem atstāt atsauksmes par uzdevumu istabām un iespēja atbildēt uz atsauksmēm gan no klienta, gan darbinieka puses.



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.

- Vārds – lietotāja vārds - burtu teksts ar izmēru līdz 128 rakstzīmēm. (piem. “Jānis”)
- Uzvārds – lietotāja uzvārds - burtu teksts ar izmēru līdz 128 rakstzīmēm. (piem. “Bērziņš”)
- Lietotājvārds – lietotāja lietotājvārds - burtu teksts ar izmēru līdz 128 rakstzīmēm. (piem. “Abols32”)
- E-pasts – lietotāja e-pasts - burtu teksts ar izmēru līdz 128 rakstzīmēm. (piem. “pasts@inbox.lv”)
- Parole – lietotāja parole - burtu, icparu un simbolu teksts ar izmēru līdz 128 rakstzīmēm. (piem.”das32FAS42sf”)
- Tālruņa numurs – lietotāja tālruņa numurs - ciparu teksts ar izmēru līdz 30 rakstzīmēm. (piem. 23342345)

2. Informācija par **kvestu istabām** sastāvēs no šādiem datiem.

- Nosaukums – kvestu istabās nosaukums - burtu teksts ar izmēru līdz 128 rakstzīmēm.(piem. “pamesta māja”)
- Apraksts – kvestu istabās apraksts - burtu teksts ar izmēru līdz 5000 rakstzīmēm.
- Attēlas cēļš - Attēlas novietošanās adrese - (piem. ‘../attēli/attēls.png’)
- Cilvēku daudzums – cik cilvēkus var apmeklēt konkrēto kvestu – skaitlis ar diapazonu. (piem. “2-6”)

3. Informācija par **rezervācijām** sastāvēs no šādiem datiem.

- Datums – rezervācijas datums - kalendāra datums, kurā veikta rezervācija. (piem. 11.11.2023)
- Laiks – rezervācijas laiks - rezervācijas laiks, burtu teksts ar izmēru līdz 5 rakstzīmēm.(piem. “12:00”)
- Maksa – lietotāja maksa veids - veids, kādā tika veikta apmaksa, burtu teksts ar izmēru līdz 128 rakstzīmēm. (piem. “skaidra nauda” vai “karte”)
- Cena – kvestu istabās rezervēšanas cena - daļskaitlis ar precizitāti līdz 2 cipariem aiz komata. (piem. 80 euro)



- Izveidošanas datums – rezervācijas izveidošanas datums - kalendāra datums, kurā rezervācija bija izveidota. (piem. 11.11.2023)

#### 4. Informācija par **komentāriem** sastāvēs no šādiem datiem.

- Komentārs – lietotāja vai administratora komentārs - teksta informācija, kuru lietotājs ir pievienojis. (piem. ‘Jānis: Ļoti patika!’)

#### 5. Informācija par **administratoriem** sastāvēs no šādiem datiem.

- Vārds – administratora vārds- burtu teksts ar izmēru līdz 128 rakstzīmēm. (piem. “Jānis”)
- Uzvārds – administratora uzvārds - burtu teksts ar izmēru līdz 128 rakstzīmēm. (piem. “Bērziņš”)
- E-pasts – administratora e-pasts - burtu teksts ar izmēru līdz 128 rakstzīmēm. (piem. “pasts@inbox.lv”)
- Parole – administratora parole - burtu, ciparu un simbolu teksts ar izmēru līdz 128 rakstzīmēm. (piem. “das32FAS42sf”)
- Personas kods – administratora personas kods - identifikācijas kods, burtu un ciparu teksts ar izmēru līdz 12 rakstzīmēm. (piem. “200484-20481”)
- Tālruņa numurs – administratora tālruņa numurs - ciparu teksts ar izmēru līdz 30 rakstzīmēm. (piem. 23342345)

#### 6. Informācija par **cenam** sastāvēs no šādiem datiem.

- Laika periods – rezervācijas laiks - burtu teksts ar izmēru līdz 128 rakstzīmēm. (piem. “10:30”)
- Cena – rezervācijas cena ar iekļautu atlaidi - rezervētā kvesta izmaksas, daļskaitlis ar precizitāti līdz 2 cipariem aiz komata. (piem. “55.50 euro”)

#### 7. Informācija par **adrese** sastāvēs no šādiem datiem.

- Adrese – kvestu istabas atrašanās vieta – burtu teksts ar cipariem līdz 128 rakstzīmēm. (piem. “Viršu iela 10”)

#### 8. Informācija par **adrese** sastāvēs no šādiem datiem.

- Vecums – vesuma ierobežojums – divciparu skaitlis. (piem. “16+”)
- Kategorija – kvestu istabas žanrs - burtu teksts ar cipariem līdz 128 rakstzīmēm. (piem. “Horror”)

### *2.1.2. Izejas informācijas apraksts*

#### **1. Kvīts izvade PDF formātā.**

Pēc rezervēšanas PDF kvītā tiek radīts rezervēšanas dati (nosaukums, adrese), rezervācijas laiks, un cena ar citiem maksājuma datiem vai rezervācijas informāciju.

#### **2. Izvades informācija PDF formātā.**

Administrators var izvadīt datus pdf formātā, piemēram, par ienākumiem, ienesīgāko nedēļas dienu un visbiežāk apmeklēto kvestu telpu.

#### **3. E-pasta paziņojums lietotājiem ar informāciju par rezervāciju.**

Pēc rezervēšanas lietotājam tiks nosūtīts e-pasts ar informāciju par rezervāciju. E-pastā būs iekļauts sveiciens, rezervētās automašīnas dati un kontaktinformācija.

## **2.2. Funkcionālās prasības**

### **1. Kvestu istabu pārvaldība.**

1.1 Administratoriem jābūt iespējai sistēmā pievienot jaunas uzdevumu istabas. Un arī nedrīkst atkārtot informāciju par telpām, piemēram, telpas nosaukumu, fotoattēlu un aprakstu.

1.2. Jābūt iespējai rediģēt esošo uzdevumu istabu informāciju.

1.3. Administratoriem jābūt iespējai dzēst kvestu istabas no sistēmas. Dzēšot kvestu istabu, tiek dzēsta visa ar šo istabu saistītā informācija, piemēram, visas lietotāju rezervācijas un komentāri, kas atstāti par šo istabu.

### **2. Darbinieku pārvaldība.**

2.1. Administratoriem jābūt iespējai pievienot, dzēst, mainīt darbinieku informāciju. Darbiniekiem nav jābūt iespējai dzēst administratorus vai citus darbiniekus vai mainīt viņu informāciju. Dzēšot informāciju par darbinieku, no datubāzes tiks dzēsta visa saistītā informācija par šo darbinieku, piemēram, komentāri un atbildes uz lietotāju komentāriem.

### **3. Rezervāciju pārvaldība.**

3.1. Administratoriem jābūt iespējai dzēst un mainīt informāciju par klientu rezervācijām. Dzēšot vai mainot rezervācijas informāciju šīs kvestu telpas lapā, tiks atbrīvots aizņemtais datums un aizņemts datums, kas bija ieplānots.

3.2. Jābūt iespējai rediģēt rezervācijas informāciju, piemēram, datumu, laiku un dalībnieku skaitu. Mainot rezervācijas informāciju, šīs kvestu telpas lapā tiks atbrīvots datums, kas bija aizņemts, un tiks aizņemts datums, kas bija plānots.

### **4. Lietotāju pārvaldība.**

- 4.1. Reģistrētie klienti varēs rezervēt kvestu telpas, izmantojot platformu, kurā klientu un rezervācijas informācija tiks glabāta drošā datubāzē. Pēc apmaksas ar karti klients e-pastā saņems paziņojumu un PDF kvīti. Informācija par jauno rezervāciju tiks pievienota lietotāja profilam.
  - 4.2. Klientiem jābūt iespējai atstāt atsauksmes par piedzīvoto kvestu istabā.
  - 4.3. Lietotājiem jāvar rediģēt savu profilu un pārvaldīt rezervācijas.
5. Meklēšanas un filtrēšanas iespējas.
  - 5.1. Visiem lietotājiem jāvar meklēt kvestu istabas pēc nosaukuma, kategorijas vai atrašanās vietas.
  - 5.2. Jābūt iespējai filtrēt kvestu istabas pēc dažādiem kritērijiem, piemēram, cenas vai atlaides.
6. Sistēmas drošība.
  - 6.1. Lai palielinātu drošību, jābūt autentifikācijai un autorizācijai visās lietotāja klasēs.
  - 6.2. Jābūt sistēmas drošības pasākumiem, lai novērstu neautorizētu piekļuvi vai datu noplūdi.
7. Komentāru sistēma.
  - 7.1. Lietotājiem jāvar rakstīt komentārus par piedzīvotajām kvestu istabām.
  - 7.2. Lietotājiem jāvar dzēst un labot savus komentārus. Ja komentārs tiek dzēsts no datubāzes, no datubāzes tiek dzēsti arī visi ar šo komentāru saistītie komentāri jeb atbildes uz komentāriem.
  - 7.3. Administrātoriem jāvar apskatīt, labot un dzēst citu lietotāju atstātos komentārus.
8. Reģistrācija un pieteikšanās.
  - 8.1. Jābūt iespējai reģistrēties jauniem klientiem. Reģistrācijas laika klientiem jāievada atbilstošus datus vajadzīgos laukos. Ja reģistrācija veikta veiksmīgi, tad lietotāja dati saglabas datu bāzi un uz e-pastu klients saņēmis ziņojums par veiksmīgo reģistrāciju.
  - 8.2. Lietotājiem jāvar izveidot un pārvaldīt savu kontu sistēmā.
  - 8.3. Jābūt iespējai pieteikties administratoriem, darbiniekiem.
9. Atsauksmju sistēma.
  - 9.1. Jābūt iespējai klientiem atstāt atsauksmes par kvestu istabām.
  - 9.2. Administrātoriem jāvar atbildēt uz klientu atsauksmēm.
10. Lietotāja pieredzes uzlabošana.
  - 10.1. Jābūt ērtai reģistrācijai un pieteikšanās procesam.

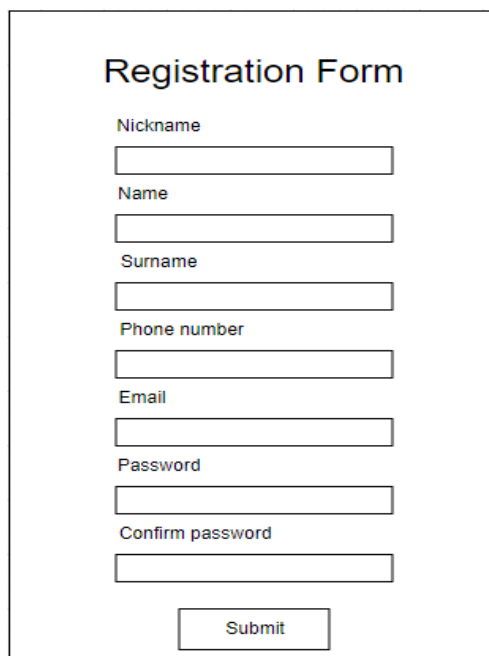
- 10.2. Sistēmai jānodrošina vienkārša un pieejama rezervēšanas procedūra.
11. Iziešana no profila.
- 11.1. Visiem lietotājiem jābūt iespējai droši izlogoties no sistēmas, lai nodrošinātu drošību un aizsargātu personīgo informāciju.
12. Filtrācija un meklēšana kvestu istabu iekšpusē.
- 12.1. Lietotājiem jāvar izmantot filtrus, lai atrastu kvestu istabas pēc specifiskiem kritērijiem, piemēram, cenas diapazona vai kvesta žanra.
- 12.2. Jābūt iespējai veikt ātru meklēšanu pēc nosaukuma vai atrašanās vietas, lai ātri atrastu interesējošās kvestu istabas.
13. Profila dzēšana.
- 13.1. Lietotājiem jābūt iespējai dzēst savu profilu, ja viņi vairs nevēlas izmantot sistēmu.
- 13.2. Dzēšot profilu, jābūt paziņojumam par iespējamo datu dzēšanu un sekām.
14. Papildu drošības pasākumi.
- 14.1. Lai novērstu nevēlamu piekļuvi un datu zādzību, jābūt papildu drošības pasākumiem, piemēram, divpakāpju autentifikācijai vai drošām paroles prasībām.
15. Notifikācijas un e-pasta apstiprinājumi.
- 15.1. Lietotājiem jāsaņem notifikācijas par svarīgiem notikumiem, piemēram, veiksmīgu rezervāciju vai atsauksmju saņemšanu.
- 15.2. Ja nepieciešams, jāiekļauj e-pasta apstiprinājumi pie dažādiem konta darbības veidiem, lai nodrošinātu drošību.

## **2.3. Nefunkcionālās prasības**

1. Jānodrošina tīmekļa lietojumprogrammas pielāgošanas ekrāna izmēriem, kas mūsdienās tiek lietoti, lai to varētu izmantot uz dažādiem monitora izmēriem.
2. Tekstiem uz ekrāna jābūt ne mazākiem par 10 pikseļiem augstumā.
3. Sistēma ir jābūt pieejama 24 stundas diennaktī, 7 dienas nedēļā.
4. Sistēmai jābūt optimizētai, lai tā darbotos arī dažādās operētājsistēmās.
5. Sistēmai jānodrošina lietotāja datu drošība, izmantojot šifrēšanas tehnoloģijas.
6. Dizainam ir jābūt saprotamam un patīkamam lietotājam.
7. Sistēmas saskarnes valodai ir jābūt pieejamai vismaz divas valodas: latviešu, angļu.
8. Rezervācijas process nedrīkst aizņemt vairāk kā 3 minūtes.

Sistēmas ekrānu skices:

- Sistēmas reģistrācijas skice (skat 2.1.att)

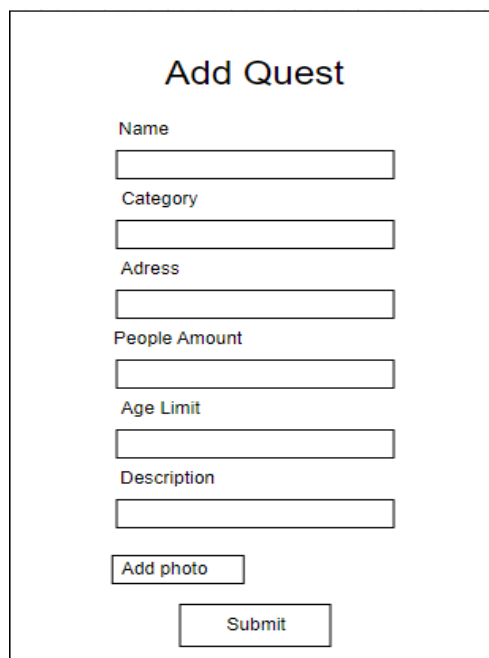


The sketch shows a registration form titled "Registration Form". It contains the following fields from top to bottom: Nickname, Name, Surname, Phone number, Email, Password, and Confirm password. Each field is represented by a rectangular input box. At the bottom of the form is a "Submit" button.

2.1.att. Reģistrācijas formas ekrāna skice

Šajā skicē tiek attēlots sistēmas interfeiss, kurā lietotāji var reģistrēties. Tajā ir 7 lauki un poga “Submit”, kura ļauj reģistrēt jauno lietotāju.

- Sistēmas kvesti istabu izveidošanas skice (skat 2.2.att)



The sketch shows a form titled "Add Quest". It contains the following fields from top to bottom: Name, Category, Address, People Amount, Age Limit, and Description. Each field is represented by a rectangular input box. Below the Description field is a button labeled "Add photo". At the bottom of the form is a "Submit" button.

2.2. att. Reģistrācijas formas ekrāna skice

Šajā skicē tiek attēlots sistēmas interfeiss, kurā administratori var pīvienot informāciju par jauno kvestu istabu. Tajā ir 6 lauki un 2 pogas “Pievienot attēlu” un “Add Quest”, kura ļauj izveidot jauno kvestu istabu.

### 3. UZDEVUMA RISINĀŠANAS LĪDZEKĻU IZVĒLES PAMATOJUMS

Šī sistēma ir paredzēta izmantošanai pārlūkprogrammā. Tā ir izstrādāta, lai nodrošinātu ātru un ērtu darbību. Rīki un tehnoloģijas, kas izmantotas šīs sistēmas izveidē, ir atlasīti, lai optimizētu lietotāja pieredzi. Turklāt šī sistēma ir veidota ar nākotnes perspektīvu. Tā ir pielāgojama un elastīga, tāpēc to var pielāgot arī mobilajām ierīcēm. Tas nozīmē, ka šī sistēma nākotnē būs pieejama gan datorā, gan mobilajā tālrunī.

Lietotāja saskarnes (frontend) izstrādē tika iesaistīti:

- **HTML (versija 5.3)** ir pazīstama arī kā hiperteksta marķēšanas valoda un tiek izmantota tīmekļa lapu saskarnes izstrādē, izmantojot iezīmēšanas valodu. Tās funkcionalitāte ideāli piemērota mūsu sistēmas prasībām.
- **CSS (versija 3)** ir viegli saprotama valoda, kura ir radīta, lai atvieglinātu tīmekļa lapu izskatu veidošanas procesu. CSS ļauj tīmekļa lapām pielāgot stilus, un lietošanas ērtums ir galvenais kritērijs, izvēloties to.
- **JavaScript (versija 1.5)** ir skriptu valoda, ko izmanto, lai piešķirtu mājas lapai pievilcību un padarītu to interaktīvu lietotājiem. Šo valodu pielieto, lai uzlabotu vietnes funkcionalitāti un darbinātu tīmekļa programmatūru.
- **Bootstrap (versija 5.3.3)** ir populārs atvērtā koda tīmekļa izstrādes ietvars, kas apvieno HTML, CSS un JavaScript elementus, lai izveidotu reaģētspējīgas un modernās tīmekļa lapas. Izvēlētajā versijā nodrošina visu nepieciešamo, lai ātri izveidotu stilīgus un lietotājiem draudzīgus interfeisus.

Servera daļa (backend) izstrādē tika iesaistīti:

- **PHP (versija 8.2.0)** ir servera puses programmēšanas valoda, kas tiek izmantota dinamisku tīmekļa lapu izstrādei. Atlasītā versija ir saskaņota ar sistēmas prasībām.
- **MySQL (versija 5.2.1)** ir relāciju datubāzes pārvaldības sistēma. Tā tiek izmantota, lai uzglabātu un efektīvi pārvaldītu sistēmas datus.
- **PHPMailer (versija 6.9)** ir PHP bibliotēka, kas nodrošina ērtu veidu, kā nosūtīt e-pastus no tīmekļa lietojumprogrammas. Tā tiek izmantota, lai izstrādātu un nosūtītu e-pastus, izmantojot mūsu sistēmu.
- **mPDF (versija 8.1.3)** ir PHP bibliotēka, kas ļauj ģenerēt PDF dokumentus no HTML satura. To izmanto, lai izveidotu PDF failus, kas nepieciešami sistēmas funkcionalitātei.

- **Stripe (versija 7.97.0)** ir PHP bibliotēka, kas nodrošina ērtu veidu, kā integrēt maksājumu apstrādes iespējas tīmekļa lietojumprogrammā. Tā tiek izmantota, lai pievienotu drošus un uzticamus maksājumu risinājumus.
- **OpenSSL (versija 3.0.7)** ir kriptogrāfijas programmatūras bibliotēka, kas nodrošina drošības protokolus tīmekļa lietojumprogrammās. Tā tiek izmantota, lai šifrētu datus un nodrošinātu drošu saziņu starp serveri un klientu.

Izstrādē izmantotās programmēšanas vides:

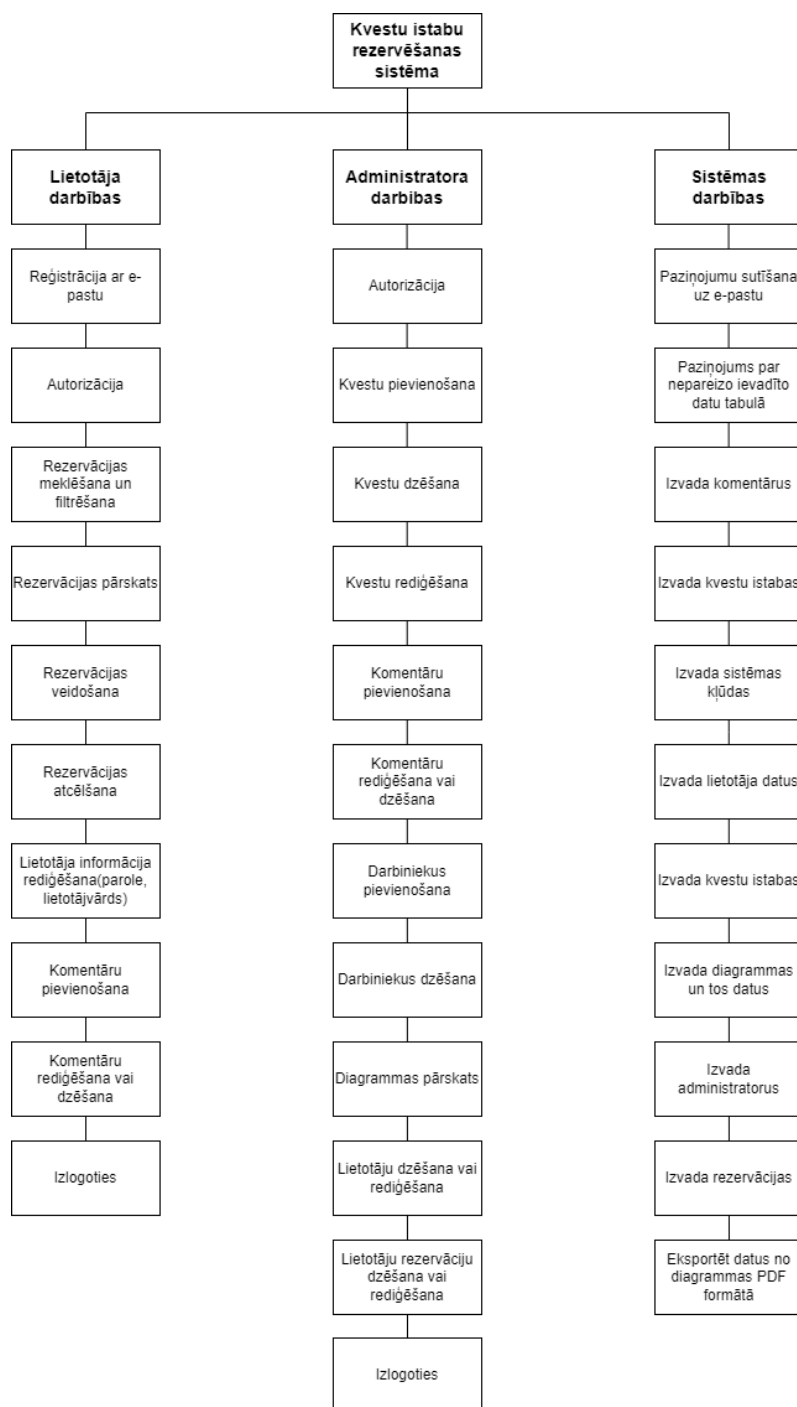
- **phpMyAdmin (versija 5.2.1)** ir MySQL datubāzes pārvaldības rīks ar grafisko saskarni, ko var izmantot, lietojot tīmekļa pārlūkprogrammu. Tas piedāvā ērtu veidu, kā pārvaldīt datubāzi.
- **XAMPP (versija 8.2.0)** ir komplekts integrētas programmatūras, ietverot Apache, MySQL, PHP un citus tīmekļa izstrādes rīkus. Izvēlēta versija ir pielāgota sistēmas prasībām.
- **Visual Studio Code (versija 1.85.0)** ir atvērtā koda izstrādes vidē paredzēta platforma, kas kalpo tīmekļa lietojumprogrammu izveidei. Izvēlēta versija nodrošina visus nepieciešamos rīkus sistēmas izstrādei.

## 4. PROGRAMMATŪRAS PRODUKTA MODELĒŠANA UN PROJEKTĒŠANA

### 4.1. Sistēmas struktūras modelis

#### 4.1.1. Sistēmas arhitektūra

Sistēmu (sk. 4.1. att.) veido trīs apakšsistēmas: lietotāju datu apstrāde, administratora datu apstrāde, sistēmas datu apstrāde.



4.1. att. FD diagramma



- **Lietotāju modulis.** Neregistrētiem lietotājiem ir atvērta iespēja izveidot jaunus sistēmas kontus, izmantojot e-pasta adresi, nodrošinot drošu un vieglu dalībnieku ieviešanu. Lietotājiem ir iespēja droši un personificēti piekļūt savam kontam, izmantojot autorizācijas datus, piemēram, lietotājvārdu un paroli. Lietotāji var izveidot rezervācijas, norādot nepieciešamo informāciju par plānoto notikumu vai pakalpojumu. Lietotājiem ir piekļuve saviem rezervāciju datiem un iespēja tos apskatīt. Lietotāji var meklēt un filtrēt pieejamos kvestus, pielāgojot tos savām vēlmēm. Lietotāji var publicēt atsauksmes par veikto rezervāciju vai piedalīšanos kvestā.
- **Administratoru modulis.** Administratoram ir iespēja pievienot, dzēst vai mainīt kvestu istabas, piemēram, adrese, nosaukums, apraksts, atlaide utl..Administrators var dzēst citus lietotājus komentārus , ja nepieciešams, izvairīties no rupjas valodas, reklāmām, apvainojumiem.
- **Sistēmas modulis.** Platforma automātiski parādīs lietotāja izdarītās kļūdas, apkopo visus komentārus atsevišķā sadaļā, nodrošinās informāciju lietotājam, kā arī attēlos ,kuri ir redzāmi kvestu istabu parkatīšanas laikā un pasūtījumus CSV failā.

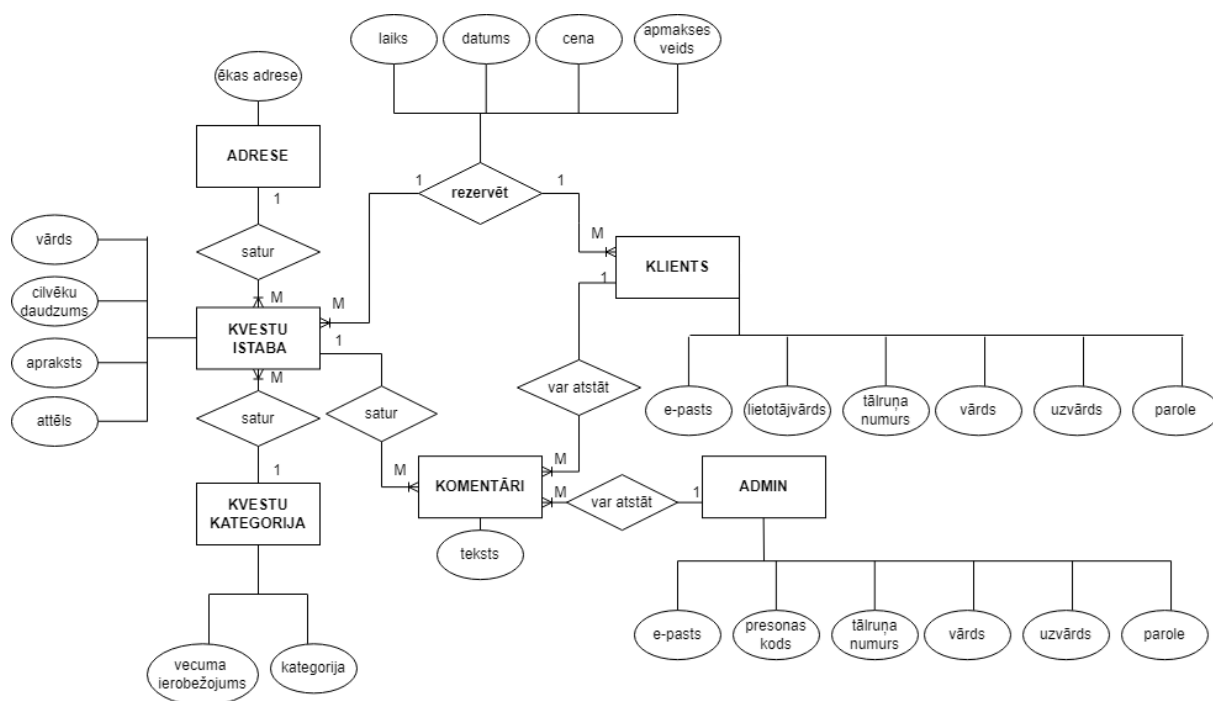
#### 4.1.2. Sistēmas ER modelis

Sistēmas ER-modelis sastāv no septiņām entitātēm (skat. 4.2. attēlu), kuras atspoguļo datu plūsmu sistēmā. Entitātes ir loģiskie objekti, kas pārstāv dažādus elementus un to īpašības, savukārt relācijas ir attiecības starp šīm entitātēm, kas nodrošina mijiedarbību un informācijas apmaiņu starp tām.

- "Admin" – šī entitāte apraksta administratorus, kuri ir atbildīgi par tīkla drošību, satura pārvaldību, atjaunināšanu un regulēšanu. Administratori nodrošina sistēmas nepārtrauktu un drošu darbību. Entitātes atribūtu kopums ietver šādus laukus: vārds, uzvārds, parole, e-pasts, telefona numurs un personas kods. Šie dati ir nepieciešami, lai nodrošinātu precīzu un drošu administratoru identificēšanu un autentifikāciju.
- "Klients" – šī entitāte raksturo lietotājus, kuri izmanto mūsu vietni, lai piekļūtu kvestu istabām un citām piedāvātajām pakalpojumām. Klientu atribūtu kopums ietver lietotāja vārdu, uzvārdu, paroli, e-pastu un telefona numuru. Šie dati ļauj sistēmai personalizēt lietotāja pieredzi un nodrošināt efektīvu komunikāciju ar klientiem.
- "Kvestu istaba" – šī entitāte sniedz informāciju par kvestu istabām, kuras tiek piedāvātas vietnē. Tajā iekļauti dati par istabu atrašanās vietu, žanru, un maksimālo un minimālo cilvēku skaitu, kas var piedalīties. Atribūtu kopums ietver nosaukumu, cilvēku skaitu,

aprakstu un attēla nosaukumu. Šī informācija palīdz lietotājiem izvēlēties piemērotāko kvestu istabu atbilstoši viņu interesēm un vajadzībām.

- "Rezervēt" – šī entitāte apraksta lietotāju veikto rezervāciju informāciju. Rezervācijas atribūtu kopums ietver laiku, datumu, atlaidi, apmaksas veidu un cenu. Šie dati ir būtiski, lai nodrošinātu efektīvu rezervācijas procesu un apstrādi, kā arī lai lietotāji varētu ērti plānot un pārvaldīt savas aktivitātes.
- "Komentāri" – šī entitāte raksturo lietotāju rakstītos komentārus par kvestu istabām un piedāvātajiem pakalpojumiem. Komentāru atribūtu kopums ietver tikai vienu lauku – tekstu, kas satur pašu komentāru. Šī informācija ir vērtīga, lai citi lietotāji varētu iepazīties ar atsauksmēm un veidot savu viedokli par piedāvātajām iespējām.
- "Adrese" – šī entitāte sniedz informāciju par ēkas adresi, kur atrodas kvestu istabas. Atribūtu kopums ietver ēkas adresi, kas ļauj lietotājiem viegli atrast un apmeklēt kvestu istabas. Precīza adrese ir svarīga, lai nodrošinātu klientu ērtības un precīzu navigāciju.
- "Kvestu kategorija" – šī entitāte apraksta katrai kvestu istabai piešķirtās kategorijas un vecuma ierobežojumus. Atribūtu kopums ietver vecuma ierobežojumu un kategoriju. Šī informācija palīdz lietotājiem izvēlēties kvestu istabas, kas atbilst viņu interesēm un vecuma grupai, nodrošinot drošu un atbilstošu pieredzi.



4.2.att. Sistēmas ER-diagramma

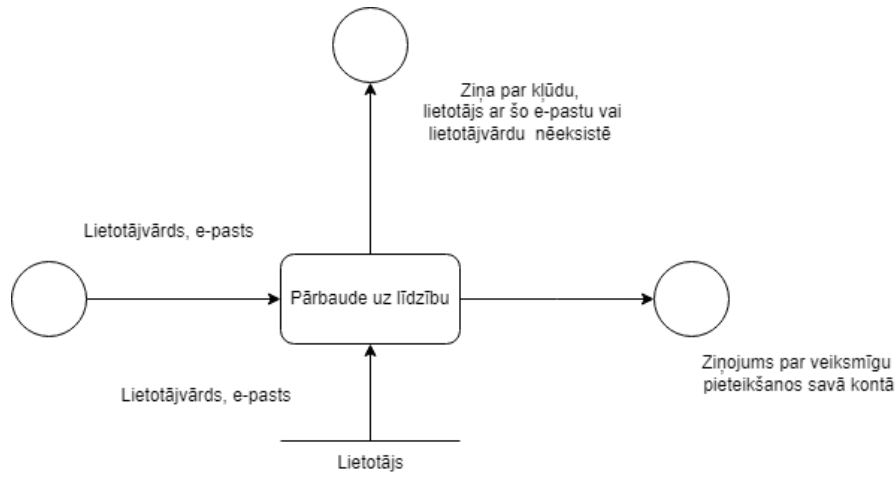
Starp šīm entītijām pastāv arī vairākas attiecības:

- **Admin (1) - (N) Komentārs.** Katram administratoram var būt vairāki komentāri, bet katrs komentārs ir saistīts tikai ar vienu administratoru. Šī attiecība nodrošina iespēju pievienot vairākus komentārus katram administratoram.
- **Lietotājs (Klients) (1) - (N) Rezervācija (Rezervēt).** Katrs lietotājs var izveidot vairākas rezervācijas, bet katrai rezervācijai ir tikai viens saistītais lietotājs. Šī attiecība norāda, ka lietotājs var veidot vairākās rezervācijas, bet katrai rezervācijai ir tikai viens lietotājs.
- **Lietotājs (Klients) (1) - (N) Komentārs.** Katrs lietotājs (klients) var būt vairāki komentāri (Komentārs), bet katrs komentārs ir saistīts tikai ar vienu lietotāju. Šī attiecība nodrošina iespēju lietotājam pievienot vairākus komentārus.
- **Kvestu istaba (1) - (N) Rezervācija (Rezervēt).** Katra kvestu istaba var būt saistīta ar vairākām rezervācijām, bet katra rezervācija ir saistīta tikai ar vienu konkrētu kvestu istabu. Šī attiecība nodrošina, ka viena kvestu istaba var tikt rezervēta daudzas reizes dažādos laikos un datumos, ļaujot dažādiem lietotājiem veikt rezervācijas uz vienu un to pašu kvestu istabu.
- **Kvestu istaba (1)– (N) Komentārs.** Katrai kvestu istabai var būt vairāki komentāri, bet katrs komentārs ir saistīts tikai ar vienu konkrētu kvestu istabu. Šī attiecība ļauj lietotājiem pievienot vairākus komentārus par vienu kvestu istabu, izsakot savu viedokli, pieredzi un atsauksmes par to.
- **Kvestu istaba (1) - (N) Kvestu kategorija.** Katra kvestu istaba var būt saistīta ar vairākām kvestu kategorijām, bet katra kvestu kategorija ir saistīta tikai ar vienu konkrētu kvestu istabu. Šī attiecība nodrošina, ka viena kvestu istaba var tikt klasificēta vairākās kategorijās, piemēram, pēc žanra un vecuma ierobežojuma.
- **Kvestu istaba (1) - (N) Adrese.** Katrai kvestu istabai var būt tikai viena adrese, bet viena adrese var būt saistīta ar vairākām kvestu istabām. Šī attiecība nozīmē, ka vienā un tajā pašā vietā (piemēram, vienā ēkā vai kompleksā) var atrasties vairākas kvestu istabas.

## 4.2. Funkcionālais sistēmas modelis

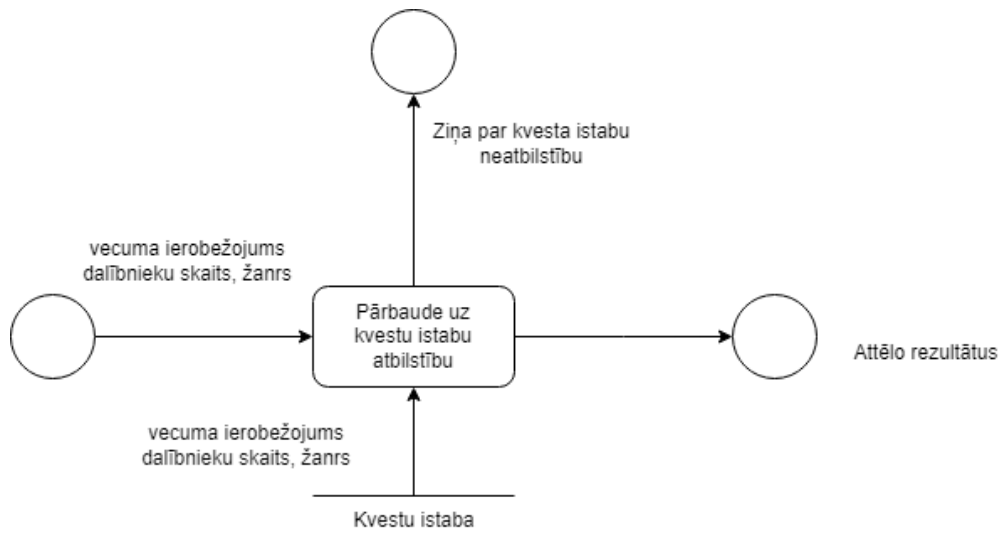
### 4.2.1. Datu plūsmu modelis

1. **Autorizācija.** Lai piekļūtu savam profilam, ir jāievada tādi dati kā e-pasta adrese un parole (skat. 4.3. att.). Ja dati ir ievadīti nepareizi, tiks parādīta kļūda par nepareizu datu ievadišanu. Ja dati ir pārbaudīti, lietotājs saņems paziņojumu, ka ir veiksmīgi pieteicies savā profilā.



4.3.att Autorizācijas datu plūsmu diagramma

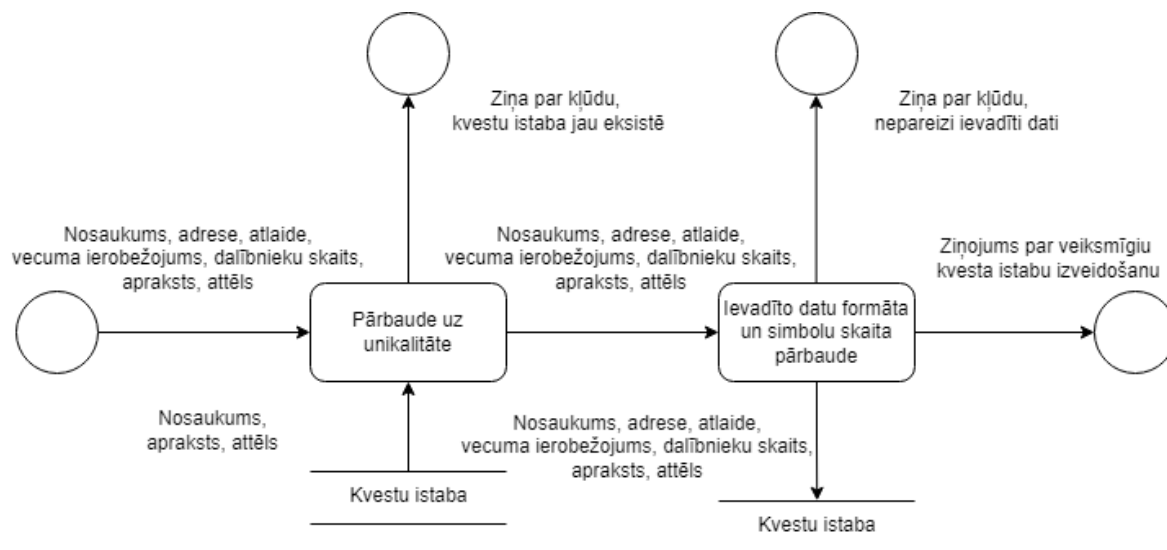
2. **Kvestu istabu meklēšana un filtrēšana.** Lietotājs ievada tādus datus kā vecuma ierobežojums, dalībnieku skaits un žanrs (skat. 4.4. att.). Ja atbilstoši ievadītajiem kritērijiem nekas nav atrasts, tiks parādīts paziņojums, ka nav atbilstības. Ja atbilstības ir, tās tiks parādītas lietotāja ekrānā.



4.4.att Kvestu istabu meklēšanas un filtrēšanas datu plūsmu diagramma

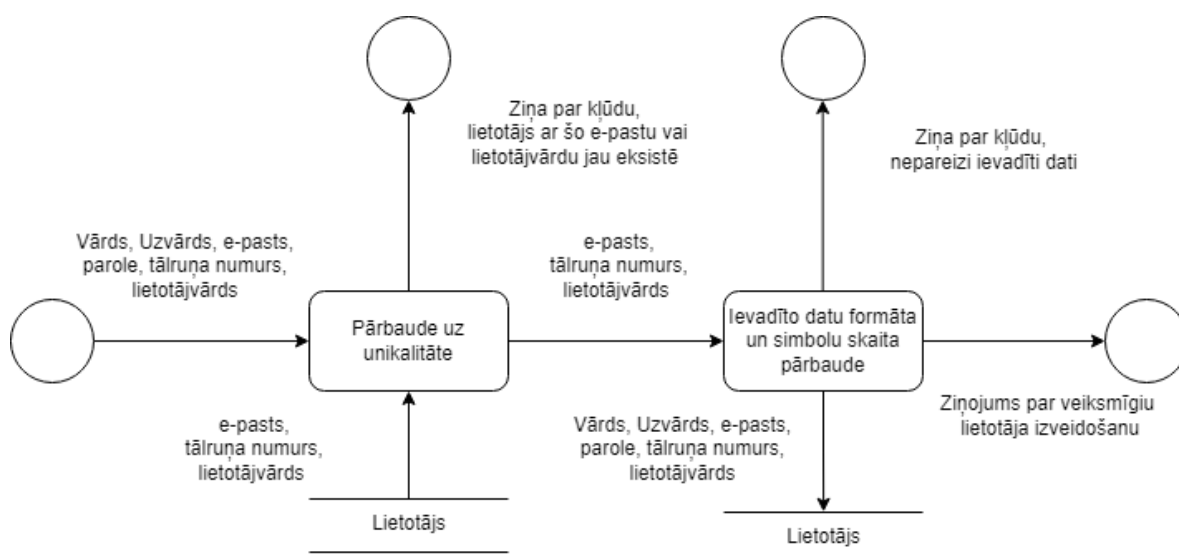
3. **Kvestu pievienošana.** Administrators, veidojot kvestu istabu, veidlapā ieraksta šādus datus (skat. 4.5. att.): nosaukums, adrese, atlaide, spēlētāju skaita

organizēšana, vecuma organizēšana, apraksts un attēls. Ja kāds no priekšmetiem nav izturējis pārbaudi, tiks parādīts paziņojums par konkrētā priekšmeta kļūdu. Ja visi elementi ir pārbaudīti, tiks parādīts paziņojums par sekmīgu kvestu istabas izveidi.



4.5.att Kvestu pievienošana datu plūsmu diagramma

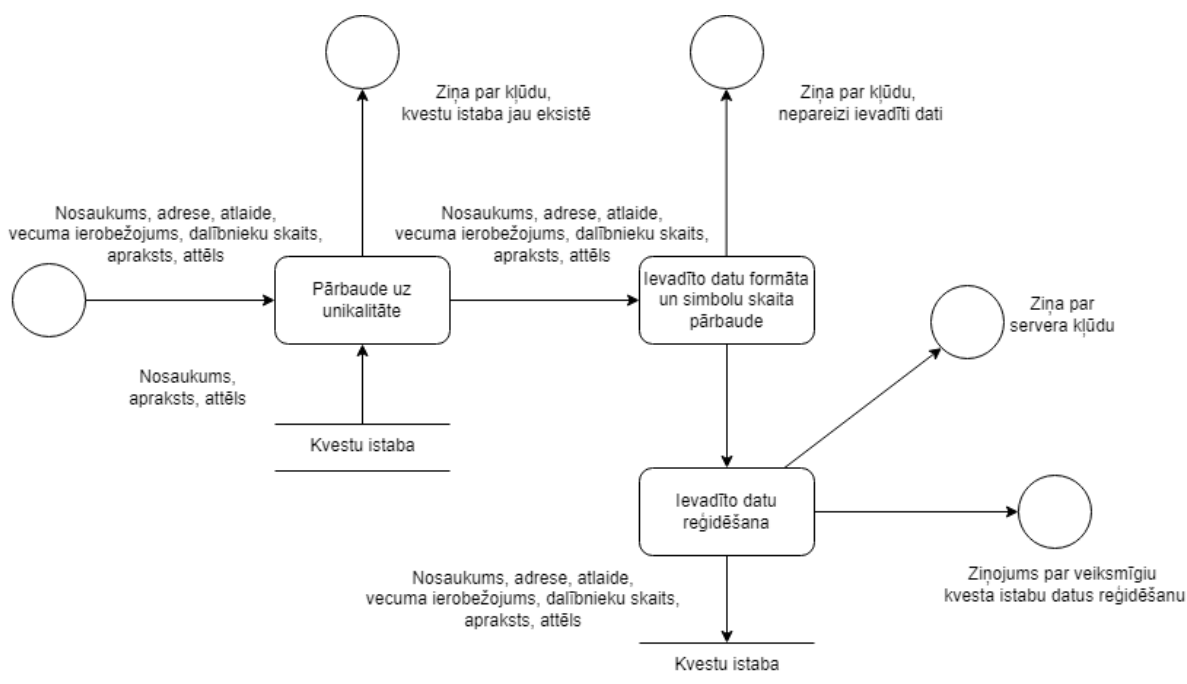
- 4. Reģistrācija.** Reģistrējoties lietotājs ievada šādus datus(skāt. 4.6. att.): vārds, uzvārds, lietotājvārds, e-pasts, parole, tālruņa numurs. Pēc tam tiks pārbaudīti uz unikālītāti, uz datu formāta un simbolu skaitu. Ja visas pārbaudes ir izturētas, profils tiks izveidots un tiks parādīts paziņojums par veiksmīgu profila izveidi.



4.6.att Reģistrācijas datu plūsmu diagramma

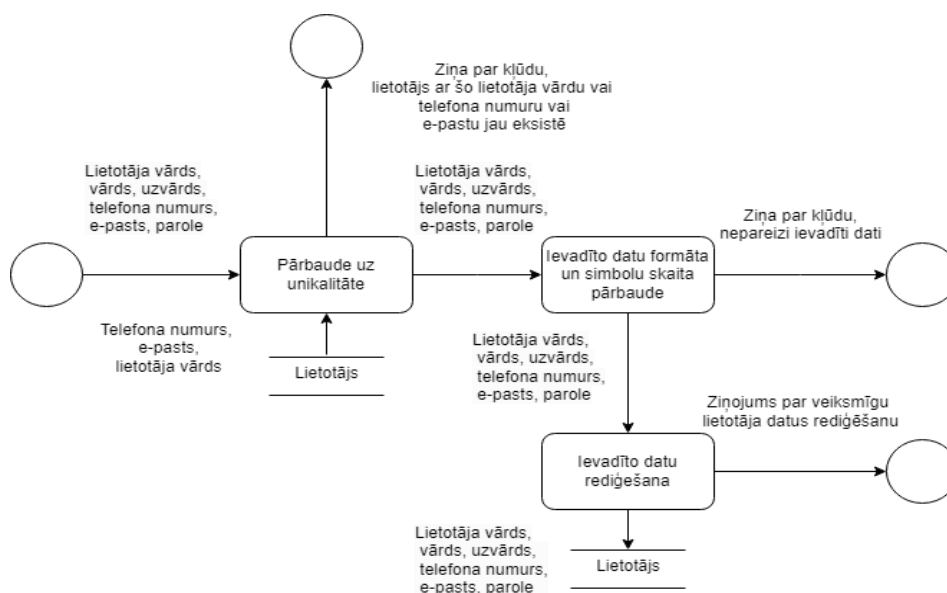
- 5. Kvestu rediģēšana.** Mainot datus, parādīsies logi ar aizpildītu informāciju (skāt. 4.7. att.), kas jau ir šajā kvesta telpā, mainot datus, jums ir jāizdzēs pašreizējie dati un jāpārraksta uz jums nepieciešamo, piemēram (adrese Stirnu iela 10 ,

Vaidavas iela 5). Pēc tam tiks pārbaudīta uz unikālītati, uz datu formāta un simbolu skaitu. Pēc pārbaudes izturēšanas jums tiks paziņots par veiksmīgu datu maiņu.



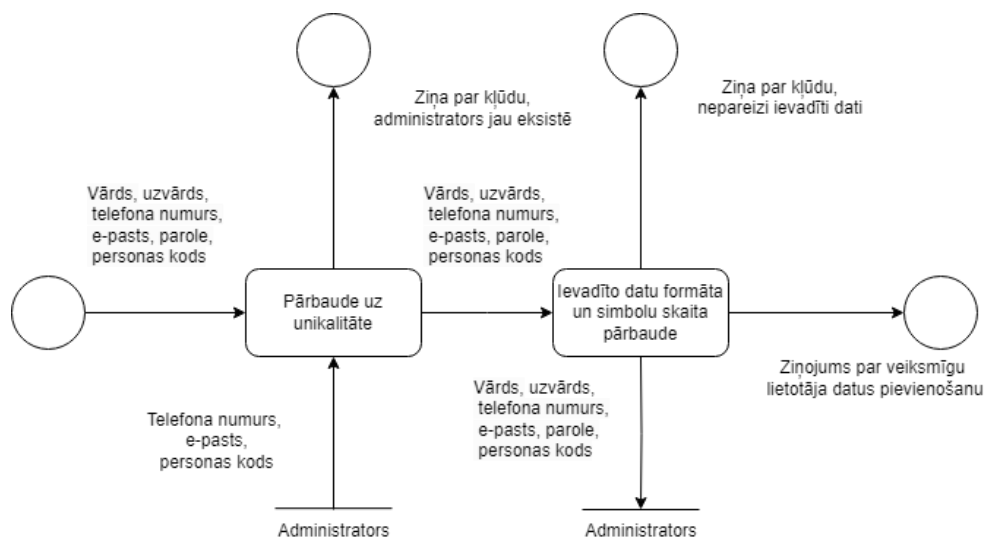
4.7.att Kvestu rediģēšana datu plūsmu diagramma

**6. Lietotāju rediģēšana.** Mainot datus, parādīsies logi ar aizpildītu informāciju par lietotāju (sk. 4.8. att.), ja maināt datus, ir nepieciešams izdzēst pašreizējos datus un pārrakstīt tos ar nepieciešamajiem datiem, piemēram, (pasts edgars123@gmail.com, edgars.berzins05@gmail.com). Pēc tam tiks pārbaudīta datu unikālītāte, datu formāts un zīmju skaits. Pēc pārbaudes izturēšanas tiks saņemts paziņojums, ka dati ir veiksmīgi mainīti.



4.8.att Lietotāju rediģēšana datu plūsmu diagramma

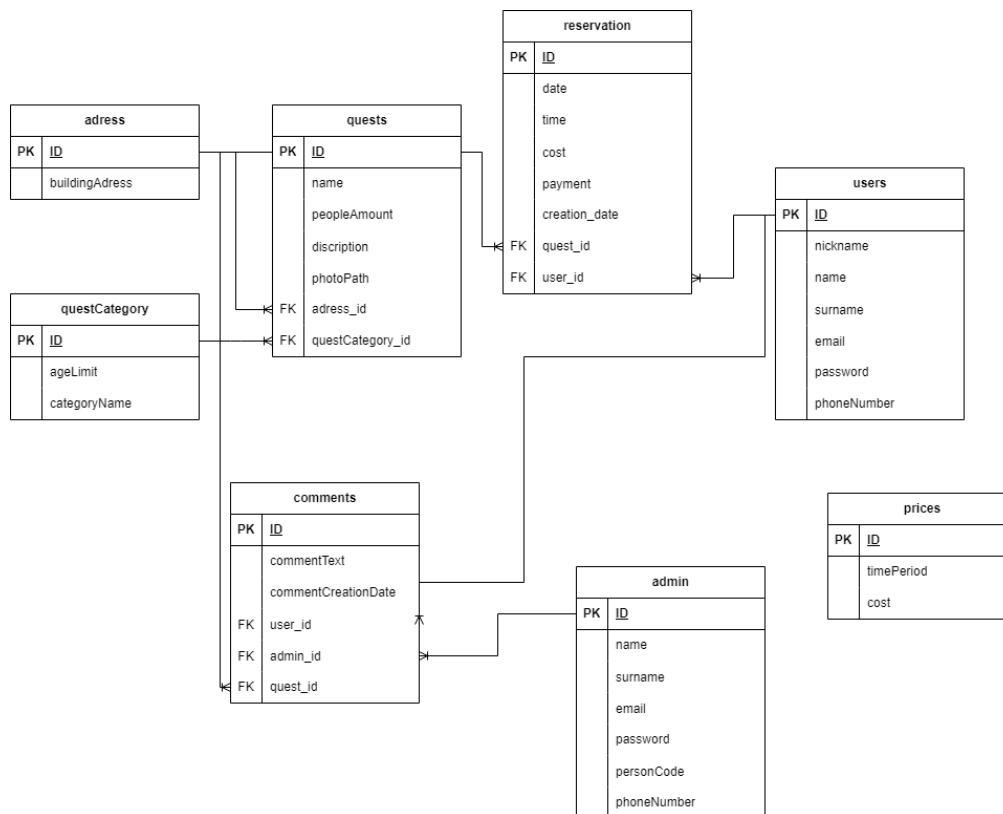
**7. Administratoru pievienošana.** Administrators, pievienojot administratoru, veidlapā ieraksta šādus datus (skat. 4.9. att.): vārds, uzvārds, telefona numurs, parole, e-pasts un personas kods. Ja kāds no priekšmetiem nav izturējis pārbaudi, tiks parādīts paziņojums par konkrētā priekšmeta kļūdu. Ja visi elementi ir pārbaudīti, tiks parādīts paziņojums par sekmīgu kvestu istabas izveidi.



4.9.att Lietotāju rediģēšana datu plūsmu diagramma

## 5. DATU STRUKTŪRU APRAKSTS

Datu bāze sastāv no 8 tabulām, kas satur sevī informāciju par lietotāju, rezervācijām, administratoriem, komentāriem un kvestu istabām. (sk. 5.1. att.) ir pielikta tabulu relāciju shēma.



5.1.att Datubāzes fiziskās struktūras shēma

1. Tabula “**admin**” glabā datus par administratoriem.
2. Tabula “**comment**” glabā datus par komentāriem.
3. Tabula “**quests**” glabā datus par kvestu istābām.
4. Tabula “**reservation**” glabā datus par rezervācijām.
5. Tabula “**users**” glabā datus par lietotājiem.
6. Tabula “**adress**” glabā datus par kvestu istabu adresi.
7. Tabula “**questCategory**” glabā datus par kvesta istabu kategorijus.
8. Tabula “**price**” glabā datus par rezervācijas laiku un cenu.



Tabula 'admin' ir saistīta ar tabulu "comment".

5.1.tabula

Tabulas "admin" struktūra

Nr.	Nosaukums	Tips	Garums	Piezīme
1	ID	Int	11	Administrators identifikācijas numurs, primārā atslēga
2	name	Varchar	128	Administrators vārds
3	surname	Varchar	128	Administrators uzvārds
4	password	Varchar	128	Administrators parole
5	email	Varchar	128	Administrators e-pasts
6	phoneNumer	Varchar	128	Administrators teledona numurs
7	personCode	Char	12	Administrators persoans kods

Tabula 'comment' ir saistīta ar tabulu "quests", "admin" un "users".

5.2.tabula

Tabulas "comment" struktūra

Nr.	Nosaukums	Tips	Garums	Piezīme
1	ID	Int	11	Komentāra identifikācijas numurs, primārā atslēga
2	text	text	-	Klienta vai administratora teksts
3	Client_ID	Int	11	Ārēja atslēga, kas norāda klientu, var būt nulle*
4	Admin_ID	Int	11	Ārēja atslēga, kas norāda klientu administratoru, var būt nulle**
5	reply_to	int	11	Ārēja atslēga, kas norāda uz kuru komentāru tiek uzrakstīta atbilde
6	creation_date	date	10	Datums kad komentārs tiek izveidots

\*Ja komentāru uzraksta administrators

\*\*Ja komentāru uzraksta klients

Tabula 'quests' ir saistīta ar tabulu "comment", "reservation", "adress" un "quest category".

5.3.tabula

Tabulas "quests" struktūra

Nr.	Nosaukums	Tips	Garums	Piezīme
1	ID	Int	11	Kvestu istaba identifikācijas numurs, primārā atslēga
2	name	Varchar	128	Kvestu istaba nosaukums
3	peopleAmount	Varchar	5	Kvestu istaba piedāvāts cilvēku skaits
4	description	text	-	Kvestu istaba apraksts
5	photoPath	Varchar	128	Kvestu istaba attēlas cēļš

6	adress_id	Int	11	Ārēja atslēga,kas norada adresi
7	questCategory_id	int	11	Ārēja atslēga,kas norada kategoriju

Tabula ‘reservation’ ir saistīta ar tabulu “quests” un “users”.

5.4.tabula

Tabulas “**reservation**” struktūra

Nr.	Nosaukums	Tips	Garums	Piezīme
1	ID	Int	11	Kvestu istaba identifikācijas numurs, primārā atslēga
2	date	Date	-	Rezervācijas datums
3	time	Varchar	5	Rezervācijas laiks
4	cost	Double	(4,2)	Rezervācijas cena
5	payment	Varchar	128	Rezervācijas apmaksas veids
6	room_ID	Int	11	Ārēja atslēga,kas norada kvestu istabu
7	client_ID	Int	11	Ārēja atslēga,kas norada klientu
8	Creation_date	date	-	Rezervācijas izveidošanas datums

Tabula ‘users’ ir saistīta ar tabulu “reservation” un “comment”.

5.5.tabula

Tabulas “**users**” struktūra

Nr.	Nosaukums	Tips	Garums	Piezīme
1	ID	Int	11	Klienta identifikācijas numurs, primārā atslēga;
2	nickname	Varchar	128	Klienta lietotājvārds;
3	name	Varchar	128	Klienta vārds;
4	surname	Varchar	128	Klienta uzvārds;
5	password	Varchar	128	Klienta parole;
6	email	Varchar	128	Klienta e-pasts;
7	phoneNumber	Int	30	Klienta teledona numurs;

Tabula ‘prices’ nav saistīta ar nevienu tabulu.

5.6.tabula

Tabulas “**prices**” struktūra

Nr.	Nosaukums	Tips	Garums	Piezīme
1	ID	Int	11	Cenas identifikācijas numurs, primārā atslēga;
2	timePeriod	Varchar	128	Rezervācijas laiks ;

3	cost	double	4,2	Rezervācijas cena;
---	------	--------	-----	--------------------

Tabula 'quest category' ir saistīta ar tabulu "quests"

5.7.tabula

Tabulas "quest category" struktūra

Nr.	Nosaukums	Tips	Garums	Piezīme
1	ID	Int	11	Kategorijas identifikācijas numurs, primārā atslēga;
2	ageLimit	int	2	Kvestu istaba vecuma ierobežojums;
3	categoryName	Varchar	128	Kvestu istaba kategorija;

Tabula 'adress' ir saistīta ar tabulu "quests"

5.8.tabula

Tabulas "adress" struktūra

Nr.	Nosaukums	Tips	Garums	Piezīme
1	ID	Int	11	Kategorijas identifikācijas numurs, primārā atslēga;
2	ageLimit	int	2	Kvestu istaba vecuma ierobežojums;
3	categoryName	Varchar	128	Kvestu istaba kategorija;

Fiziskajā līmenī dati ir organizēti un glabājas, izmantojot tabulu saišu shēmu (skat. 5. attēlu). Šī struktūra nodrošina efektīvu datu glabāšanu un atgriešanu, nodrošinot labu veiktspēju un optimizētu datu piekļuvi.

## **6. LIETOTĀJA CEĻVEDIS**

### **6.1.Sistēmas struktūras modelis**

Sistēma tika realizēta kā interneta tīmekļa vietne, kuras lietošana neprasa specifisku programmu instalāciju, kā arī var tikt palaista no jebkuras ierīces, kurai ir:

- viena no jaunākajām pārlūkprogrammu versijām;
  - Safari – 16. versija un jaunāk,
  - Windows Edge – 123.0.2420.74 un jaunāk,
  - Google Chrome – 125.0.6422.112 un jaunāk,
  - Opera – 110.0.5130.39 un jaunāk,
  - Mozilla Firefox – 126.0 un jaunāk;
- Stabils interneta savienojums, jo labāks savienojums, jo veiksmīgāk nortēs programmas darbība;
- Aparatūra, lai uz tās varētu darbināt iepriekš šajā sadaļā minētās pārlūkprogrammas.

### **6.2.Sistēmas instalācija un palaišana**

Lai uzstādītu un palaistu sistēmu, veiciet šādas darbības:

1. Lejupielādējiet un instalējiet XAMPP.
  - Apmeklējiet XAMPP oficiālo mājaslapu un lejupielādējiet jaunāko versiju, kas atbilst jūsu operētājsistēmai.
  - Instalējiet XAMPP, izpildot instalēšanas vedni.
2. Projekta lejupielāde no GitHub.
  - Apmeklējiet manu GitHub repozitoriju un lejupielādējiet projekta failus (<https://github.com/KIvanovs/DBQuestRoom.git>).
  - Izkopējiet lejupielādētos failus uz xampp/htdocs mapi (C:\xampp\htdocs).
3. XAMPP servera palaišana.
  - Atveriet XAMPP Control Panel.
  - Palaidiet Apache un MySQL serverus, nospiežot "Start" pogu pie katra no tiem.
4. Datubāzes instalēšana.
  - Atveriet interneta pārlūku.
  - Adrešu joslā ievadiet localhost un nospiediet Enter taustiņu.

- Atveriet “phpMyAdmin”.
- Atveriet “Import” sadaļu .
- Noklikšķiniet uz “Choose the file” un izvēlieties failu, kas atrodas xampp/htdocs/ DBQuestRoom/testdb.php , un noklikšķiniet uz “Import”.

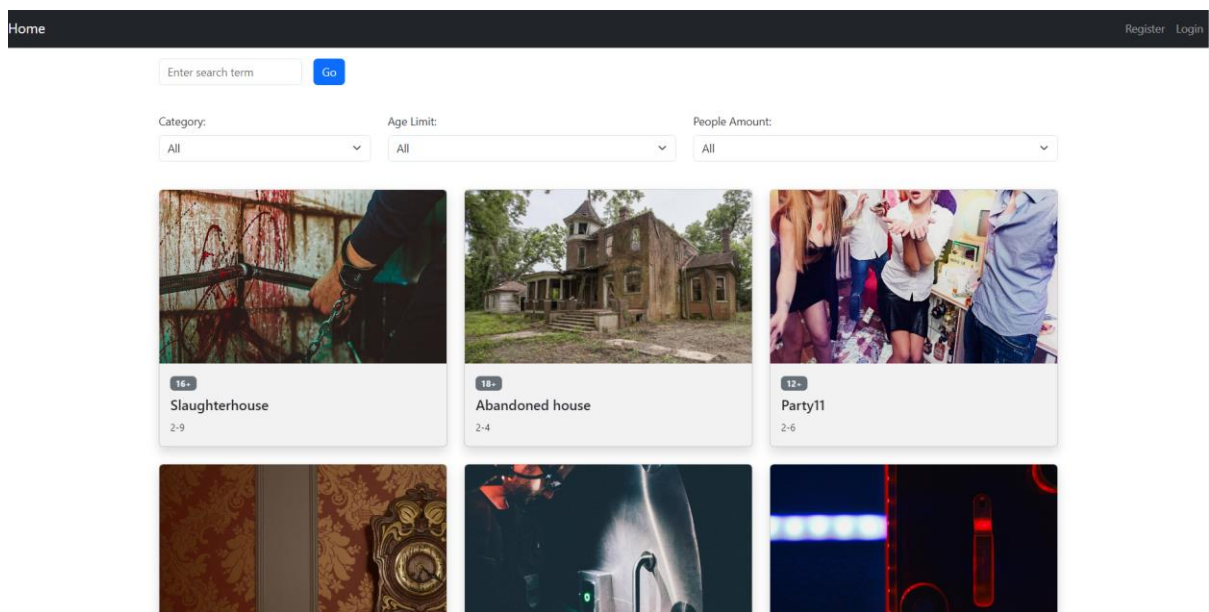
#### 5. Sistēmas palaišana pārlūkā.

- Atveriet interneta pārlūku.
- Adrešu joslā ievadiet <https://site.local/DBquestroom> un nospiediet Enter taustiņu.

## 6.3.Programmas apraksts

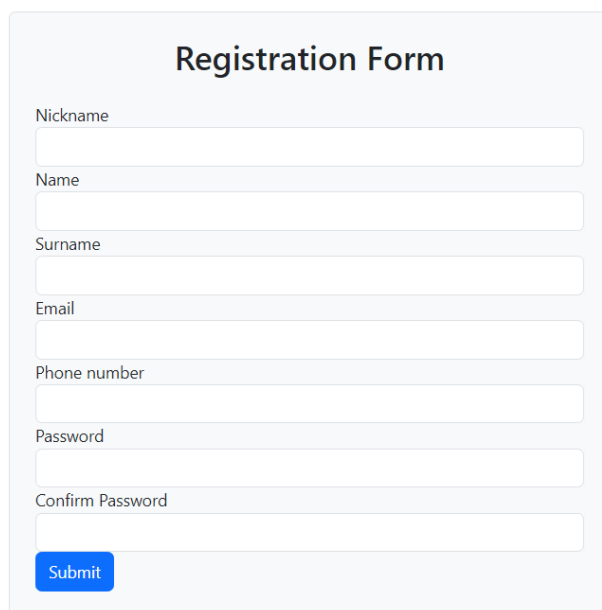
### Lietotāju reģistrācija un autorizācija

Lai reģistrētos vietnē, sākumlapā navigācijas joslā noklikšķiniet uz “Register” un priekš autorizācijas uz pogu “Login”(sk. 6.1. att.).



6.1.att Sākuma lapas skats

Lai reģistrētos vietnē, ievadiet datus vajadzīgajos laukos un nospiediet pogu “Submit” (sk. 6.2. att.).



**Registration Form**

Nickname

Name

Surname

Email

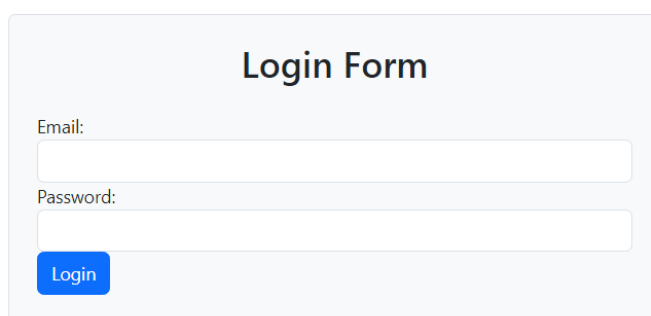
Phone number

Password

Confirm Password

6.2.att Reģistrācijas forma

Lai pieteiktos, ievadiet datus vajadzīgajos laukos un nospiediet pogu “Login” (sk. 6.3. att.).



**Login Form**

Email:

Password:

6.3.att Autorizācijas forma

### **Kvesta istabu meklēšana un rezervēšana**


Sākumlapā lietotājs var ievadīt jebkuru meklēšanas terminu. Pēc ievadīšanas jāklikšķina uz pogas "Go", lai aktivizētu meklēšanu. Kā arī lietotājs var izmantot kategorijas, vecuma ierobežojuma un cilvēku skaita nolaižamās izvēlnes, lai sašaurinātu meklēšanas rezultātus (sk. 6.4. att.).

Home
Register
Login

House
Go

Category:
Age Limit:
People Amount:

Horror
16+
2-9



16+
Slaughterhouse
2-9

6.4.att Rezultāts pēc meklēšanas un filtrēšanas

Noklikšķinot uz izvēlētās uzdevumu telpas kartes, tiks atvērta lapa ar detalizētu informāciju par uzdevumu telpu. Šajā lapā (skatīt 6.5. attēlu) izvēlamies datumu, noklikšķinot uz datuma kalendārā un izvēloties vienu no piedāvātajiem laikiem. Pēc tam izvēlamies maksājuma veidu, ja izvēlamies maksāt ar karti, ir jāievada kartes dati. Pēc datu aizpildīšanas nospiediet pogu "Rezervēt".

## Reservation

Date:

June 2024
Previous month
Next month

Mo	Tu	We	Th	Fr	Sa	Su
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Time:

10:00 (60.00 EUR)
11:30 (60.00 EUR)
13:30 (60.00 EUR)
14:30 (60.00 EUR)
16:00 (60.00 EUR)
17:30 (60.00 EUR)
19:00 (60.00 EUR)
20:30 (80.00 EUR)
22:00 (80.00 EUR)

## Payment

You have selected:

Total price:

Payment method:

Card

Номер карты
MM / TT CVC

Reserve

6.5.att Rezervācijas forma

Pēc rezervējuma izveides lietotājs saņem e-pasta paziņojumu, ka rezervējums ir izveidots. E-pastā būs informācija par meklēšanas telpu, datumiem, laikiem un kvīts pdf formātā (sk. 6.6.att.).

Quest Room Reservation

Invoice #: 66648832bcc49

Created: 2024-06-08

Due: 2024-07-08

Kirill Quest Room

1234 Main St

Anytown, CA 12345

testpochta testpochta

ekspunser@gmail.com

Payment Method

cash

Payment Method

cash

Item

Price

Quest Room Reservation - House 12

\$60

Date and Time

2024-06-28 at 19:00

Location

Krasta iela 32

Total: \$60

6.6.att Rezervācijas kvīts

Komentāru pievienošana, dzēšana un rediģēšana.

Tajā pašā vietā, kur mēs izveidojam rezervāciju uz kvestu istabu, ritinot uz lapas apakšā, jūs varat uzrakstīt komentāru un publicēt to, noklikšķinot uz pogas "Submit". Jūs varat atbildēt uz komentāru, noklikšķinot uz pogas "Reply". Noklikšķinot uz pogas "Replies", jūs varat redzēt, kādas atbildes tika sniegtas uz konkrēto komentāru. Ja esat komentāra autors, varat arī dzēst savu komentāru, noklikšķinot uz pogas "Delete" un varat mainīt savu komentāru, noklikšķinot uz pogas "Update" (sk. 6.7. att.).

Leave a Comment

Enter your comment

Enter your comment

Submit

@QuestHunter34

2024-05-25

Ļoti patika vai Jūs varat ieteikt tāda paša žanra kvestu istabu?

Update

Delete

Reply

Replies

@ErnestsBOY

2024-05-25

iesaku the bank job

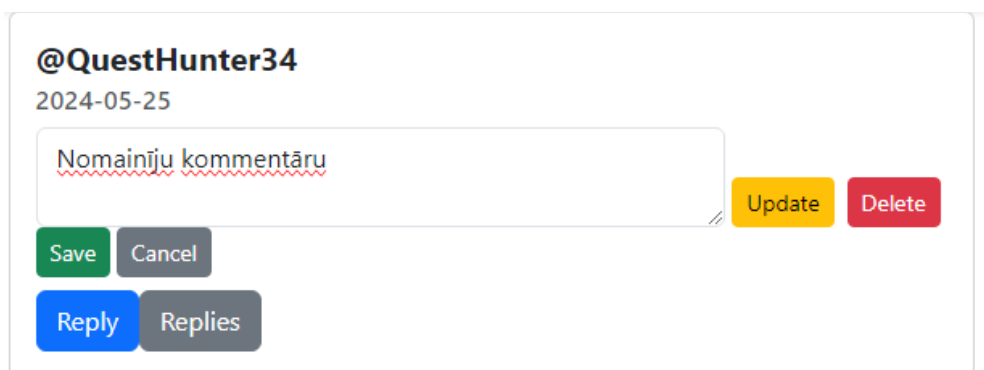
Reply

Replies

6.7.att Komentāru forma



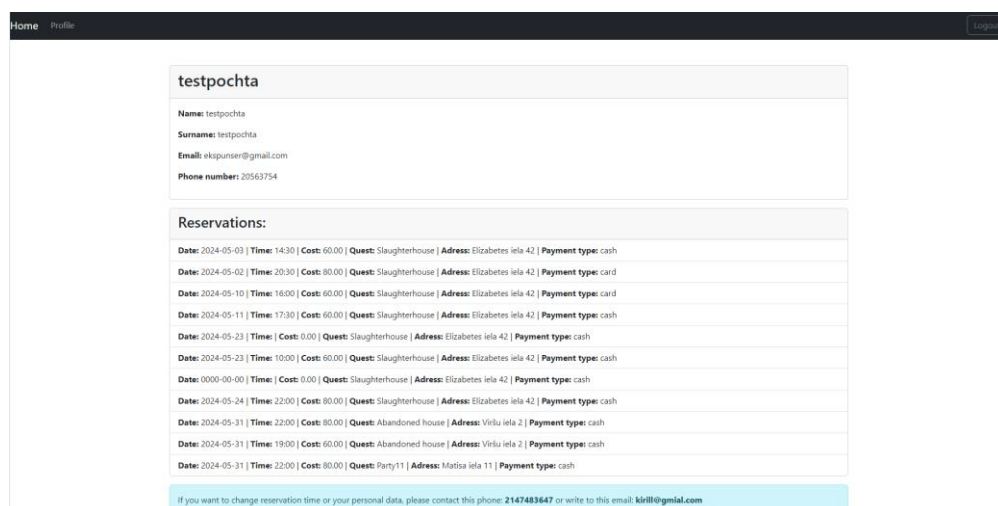
Noklikšķinot uz pogas “Atjaunināt”, jūsu komentārs kļūst pārrakstāms, un jūs varat to pārrakstīt un saglabāt izmaiņas, noklikšķinot uz pogas “Save”, bet, ja pārdomājat, varat noklikšķināt uz pogas “Cancel” (sk. 6.8. att.).



6.8.att Komentāru rediģēšana forma

## Profila apskate

Ja esat autorizējies vietnē, navigācijas joslā pa labi no pogas “Home” būs poga “Profile”, kuru noklikšķinot tiks atvērta jūsu profila lapa ar informāciju par jums un jūsu rezervācijām, ko jebkad esat izdarījis (sk. 6.9. att.).



Date	Time	Cost	Quest	Address	Payment type
2024-05-03	14:30	60.00	Slaughterhouse	Elizabetes iela 42	cash
2024-05-02	20:30	80.00	Slaughterhouse	Elizabetes iela 42	card
2024-05-10	16:00	60.00	Slaughterhouse	Elizabetes iela 42	card
2024-05-11	17:30	60.00	Slaughterhouse	Elizabetes iela 42	cash
2024-05-23	0:00	0.00	Slaughterhouse	Elizabetes iela 42	cash
2024-05-23	10:00	60.00	Slaughterhouse	Elizabetes iela 42	cash
0000-00-00	0:00	0.00	Slaughterhouse	Elizabetes iela 42	cash
2024-05-24	22:00	80.00	Slaughterhouse	Elizabetes iela 42	cash
2024-05-31	22:00	80.00	Abandoned house	Viršu iela 2	cash
2024-05-31	19:00	60.00	Abandoned house	Viršu iela 2	cash
2024-05-31	22:00	80.00	Party11	Matīsa iela 11	cash

If you want to change reservation time or your personal data, please contact this phone: 2147483647 or write to this email: kirill@gmail.com

6.9.att Lietotāja profila lapa

## Administratora jeb darbinieka pievienošana

Ja esat ienācis sistēmā kā administrators, navigācijas joslā ir poga “Admin/Employee info”, uz kuras noklikšķinot, jūs tiekat novirzīts uz lapu, kurā varat pievienot jaunu darbinieku vietnes sistēmai, aizpildot nepieciešamos datus nepieciešamajos laukos, nospiežot pogu “Add” (sk 6.10. att.).

## Add Employee

Name:

Surname:

Email:

Password:

Personal Code:

Phone Number:

Add

6.10.att Administratora jeb darbinieka pievienošanas forma

Administratora pievienošanas veidlapā ir tabula ar informāciju par administratoru. Administrators var mainīt darbinieka datus, nospiežot pogu "Update" un var arī dzēst darbinieku, nospiežot pogu "Delete" (sk. 6.11. att.).

### Employee List

ID	Name	Surname	Email	Personal Code	Phone Number	Actions
1	Kirill	Ivanovs	kirill@gmial.com	200303-20102	2147483647	
4	nameasd	surname	asd@asd.zxc	989878-98043	25856312	Delete Update
5	asd	asd	123@gmail.com	132123-12316	123123123	Delete Update
7	Edgars	Pavlovs	asd@asd.asd	684965-39865	596874657	Delete Update
8	dfgh	dfgh	ponchik@gmail.com	200303-20134	2147483647	Delete Update
9	admin	admin	email@gmail.com	123543-34576	7654567	Delete Update
10	edgar	kozlovs	kozlovs@gmai.com	765098-56784	848356	Delete Update
19	testtasdasdssss	testtsss	test@gmial.com	909090-09909	9090999	Delete Update

6.11.att Informācija par administratoriem

Noklikšķinot uz pogas "Update", varat mainīt vajadzīgos datus, un, lai saglabātu izmaiņas, noklikšķiniet uz pogas "Update" (sk. 6.12. att.).

6.12.att Informācija par administratoriem




## Kvestu istabu pievienošana

Ja esat ienācis sistēmā kā administrators, navigācijas joslā ir poga “Quest room info”, uz kuras noklikšķinot, jūs tiekat novirzīts uz lapu, kurā varat pievienot jaunu kvestu istabu, aizpildot nepieciešamos datus nepieciešamajos laukos, nospiežot pogu “Add Quest” (sk 6.13. att.).

6.13.att Kvestu istabu pievienošanas forma

Kvestu istabu pievienošanas veidlapā ir tabula ar informāciju par kvestu istabu. Administrators var mainīt kvesta istabas datus, nospiežot pogu "Update" un var arī dzēst kvestu istabu, nospiežot pogu "Delete" (sk 6.14. att.).

## Quest List

ID	Name	Category	Address	People Amount	Age Limit	Description	Photo Path	Photo	Update	Delete
3	Slaughterhouse	Horror	Elizabetes iela 42	2-9	16	You woke up in the slaughterhouse maniac and you realize that it has become prey. He fastened the handcuffs you around solid blood, and heard the cries of the victims in the distance. After a while, after hitting an ax, shouting disappear ... Do you realize that you do not want that would you all so over and you at any cost necessary to vybratsya. Think of how to do it, otherwise your fate will be the same.	../images/e026b59f-b96f-48e2-b4f3-f21d38a7f8a1.jpg.1200x500_q85_crop-smart_upscale.jpg		<a href="#">Update</a>	<a href="#">Delete</a>
4	Abandoned house	Detective	Viršu iela 2	2-4	18	The horror hidden in the ancient house breaks out, enveloping them with invisible bonds and dragging them into the labyrinths of an endless nightmare.	../images/99a6cf68-1823-4a48-83cb-b18f188b77d8-spooky-homes-around-the-world-moulthrop-house.jpg		<a href="#">Update</a>	<a href="#">Delete</a>
5	Party11	For beginners	Matisa iela 11	2-6	12	Yesterday was great, but not today. You woke up after office corporate party in underwear. Trying to build a head last night and found the annual report, which now need to pass. The chief had already left to you and will be there within 60 minutes.	../images/6652340c78f7b.jpg		<a href="#">Update</a>	<a href="#">Delete</a>

### 6.14.att Informācija par administratoriem jeb darbiniekiem

Noklikšķinot uz pogas “Update”, tiks atvērta jauna lapa ar informāciju par meklējumu telpu, kuru vēlaties mainīt. Pēc tam, kad esat mainījis informāciju, kuru vēlaties saglabāt, noklikšķiniet uz pogas “Update”, lai to saglabātu (sk 6.15. att.).

#### Edit Room

Name:


Category:

Address:

People Amount:

Age Limit:

Description:

Current photo:  


New Photo:  No file chosen

### 6.15.att Kvestu istabu rediģēšanas forma

#### Lietotāja datu apskatīšana, dzēšana un modificēšana

Ja esat ienācis sistēmā kā administrators, navigācijas joslā ir poga “User’s info”, uz kuras noklikšķinot, jūs tiek novirzīts uz lapu, kurā varat apskatīt informāciju par lietotāju. Administrators var mainīt lietotāja datus, nospiežot pogu "Update" un var arī dzēst lietotāju, nospiežot pogu "Delete" (sk. 6.16. att.).

<a href="#">Home</a> <a href="#">Admin/Employee info</a> <a href="#">Quest room info</a> <a href="#">User's info</a> <a href="#">Charts</a> <span>Logout</span>							
User Profiles							
ID	Nickname	Name	Surname	Email	Phone Number	Reservations	Actions
10	poncik	Kirils	Ivanovs	poncik@gmail.com	22222222	User has 1 reservations	<a href="#">Delete</a> <a href="#">Update</a>
12	Misha35	Misha	Pavlov	misha@gmail.com	20202123	No reservations found for this user.	<a href="#">Delete</a> <a href="#">Update</a>
13	Bublik	Anton	Pavlov	pavlov@gmail.com	20405735	User has 3 reservations	<a href="#">Delete</a> <a href="#">Update</a>
15	QuestBoy32	asdfsdf	asdfsdf	ponchihik@gmail.com	34563456	No reservations found for this user.	<a href="#">Delete</a> <a href="#">Update</a>
16	Dmitry Cool	Dmitry	Sminov	dmitry.sminov@gmail.com	689678967	No reservations found for this user.	<a href="#">Delete</a> <a href="#">Update</a>
17	Batep	Andrej	Ivanov	pochta@gmail.com	23457456	No reservations found for this user.	<a href="#">Delete</a> <a href="#">Update</a>
18	Deniss25	Deniss	Kozlov	deniss25@gmail.com	20950694	User has 1 reservations	<a href="#">Delete</a> <a href="#">Update</a>
19	Kirils12	Kirils	Ivanovs	kirils12@gmail.com	204050645	User has 1 reservations	<a href="#">Delete</a> <a href="#">Update</a>
21	asd	asd	asd	asd@gmail.com	2147483647	No reservations found for this user.	<a href="#">Delete</a> <a href="#">Update</a>
23	asdasd	asdasd	asdasd	asdasd@gmail.com	2147483647	User has 23 reservations	<a href="#">Delete</a> <a href="#">Update</a>
24	user	user	user	useruser@user.com	1231233213	User has 3 reservations	<a href="#">Delete</a> <a href="#">Update</a>
25	reply	reply	reply	manapiw843@dxice.com	23423424	No reservations found for this user.	<a href="#">Delete</a> <a href="#">Update</a>
31	testpochta	testpochta	testpochta	ekspunser@gmail.com	20563754	User has 11 reservations	<a href="#">Delete</a> <a href="#">Update</a>

6.16.att Informācija lietotājiem

Noklikšķinot uz pogas “Update”, tiks atvērta jauna lapa ar informāciju par lietotāju, kura informāciju vēlaties mainīt. Pēc tam, kad esat mainījis informāciju, kuru vēlaties saglabāt, noklikšķiniet uz pogas “Update”, lai to saglabātu (sk. 6.17. att.).

---

Nickname:

Name:

Surname:

Email:

Password:

Phone Number:

[Update](#)

6.17.att Lietotāju rediģēšana forma

Arī tajā pašā lapā rezervējumu slejā varat detalizēti apskatīt lietotāja veiktos rezervējumus, noklikšķinot uz bultiņas. Noklikšķinot uz tās, parādās informācija par rezervācijām, kuras administrators vai darbinieks var dzēst vai mainīt (sk. 6.18. att.).

Reservations

User has 1 reservations

No reservations found for this user.

User has 3 reservations

Reservation ID: 55, Date: 2023-04-27, Time: 13:30 Room ID: 5

Delete

Update

Reservation ID: 56, Date: 2023-04-27, Time: 13:30 Room ID: 6

Delete

Update

Reservation ID: 57, Date: 2023-05-19, Time: 10:00 Room ID: 6

Delete

Update

6.18.att Sīkāka informācija par lietotāju rezervācijām

Noklikšķinot uz pogas “Atjaunināt”, tiks atvērta jauna lapa ar informāciju par konkrēta lietotāja rezervējumu, kura informāciju vēlaties mainīt. Pēc tam, kad esat mainījis informāciju, kuru vēlaties saglabāt, noklikšķiniet uz pogas “Atjaunināt”, lai to saglabātu. (sk. 6.19.att.).

## Update Reservation

Date:

2024-06-10

June

June

2024

2024

Previous month

Next month

Mo	Tu	We	Th	Fr	Sa	Su
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Update Reservation

Time:

10:00 (60.00 EUR)

11:30 (60.00 EUR)

13:30 (60.00 EUR)

14:30 (60.00 EUR)

16:00 (60.00 EUR)

17:30 (60.00 EUR)

19:00 (60.00 EUR)

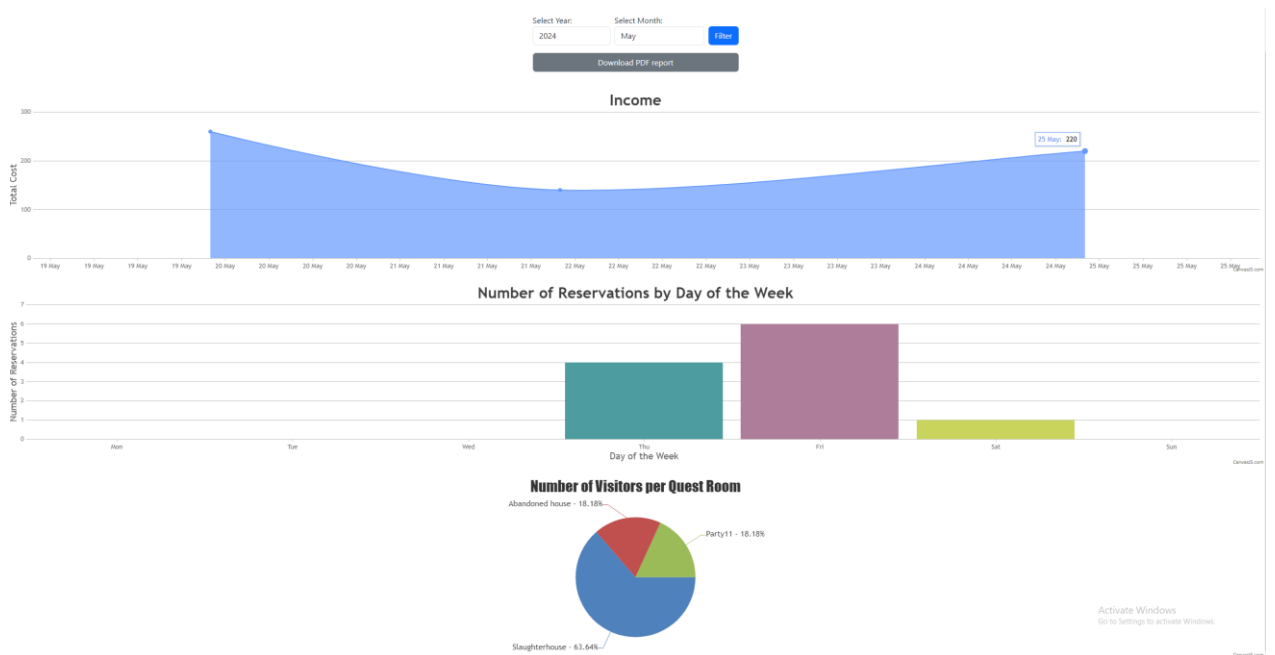
20:30 (80.00 EUR)

22:00 (80.00 EUR)

6.19.att Lietotāja rezervāciju rediģēšana forma

## Statistikas apskate un izejas informācijas pārskats

Ja esat ienācis sistēmā kā administrators, navigācijas joslā ir poga “Charts”, uz kuras noklikšķinot, jūs tiek novirzīts uz lapu, kurā atlases joslā izvēlaties vēlamo mēnesi un gadu un noklikšķinot uz pogas “Filter”, varat skatīt detalizētu pārskatu par veiktajām rezervācijām, populārāko lietotāju izvēli un mēneša ienākumiem (sk. 6.20. att.).



6.20.att Statistikas apskate

Noklikšķinot uz pogas Lejupielādēt PDF pārskatu, jūs lejupielādēsiet arī iepriekš filtrētā mēneša PDF failu (sk. 6.21. att.).

### Reservation Report

For the month of 05/2024

#### Total Cost by Date

Date	Total Cost
2024-05-02	80.00
2024-05-03	60.00
2024-05-10	60.00
2024-05-11	60.00
2024-05-23	60.00
2024-05-24	80.00
2024-05-30	300.00
2024-05-31	280.00
Total	980.00

#### Number of Reservations by Day of the Week

Day of the Week	Number of Reservations
Monday	0
Tuesday	0
Wednesday	0
Thursday	7
Friday	7
Saturday	1

#### Visitors by Rooms and Quests

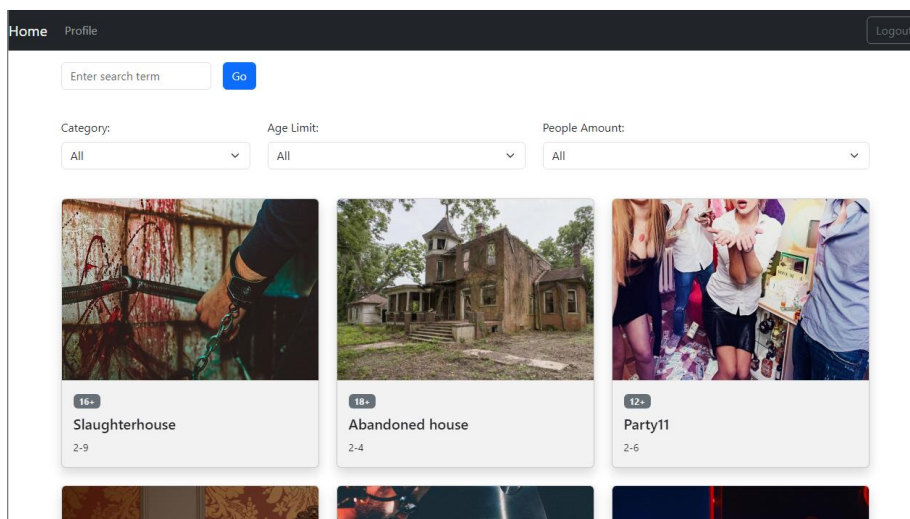
Quest	Visitors
Abandoned house	4
Party11	2
Slaughterhouse	7
The Bank job	2

Report generated on 2024-06-02

6.21.att PDF faila apskate

## Sesijas beigšana

Ja esat pieteicies kā lietotājs vai administrators, bet vēlaties izbeigt savu sesiju. Noklikšķiniet uz Iziet navigācijas joslā, kas atrodas vietnes augšējā labajā stūrī, un jūs tiksiet pieteikts no sava profila (sk. 6.22. att.).



6.22.att Sesijas beigšana

## 6.4. Testa piemērs

Šajā sadaļā ir sniegti vairāki testēšanas piemēri, kas palīdz pārbaudīt sistēmas funkcionalitāti un atbilstību noteiktajām prasībām. Katrai testējamajai prasībai ir sniegti konkrēti ievades dati vai situācijas apraksts, sagaidāmais rezultāts un testa statusa novērtējums. Šie testi aptver dažādus lietošanas gadījumus, sākot no jauna lietotāja reģistrācijas līdz komentāru dzēšanai un darbinieku datu rediģēšanai. Turpmāk tabulā (sk. 6.1.tabula) ir apkopoti testu piemēri, kas atspoguļo dažādas sistēmas mijiedarbības situācijas un to sagaidāmos rezultātus.

“Testēšanas pārskats”

6.1. tabula

Nr.	Prasības numurs	Prasības nosaukums	Ievaddati/situācijas apraksts	Sagaidāmais rezultāts	Statuss
1.	8.1.	Jauna lietotāja reģistrācija	Pareizi dati: lietotāja vārds, vārds, uzvārds, e-pasts, tālruna numurs, 1. parole, 2. parole	Datubāzē tiek pievienots ieraksts, uz e-pasta tiek atsūtīts paziņojums par veiksmīgu reģistrāciju	Pareizi
2.			Lietotāja vārds = Edgars332	Paziņojums par jau eksistējošo lietotāju vārdu	Pareizi



## 6.1. tabulas turpinājums

3.			Lietotāja e-pasts = edgars332gmail.com	Paziņojums par nepareizo e-pasta formātu	Pareizi
4.	4.1.	Kvestu istabu rezervācija	Pareizi dati : Datums, laiks , apmaksas veids	Databāzē tiek pievienots ieraksts, uz e-pasta tiek atsūtīts paziņojums par veiksmīgu reģistrāciju	Pareizi
5.			Lietotāja kartes vārds uzvārds = Ernests32 Dzērve#	Paziņojums par nepareizo vārdu uzvārdu formātu	Pareizi
6.			Lietotāja kartes numurs = 1234 1234 1234 1234	Paziņojums par to, ka Jūsu kartes dati jau bija saglabāti sistēmā.	Pareizi
7.	2.1.	Darbinieku datu rediģēšana	Pareizi dati : e-pasts <a href="mailto:ernests@gmail.com">ernests@gmail.com</a> nomainīts uz ernestsBerzins@gmail.com	Datubāzē ir atjaunināts e-pasta lauks	Pareizi
8.			Tālruņa numurs 23404583 nomainīts uz 20493758	Paziņojums par jau eksistējošo tālruņa numuru	Pareizi
9.			Parole nomaina qwerty2\$ uz asdasdasd1	Paziņojums par to ka parolē nav simbola	Pareizi
10.	7.2.	Komentāru dzēšana	Peles klikšķis uz pogas “Dzēst”	Ieraksts tiek dzēsts no datu bāze un visi saistīti dati arī tiek dzēsti	Pareizi
11.	1.1.	Kvestu istabu pievienošana	Pareizi dati: nosaukums, adrese, kategorija, vecumu ierobežojums, cilvēku skaits, apraksts, attēls	Datubāzē tiek pievienots ieraksts.	Pareizi
12.			Attēls photo.pdf	Paziņojums par nepareizo attēlu formātu	Pareizi
13.			Nosaukums = Scaryhouse	Paziņojums par jau eksistējošo nosaukumu	Pareizi

Atkārtotajā testēšanā visi testi tika veiksmīgi izpildīti, apliecinot, ka visas atrastās kļūdas ir izlabotas un sistēma darbojas atbilstoši noteiktajām prasībām.

## NOBEIGUMS

Sistēmas izstrādes laikā tika veikti visi nepieciešamie uzdevumi, un pašlaik projekts ir pilnībā pabeigts. Kvalifikācijas darba ietvaros tika nodrošināta produkta pilnīga funkcionalitāte, pievēršot uzmanību lietotāju draudzīgumam, elastībai papildu moduļu un funkciju integrācijai, sistēmas pārnesamībai uz citām platformām un optimizācijai vājām ierīcēm. Lai sasniegtu šos mērķus, tika izmantotas tādas tehnoloģijas kā Visual Studio un Visual Studio Code, XAMPP, OpenSSL, HTML5, CSS3, JavaScript, PHP un Bootstrap, Stripe un MPdf.

Izstrādes procesā tika veikti vairāki nozīmīgi uzlabojumi un pievienotas papildu funkcijas. Projekta gaitā tika izstrādāta iespēja veikt maksājumus ar karti, nodrošinot tiešu līdzekļu ieturēšanu. Tāpat tika izveidota funkcija, kas ļauj nosūtīt ziņojumus uz e-pastu un ģenerēt un izvadīt informāciju PDF failos. Šo papildfunkciju pievienošana nodrošināja plašākas iespējas un uzlaboja sistēmas kopējo funkcionalitāti.

Nākotnē plānots uzlabot administratora paneli, lietotāju profilus un pievienot mobilās notifikācijas, lai sistēma būtu vēl lietotājam draudzīgāka un pielāgojamāka. Turpmākie uzlabojumi nodrošinās sistēmas atbilstību jauniem lietotāju pieprasījumiem un tehnoloģiju attīstības tendencēm, tādējādi padarot to vēl pievilcīgāku un funkcionālāku dažādiem klientiem.

## INFORMĀCIJAS AVOTI

1. Kvestu istaba viente - <https://escaperoommap.lv/> - (Resurss apsakstīts 01.01.2024).
2. Kvestu istaba viente <https://www.kvesti.lv/lv> - (Resurss apsakstīts 01.01.2024).
3. Kvestu istaba viente <https://exitgame.lv/ru/> - (Resurss apsakstīts 01.01.2024).
4. Hypertext Preprocessor (PHP) dokumentācija - <https://www.php.net/> - (Resurss apsakstīts 01.01.2024).
5. CSS dokumentācija - <https://developer.mozilla.org/en-US/docs/Web/CSS> - (Resurss apsakstīts 01.01.2024).
6. HTML5 dokumentācija – <https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5> - (Resurss apsakstīts 01.01.2024).
7. MySQL (SQL) dokumentācija - <https://dev.mysql.com/doc/> - (Resurss apsakstīts 01.01.2024).
8. XAMPP dokumentācija - [https://www.apachefriends.org/faq\\_windows.html](https://www.apachefriends.org/faq_windows.html) - (Resurss apsakstīts 18.01.2024).
9. MPDF dokumentācija - <https://mpdf.github.io/> - (Resurss apsakstīts 20.04.2024).
10. Mans projekts - <https://github.com/KIvanovs/DBQuestRoom> - (Resurss apsakstīts 10.06.2024).
11. PHPMailer dokumentācija - <https://github.com/PHPMailer/PHPMailer> - (Resurss apsakstīts 20.04.2024).
12. Diagrammu izveidei - <https://app.diagrams.net/> - (Resurss apsakstīts 10.02.2024).
13. Problēmu risināšana - <https://stackoverflow.com/> - (Resurss apsakstīts 10.02.2024).
14. Bootstrap dokumentācija - <https://getbootstrap.com/> - (Resurss apsakstīts 17.05.2024).
15. OpenSSL - <https://slproweb.com/products/Win32OpenSSL.html> - (Resurss apsakstīts 28.05.2024).
16. Stripe documentation - <https://docs.stripe.com/payments?payments=popular> - (Resurss apsakstīts 28.05.2024).

# PIELIKUMI

**//administratoru pievienošana**

```

<?php

include '../includes/dbcon.php';

if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// Check if form is submitted
if($_SERVER['REQUEST_METHOD'] === 'POST'){

    // Get form data
    $name = $_POST['name'];
    $surname = $_POST['surname'];
    $email = $_POST['email'];
    $password = password_hash($_POST['password'], PASSWORD_DEFAULT); // Hash the password
    $password_check = $_POST['password'];
    $personCode = $_POST['personCode'];
    $phoneNumber = $_POST['phoneNumber'];

    // Validate form data
    if (empty($name) || empty($surname) || empty($email) || empty($password_check) ||
empty($personCode) || empty($phoneNumber)) {
        echo "Please fill in all fields!";
        exit();
    }

    if (strlen($name) > 30 || strlen($surname) > 30 || strlen($email) > 30 || strlen($phoneNumber) >
30){
        echo "Too long text , maximum 30 symbols!";
        exit();
    }

    if (strlen($personCode) !== 12){
        echo "Personal Code should be 12 symbols!";
        exit();
    }

    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        echo "Invalid email format!";
        exit();
    }

    if (!preg_match("/^[a-zA-Z ]*$/", $name) || !preg_match("/^[a-zA-Z ]*$/", $surname)) {
        echo "Name and surname should only contain letters and spaces!";
        exit();
    }

    if (!preg_match("/^[0-9]*$/", $phoneNumber)) {
        echo "Invalid phone number format!";
        exit();
    }

    if (!preg_match("/^[0-9-]*$/", $personCode)) {

```

```

        echo "Invalid personal code format!";
        exit();
    }

    if (strlen($password_check) < 8) {
        echo "Password should be at least 8 characters long!";
        exit();
    }

    // Check if nickname, email or phone number already exists in database
    $check_duplicate_user_query = "SELECT * FROM users WHERE email = '$email' OR
    phoneNumber = '$phoneNumber'";
    $check_duplicate_user_result = $conn->query($check_duplicate_user_query);

    if ($check_duplicate_user_result->num_rows > 0) {
        echo "Email or phone number already exists!";
        exit();
    }

    $check_duplicate_admin_query = "SELECT * FROM admin WHERE personCode = '$personCode'
    OR email = '$email' OR phoneNumber = '$phoneNumber'";
    $check_duplicate_admin_result = $conn->query($check_duplicate_admin_query);

    if ($check_duplicate_admin_result->num_rows > 0) {
        echo "Personal code, email or phone number already exists!";
        exit();
    }

    // Insert data into database with duplicate check
    $query = "INSERT INTO admin (name, surname, email, password, personCode, phoneNumber)
    VALUES ('$name', '$surname', '$email', '$password', '$personCode', '$phoneNumber')";

    if(mysqli_query($conn, $query)){
        if(mysqli_affected_rows($conn) > 0){
            header("Location: ../admin/admin.php");
            exit();
        } else{
            echo "Error: Email, person code, or phone number already exists in the database.";
        }
    } else{
        echo "Error: " . mysqli_error($conn);
    }
}

mysqli_close($conn);

```

?>

**// administratora dzěšana**

<?php

```

include '../includes/dbcon.php';

if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

```

```

// If an admin is being deleted
if (isset($_POST['delete_admin'])) {
    $admin_id = $_POST['admin_id'];

    // Delete the admin from the database
    $delete_query = "DELETE FROM admin WHERE ID = $admin_id";
    mysqli_query($conn, $delete_query);
}

header("Location: ../admin/admin.php");
exit();
?>

```

### **//administratoru rediģēšana**

```
<?php
```

```

include '../includes/header.php';
// Check if the user is not an admin
if (!isset($_SESSION['admin_id']) || empty($_SESSION['admin_id'])) {
    // Redirect the user to the admin login page
    header("Location: ../register_login/loginform.php");
    exit();
}

include '../includes/dbcon.php';

if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// Check if form is submitted
if(isset($_POST['update_admin'])){

    // Get admin ID
    $admin_id = $_POST['admin_id'];

    // Get form data
    $name = $_POST['name'];
    $surname = $_POST['surname'];
    $email = $_POST['email'];
    $personCode = $_POST['personCode'];
    $password = mysqli_real_escape_string($conn, $_POST['password']);
    $phoneNumber = $_POST['phoneNumber'];

    // Validate form data
    if (empty($name) || empty($surname) || empty($email) || empty($personCode) ||
empty($phoneNumber)) {
        echo "Please fill in all fields!";
        exit();
    }

    if (strlen($name) > 30 || strlen($surname) > 30 || strlen($email) > 30 || strlen($phoneNumber) >
30){
        echo "Too long text , maximum 30 symbols!";
        exit();
    }
}

```

```

    }

    if (strlen($personCode) !== 12){
        echo "Personal Code should be 12 symbols!";
        exit();
    }

    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        echo "Invalid email format!";
        exit();
    }

    if (!preg_match("/^[a-zA-Z ]*$/", $name) || !preg_match("/^[a-zA-Z ]*$/", $surname)) {
        echo "Name and surname should only contain letters and spaces!";
        exit();
    }

    if (!preg_match("/^[0-9]*$/", $phoneNumber)) {
        echo "Invalid phone number format!";
        exit();
    }

    if (!preg_match("/^[0-9]*$/", $personCode)) {
        echo "Invalid personal code format!";
        exit();
    }

    // Check if email, personCode and phoneNumber already exist in database
    $query = "SELECT * FROM admin WHERE (email='$email' OR personCode='$personCode' OR
phoneNumber='$phoneNumber') AND ID!='$admin_id'";
    $result = mysqli_query($conn, $query);
    if (mysqli_num_rows($result) > 0) {
        echo "Error: Email, person code, or phone number already exists in the database A.";
        exit();
    }

    $query = "SELECT * FROM users WHERE email='$email' OR phoneNumber='$phoneNumber'";
    $result = mysqli_query($conn, $query);
    if (mysqli_num_rows($result) > 0) {
        echo "Error: Email, person code, or phone number already exists in the database U.";
        exit();
    }

    // Hash the password if it's not empty
    if (!empty($password)) {
        if (strlen($password) < 8) {
            echo "Password should be at least 8 characters long!";
            exit();
        }
        $hashed_password = password_hash($password, PASSWORD_DEFAULT);
        $query = "UPDATE admin SET name='$name', surname='$surname', email='$email',
password='$hashed_password', personCode='$personCode', phoneNumber='$phoneNumber'
WHERE ID='$admin_id'";
    } else {
        $query = "UPDATE admin SET name='$name', surname='$surname', email='$email',
personCode='$personCode', phoneNumber='$phoneNumber' WHERE ID='$admin_id'";
    }
    // Update data in database

```



```

if(mysqli_query($conn, $query)){
    header("Location: ../admin/admin.php");
    exit();
} else{
    echo "Error updating admin: " . mysqli_error($conn);
}
}

// Get current admin data
$admin_id = $_POST['admin_id'];
$query = "SELECT * FROM admin WHERE ID='$admin_id'";
$result = mysqli_query($conn, $query);

```

### //PDF failu generešana

```

<?php
include '../includes/dbcon.php';
require_once '../vendor/autoload.php';

$mpdf = new \Mpdf\Mpdf();

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Получаем выбранные месяц и год из POST-запроса
$month = isset($_POST['selectedMonth']) ? $_POST['selectedMonth'] : date('m');
$year = isset($_POST['selectedYear']) ? $_POST['selectedYear'] : date('Y');

// Company header and title
$html = '
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <style>
        body { font-family: Arial, sans-serif; }
        .header { text-align: center; margin-bottom: 20px; }
        .header img { max-width: 150px; }
        h1 { color: #333; }
        h2 { color: #555; margin-top: 40px; }
        table { width: 100%; border-collapse: collapse; margin-top: 20px; }
        th, td { border: 1px solid #ddd; padding: 10px; text-align: left; }
        th { background-color: #f2f2f2; }
        .total { font-weight: bold; }
    </style>
</head>
<body>
    <div class="header">
        
        <h1>Reservation Report</h1>
        <p>For the month of ' . $month . ' ' . $year . '</p>
    </div>
';

// First query - Total cost by date
$sql1 = "SELECT SUM(cost) as total_cost, DATE(date) as date
FROM reservation
WHERE MONTH(date) = '$month' AND YEAR(date) = '$year'";

```

```

        GROUP BY date";
$result1 = $conn->query($sql1);
$html .= '<h2>Total Cost by Date</h2>';
$html .= '<table><thead><tr><th>Date</th><th>Total Cost</th></tr></thead><tbody>';
$total_month_cost = 0;
while ($row = $result1->fetch_assoc()) {
    $html .= '<tr><td>' . $row['date'] . '</td><td>' . number_format($row['total_cost'], 2) . '</td></tr>';
    $total_month_cost += $row['total_cost'];
}
$html .= '<tr class="total"><td>Total</td><td>' . number_format($total_month_cost, 2) . '</td></tr>';
$html .= '</tbody></table>';

// Second query - Number of reservations by day of the week
$days_of_week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'];
$sql2 = "SELECT WEEKDAY(date) AS day_of_week, COUNT(*) AS reservation_count
        FROM reservation
        WHERE MONTH(date) = '$month' AND YEAR(date) = '$year'
        GROUP BY day_of_week";
$result2 = $conn->query($sql2);

$reservations_by_day = array_fill(0, 7, 0);
while ($row = $result2->fetch_assoc()) {
    $reservations_by_day[$row['day_of_week']] = $row['reservation_count'];
}

$html .= '<h2>Number of Reservations by Day of the Week</h2>';
$html .= '<table><thead><tr><th>Day of the Week</th><th>Number of
Reservations</th></tr></thead><tbody>';
foreach ($days_of_week as $index => $day_name) {
    $html .= '<tr><td>' . $day_name . '</td><td>' . $reservations_by_day[$index] . '</td></tr>';
}
$html .= '</tbody></table>';

// Third query - Visitors by rooms and quests
$sql3 = "SELECT q.name, COUNT(r.room_id) as visitors
        FROM reservation r
        JOIN quests q ON r.room_id = q.id
        WHERE MONTH(r.date) = '$month' AND YEAR(r.date) = '$year'
        GROUP BY q.name";
$result3 = $conn->query($sql3);
$html .= '<h2>Visitors by Rooms and Quests</h2>';
$html .= '<table><thead><tr><th>Quest</th><th>Visitors</th></tr></thead><tbody>';
while ($row = $result3->fetch_assoc()) {
    $html .= '<tr><td>' . $row['name'] . '</td><td>' . $row['visitors'] . '</td></tr>';
}
$html .= '</tbody></table>';

// Footer
$html .= '
    <p>Report generated on ' . date('Y-m-d') . '</p>
</body>
</html>';

$mpdf->WriteHTML($html);
$mpdf->Output('reservation_report_' . $month . '_' . $year . '.pdf', 'D');

// Close connection
$conn->close();
?>

```

**//komentāru dzēšana**

```

<?php
session_start();

include '../includes/dbcon.php';

if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// Get the comment ID from the form
$comment_id = $_POST['comment_id'];
$quest_id = $_POST['quest_id'];
// Start transaction
mysqli_begin_transaction($conn);

try {
    // First, delete all replies to this comment
    $delete_replies_query = "DELETE FROM comment WHERE reply_to = $comment_id";
    $delete_replies_result = mysqli_query($conn, $delete_replies_query);

    if (!$delete_replies_result) {
        throw new Exception("Error deleting replies: " . mysqli_error($conn));
    }

    // Then, delete the main comment
    $delete_comment_query = "DELETE FROM comment WHERE ID = $comment_id";
    $delete_comment_result = mysqli_query($conn, $delete_comment_query);

    if (!$delete_comment_result) {
        throw new Exception("Error deleting comment: " . mysqli_error($conn));
    }

    // Commit the transaction
    mysqli_commit($conn);

    // The comment and its replies were deleted successfully
    header("Location: ../room/quest_info.php?ID=$quest_id");
    exit();
} catch (Exception $e) {
    // Rollback the transaction if any query failed
    mysqli_rollback($conn);

    // Display error message
    echo "Error: " . $e->getMessage();
}

mysqli_close($conn);
?>

```

### **// komentāru rediģēšana**

```

<?php
session_start();
include '../includes/dbcon.php';

if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

$comment_id = $_POST['comment_id'];
$quest_id = $_POST['quest_id'];
$comment_text = mysqli_real_escape_string($conn, $_POST['comment']);

```

```

$update_query = "UPDATE comment SET comment = '$comment_text' WHERE ID = $comment_id";

if (mysqli_query($conn, $update_query)) {
    header("Location: ../room/quest_info.php?ID=$quest_id");
    exit();
} else {
    echo "Error updating comment: " . mysqli_error($conn);
}

mysqli_close($conn);
?>

```

### //komentāru pievienošana

```

<?php
session_start();

if (isset($_SESSION['user_id'])) {
    $user_id = $_SESSION['user_id'];
} elseif (isset($_SESSION['admin_id'])) {
    $admin_id = $_SESSION['admin_id'];
} else {

    header("Location: ../register_login/loginform.php");
    exit();
}

// Handle comment submission
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['comment'])) {
    include '../includes/dbcon.php'; // Reconnect to the database

    if (!$conn) {
        die("Connection failed: " . mysqli_connect_error());
    }

    $comment = mysqli_real_escape_string($conn, $_POST['comment']);
    $quest_id = mysqli_real_escape_string($conn, $_POST['quest_id']);
    $reply_to = isset($_POST['reply_to']) ? mysqli_real_escape_string($conn, $_POST['reply_to']) : null;

    // Check admin or user
    if (isset($user_id)) {
        $insert_query = "INSERT INTO comment (comment, user_id, admin_id, quest_id, reply_to,
creation_date)
VALUES ('$comment', '$user_id', NULL, '$quest_id', '$reply_to',
CURRENT_TIMESTAMP)";
    } elseif (isset($admin_id)) {
        $insert_query = "INSERT INTO comment (comment, user_id, admin_id, quest_id, reply_to,
creation_date)
VALUES ('$comment', NULL, '$admin_id', '$quest_id', '$reply_to',
CURRENT_TIMESTAMP)";
    }

    if (mysqli_query($conn, $insert_query)) {
        echo "Comment added successfully!";
        // Redirect to the quest_info.php page to prevent form resubmission
        header("Location: ../room/quest_info.php?ID=$quest_id");
        exit();
    } else {
        echo "Error: " . $insert_query . "<br>" . mysqli_error($conn);
    }
}

```

```

        // Close database connection
        mysqli_close($conn);
    } else {

        header("Location: ../room/quest_list.php");
        exit();
    }
?>

```

### **// lietotāju dzēšana**

```

<?php
session_start();

// Check if the user is not an admin
if (!isset($_SESSION['admin_id']) || empty($_SESSION['admin_id'])) {
    // Redirect the user to the admin login page
    header("Location: ../register_login/loginform.php");
    exit();
}

include '../includes/dbcon.php';

if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// Get the user ID to delete
$user_id = $_POST['user_id'];

// Delete the user and their reservations
$query = "DELETE FROM users WHERE ID = " . $user_id;
mysqli_query($conn, $query);

$query = "DELETE FROM reservation WHERE client_id = " . $user_id;
mysqli_query($conn, $query);

mysqli_close($conn);

// Redirect the user back to the user list
header("Location: ../profile/profile_info.php");
exit();
?>

```

### **// lietotāju rediģēšana**

```

<?php
session_start();

// Check if the user is not an admin
if (!isset($_SESSION['admin_id']) || empty($_SESSION['admin_id'])) {
    // Redirect the user to the admin login page
    header("Location: ../register_login/loginform.php");
    exit();
}

include '../includes/dbcon.php';

if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

```

```

}

// Get the user ID to update
$user_id = $_POST['user_id'];

// Get the new user data from the form
$nickname = mysqli_real_escape_string($conn, $_POST['nickname']);
$name = mysqli_real_escape_string($conn, $_POST['name']);
$surname = mysqli_real_escape_string($conn, $_POST['surname']);
$email = mysqli_real_escape_string($conn, $_POST['email']);
$password = mysqli_real_escape_string($conn, $_POST['password']);
$phone_number = mysqli_real_escape_string($conn, $_POST['phoneNumber']);

// Validate form data
if (empty($nickname) || empty($name) || empty($surname) || empty($email) ||
empty($phone_number)) {
    echo "Please fill in all fields!";
    exit();
}

if (strlen($nickname) > 30 || strlen($name) > 30 || strlen($surname) > 30 || strlen($email) > 30
|| strlen($phone_number) > 30){
    echo "Too long text , maximum 30 symbols!";
    exit();
}

if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    echo "Invalid email format!";
    exit();
}

if (!preg_match("/^[a-zA-Z ]*$/", $name) || !preg_match("/^[a-zA-Z ]*$/", $surname)) {
    echo "Name and surname should only contain letters and spaces!";
    exit();
}

if (!preg_match("/^[0-9]*$/", $phone_number)) {
    echo "Invalid phone number format!";
    exit();
}

// Check if nickname, email or phone number already exists in database
$check_duplicate_user_query = "SELECT * FROM users WHERE (email = '$email' OR
phoneNumber = '$phone_number') AND ID!='$user_id'";
$check_duplicate_user_result = $conn->query($check_duplicate_user_query);

if ($check_duplicate_user_result->num_rows > 0) {
    echo "Email or phone number already exists!";
    exit();
}

$check_duplicate_admin_query = "SELECT * FROM admin WHERE email = '$email' OR
phoneNumber = '$phone_number'";
$check_duplicate_admin_result = $conn->query($check_duplicate_admin_query);

if ($check_duplicate_admin_result->num_rows > 0) {
    echo "Personal code, email or phone number already exists!";
    exit();
}

```

```

// Hash the password if it's not empty
if (!empty($password)) {
    if (strlen($password) < 8) {
        echo "Password should be at least 8 characters long!";
        exit();
    }
    $hashed_password = password_hash($password, PASSWORD_DEFAULT);
    $query = "UPDATE users SET nickname='$nickname', name='$name', surname='$surname',
email='$email', password='$hashed_password', phoneNumber='$phone_number' WHERE
ID=$user_id";
} else {
    $query = "UPDATE users SET nickname='$nickname', name='$name', surname='$surname',
email='$email', phoneNumber='$phone_number' WHERE ID=$user_id";
}

if (mysqli_query($conn, $query)) {
    // User updated successfully, redirect back to the user list
    header("Location: ../profile/profile_info.php");
    exit();
} else {
    // Error updating user, display an error message
    echo "Error updating user: " . mysqli_error($conn);
}

mysqli_close($conn);

```

### **//autorizācija**

```

<?php
session_start();

// unset all session variables
$_SESSION = array();

// destroy session
session_destroy();

session_start();

include '../includes/dbcon.php';

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// check if form is submitted
if (isset($_POST['login'])) {
    $email = $_POST['email'];
    $password = $_POST['password'];

    // retrieve user data from users table
    $sql = "SELECT * FROM users WHERE email='$email'";
    $result = $conn->query($sql);

    if ($result->num_rows == 1) {
        // user exists, check if password matches
        $row = $result->fetch_assoc();
        if (password_verify($password, $row['password'])) {
            // login successful, redirect to dashboard

```

```

        $_SESSION['user_id'] = $row['ID'];
        $_SESSION['nickname'] = $row['nickname'];
        header("Location: ../room/quest_list.php");
        exit();
    } else {
        echo "Invalid password. Please try again.";
    }
} else {
    // retrieve admin data from admin table
    $sql = "SELECT * FROM admin WHERE email='$email'";
    $result = $conn->query($sql);

    if ($result->num_rows == 1) {
        // admin exists, check if password matches
        $row = $result->fetch_assoc();
        if (password_verify($password, $row['password'])) {
            // login successful, redirect to dashboard
            $_SESSION['admin_id'] = $row['ID'];
            $_SESSION['admin_name'] = $row['name'];
            header("Location: ../room/quest_list.php");
            exit();
        } else {
            echo "Invalid password. Please try again.";
        }
    } else {
        echo "Invalid email. Please try again.";
    }
}
}

// close database connection
$conn->close();
?>

```

## //reģistrācija

```

<?php
include '../includes/dbcon.php';
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Get form data
$nickname = $_POST['nickname'];
$name = $_POST['name'];
$surname = $_POST['surname'];
$email = $_POST['email'];
$password = $_POST['password'];
$confirm_password = $_POST['confirm_password'];
$phone = $_POST['phone'];

// Validate form data
if (empty($nickname) || empty($name) || empty($surname) || empty($email) ||
empty($password) || empty($confirm_password) || empty($phone)) {
    echo "Please fill in all fields!";
}

```



```

        exit();
    }

    if (strlen($nickname) > 30 || strlen($name) > 30 || strlen($surname) > 30 || strlen($email) > 30
|| strlen($confirm_password) > 30 || strlen($phone) > 30){
        echo "Too long text , maximum 30 symbols!";
        exit();
    }

    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        echo "Invalid email format!";
        exit();
    }

    if (!preg_match("/^[a-zA-Z ]*$/", $name) || !preg_match("/^[a-zA-Z ]*$/", $surname)) {
        echo "Name and surname should only contain letters and spaces!";
        exit();
    }

    if (!preg_match("/^[0-9]*$/", $phone)) {
        echo "Invalid phone number format!";
        exit();
    }

    if (strlen($password) < 8) {
        echo "Password should be at least 8 characters long!";
        exit();
    }

    if ($password !== $confirm_password) {
        echo "Password and confirm password do not match!";
        exit();
    }

    // Hash password
    $hashed_password = password_hash($password, PASSWORD_DEFAULT);

    // Check if nickname, email or phone number already exists in database
    $check_duplicate_user_query = "SELECT * FROM users WHERE nickname = '$nickname' OR
email = '$email' OR phoneNumber = '$phone'";
    $check_duplicate_user_result = $conn->query($check_duplicate_user_query);

    if ($check_duplicate_user_result->num_rows > 0) {
        echo "Nickname ,email or phone number already exists!";
        exit();
    }

    $check_duplicate_admin_query = "SELECT * FROM admin WHERE email = '$email' OR
phoneNumber = '$phone'";
    $check_duplicate_admin_result = $conn->query($check_duplicate_admin_query);

    if ($check_duplicate_admin_result->num_rows > 0) {
        echo "Email or phone number already exists!";
        exit();
    }

    // Insert data into database
    $sql = "INSERT INTO users (nickname, password, email, name, surname, phoneNumber)
VALUES ('$nickname', '$hashed_password', '$email' , '$name', '$surname', '$phone')";

    if ($conn->query($sql) === TRUE) {
        echo "New record created successfully";
    }

```

```

echo "<p>Please <a href='../register_login/loginform.php'>log in</a> to start session.</p>";

require '../vendor/autoload.php';

$mail = new PHPMailer(true);
try {
    $mail->isSMTP();
    $mail->Host = 'smtp.gmail.com';
    $mail->SMTPAuth = true;
    $mail->Username = 'kirillquestroom@gmail.com'; // SMTP username
    $mail->Password = 'tdiscwhdffittzmz'; // SMTP password
    $mail->SMTPSecure = 'tls';
    $mail->Port = 587;

    $mail->setFrom('kirillquestroom@gmail.com', 'KirillQuestRoom');
    $mail->addAddress($email, $name); // Add a recipient

    $mail->isHTML(true); // Set email format to HTML
    $mail->Subject = 'Registration Successful';
    $mail->Body = "Hello $name,<br><br>Thank you for registering on our
site.<br><br>Best Regards,<br>The Team Kirill Quest Room";

    $mail->send();
    echo 'Message has been sent';
} catch (Exception $e) {
    echo 'Message could not be sent. Mailer Error: ', $mail->ErrorInfo;
}
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

// Close database connection
$conn->close();
?>

```

#### **// kvestu istabu pievienošana**

```

<?php
include '../includes/dbcon.php';

function generateRandomFilename($prefix = 'room_image') {
    $randomNumber = rand(1, 999999999); // You can adjust the range as needed
    return $prefix . $randomNumber;
}

if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    // Get form data
    $name = mysqli_real_escape_string($conn, $_POST['name']);
    $adress = mysqli_real_escape_string($conn, $_POST['adress']); // Escape the address string
    $category = mysqli_real_escape_string($conn, $_POST['category']); // Escape the category string
    $peopleAmount = $_POST['peopleAmount'];
    $ageLimit = $_POST['ageLimit'];
    $description = mysqli_real_escape_string($conn, $_POST['description']);
}

```

```

// Validate form data

if (empty($name) || empty($category) || empty($adress) || empty($peopleAmount) ||
empty($ageLimit) || empty($description)) {
    echo "Please fill in all fields!";
    exit();
}

if (strlen($name) > 50 || strlen($category) > 30 || strlen($adress) > 50 || strlen($peopleAmount)
> 30 || strlen($ageLimit) > 30 || strlen($description) > 500) {
    echo "Too long text, maximum 30 symbols for the name, category, address, discount
and peopleAmount and maximum 500 symbols for the description!";
    exit();
}

if (!preg_match("/^[a-zA-Z ]*$/", $category)) {
    echo "Category should only contain letters and spaces!";
    exit();
}

if (!preg_match('/^[0-9-]+$/', $peopleAmount)) {
    echo "People Amount should be a number and hyphen!";
    exit();
}

if (!is_numeric($ageLimit) || $ageLimit <= 0 && $ageLimit >= 99) {
    echo "Age Limit should be a number greater than 0 and without symbols!";
    exit();
}

// Check for duplicate name
$query = "SELECT * FROM quests WHERE name='$name'";
$result = mysqli_query($conn, $query);

if (mysqli_num_rows($result) > 0) {
    echo "Quest with this name already exists!";
    exit();
}

// Process uploaded file
if (isset($_FILES['photo']) && $_FILES['photo']['error'] === UPLOAD_ERR_OK) {
    $file = $_FILES['photo'];
    $filetype = $file['type'];

    // Check if the file is an image
    $allowed_types = ['image/jpeg', 'image/png', 'image/gif', 'image/jpg'];
    if (!in_array($filetype, $allowed_types)) {
        die('Invalid file type.');
```

```

    }

    // Generate a random filename
    $filename = generateRandomFilename();

    // Ensure the filename is unique
    while (file_exists('./images/' . $filename)) {
        $filename = generateRandomFilename();
    }
}
```

```

$upload_dir = './images/';
```

```

$upload_path = $upload_dir . $filename;

move_uploaded_file($file['tmp_name'], $upload_path);
} else {
    die('Please upload a photo!');
}

// Insert data into 'adress' table
$sql_insert_adress = "INSERT INTO adress (buildingAdress) VALUES ('$adress')";
if (mysqli_query($conn, $sql_insert_adress)) {
    // Get the ID of the inserted address
    $adress_id = mysqli_insert_id($conn);

    // Insert data into 'questcategory' table
    $sql_insert_questcategory = "INSERT INTO questcategory (ageLimit, categoryName)
VALUES ('$ageLimit', '$category')";
    if (mysqli_query($conn, $sql_insert_questcategory)) {
        // Get the ID of the inserted category
        $questcategory_id = mysqli_insert_id($conn);

        // Insert data into 'quests' table with photo path, address_id, and
questcategory_id
        $sql_insert_quest = "INSERT INTO quests (name, peopleAmount,
description, photoPath, adress_id, questCategory_id) VALUES ('$name', '$peopleAmount',
'$description', '$upload_path', '$adress_id', '$questcategory_id')";
        if (mysqli_query($conn, $sql_insert_quest)) {
            echo "New quest added successfully!";
            echo "<a href=../room/quest_form.php>Back to quests info page<a>";
";
        } else {
            echo "Error inserting into 'quests' table: " . mysqli_error($conn);
        }
    } else {
        echo "Error inserting into 'questcategory' table: " . mysqli_error($conn);
    }
} else {
    echo "Error inserting into 'adress' table: " . mysqli_error($conn);
}
}

// Close database connection
mysqli_close($conn);
?>

```

### **//rezervāciju dzēšana**

```

<?php
session_start();

include '../includes/dbcon.php';

if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// Get the room ID from the form
$reserv_id = $_POST['reserv_id'];

// Delete the comment from the database
$query = "DELETE FROM reservation WHERE ID = $reserv_id";
$result = mysqli_query($conn, $query);

```

```

if ($result) {
    // The comment was deleted successfully,
    header("Location: ../profile/profile_info.php");
    exit();
} else {
    // The comment was not deleted, show an error message
    echo "Error deleting comment.";
}

mysqli_close($conn);
?>

// kvestu istabu dzēšana

<?php
session_start();

include '../includes/dbcon.php';

if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// Get the room ID from the form
$id = $_POST['id'];

// Fetch the photo path, adress_id, and questCategory_id from the database
$query = "SELECT photoPath, adress_id, questCategory_id FROM quests WHERE ID = $id";
$result = mysqli_query($conn, $query);

if ($result) {
    $row = mysqli_fetch_assoc($result);
    $photoPath = $row['photoPath'];
    $adress_id = $row['adress_id'];
    $questCategory_id = $row['questCategory_id'];

    // Delete the photo file
    if (unlink($photoPath)) {
        // Photo file deleted successfully, now delete the record from the database
        $deleteQuestQuery = "DELETE FROM quests WHERE ID = $id";
        $deleteAdressQuery = "DELETE FROM adress WHERE ID = $adress_id";
        $deleteCategoryQuery = "DELETE FROM questcategory WHERE ID = $questCategory_id";

        // Start transaction
        mysqli_begin_transaction($conn);

        try {
            $deleteQuestResult = mysqli_query($conn, $deleteQuestQuery);
            $deleteAdressResult = mysqli_query($conn, $deleteAdressQuery);
            $deleteCategoryResult = mysqli_query($conn, $deleteCategoryQuery);

            if ($deleteQuestResult && $deleteAdressResult && $deleteCategoryResult) {
                // Commit transaction
                mysqli_commit($conn);
                // The record was deleted successfully
                header("Location: ../room/quest_form.php");
                exit();
            } else {

```

```

        // Rollback transaction
        mysqli_rollback($conn);
        // The record was not deleted, show an error message
        echo "Error deleting record from the database.";
    }
} catch (Exception $e) {
    // Rollback transaction in case of error
    mysqli_rollback($conn);
    echo "Error: " . $e->getMessage();
}
} else {
    // Error deleting the photo file
    echo "Error deleting photo file.";
}
} else {
    // Error fetching photo path from the database
    echo "Error fetching photo path from the database.";
}
}

mysqli_close($conn);
?>

```

### //kvestu istabu filtrēšana un meklēšana

```

<?php
$pageTitle = 'Quest list';
include_once '../includes/header.php';

echo "
<div class='d-flex container py-3 '>
  <form method='GET' class='row g-3'>
    <div class='col-12 d-flex'>
      <input type='text' class='form-control me-3' name='search' id='search' placeholder='Enter search
term'>
      <button type='submit' class='btn btn-primary'>Go</button>
    </div>
  </form>
</div>";

include '../includes/dbcon.php';

if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// Select all categories from the database
$query = "SELECT DISTINCT categoryName FROM questcategory";
$result = mysqli_query($conn, $query);
$categories = mysqli_fetch_all($result, MYSQLI_ASSOC);

// Select all age limits from the database
$query = "SELECT DISTINCT ageLimit FROM questcategory ORDER BY ageLimit DESC";
$result = mysqli_query($conn, $query);
$ageLimits = mysqli_fetch_all($result, MYSQLI_ASSOC);

// Select all age limits from the database
$query = "SELECT DISTINCT peopleAmount FROM quests";

```

```

$result = mysqli_query($conn, $query);
$peopleAmounts = mysqli_fetch_all($result, MYSQLI_ASSOC);

// Don't close database connection and save data in result
mysqli_free_result($result);
?>
<div class='container my-4'>
<div class='row'>
<div class='col-md-3 mb-3'>
<label for='category' class='form-label'>Category:</label>
<select id='category' class='form-select' onchange='filterCards()'>
<option value=''>All</option>
<?php foreach ($categories as $category): ?>
<option value='<?php echo htmlspecialchars($category['categoryName']); ?>'><?php echo
htmlspecialchars($category['categoryName']); ?></option>
<?php endforeach; ?>
</select>
</div>
<div class='col-md-4 mb-3'>
<label for='ageLimit' class='form-label'>Age Limit:</label>
<select id='ageLimit' class='form-select' onchange='filterCards()'>
<option value=''>All</option>
<?php foreach ($ageLimits as $ageLimit): ?>
<option value='<?php echo htmlspecialchars($ageLimit['ageLimit']); ?>'><?php echo
htmlspecialchars($ageLimit['ageLimit']); ?>+</option>
<?php endforeach; ?>
</select>
</div>
<div class='col-md-5 mb-3'>
<label for='peopleAmount' class='form-label'>People Amount:</label>
<select id='peopleAmount' class='form-select' onchange='filterCards()'>
<option value=''>All</option>
<?php foreach ($peopleAmounts as $peopleAmount): ?>
<option value='<?php echo htmlspecialchars($peopleAmount['peopleAmount']); ?>'><?php echo
htmlspecialchars($peopleAmount['peopleAmount']); ?></option>
<?php endforeach; ?>
</select>
</div>
</div>
</div>

<?php
// Select all quests from the database, or search for a specific quest if a search query was submitted
$query = "SELECT q.ID, q.name, q.peopleAmount, q.description, q.photoPath,
a.buildingAdress,
qc.ageLimit, qc.categoryName
FROM quests q
LEFT JOIN adress a ON q.adress_id = a.ID
LEFT JOIN questcategory qc ON q.questCategory_id = qc.ID";

if (isset($_GET['search'])) {
$search = mysqli_real_escape_string($conn, $_GET['search']);
$query .= " WHERE q.name LIKE '%$search%'";
}

$result = mysqli_query($conn, $query);

?>

```

```

<div class='container my-4'>
  <div class='row'>
    <?php while ($row = mysqli_fetch_assoc($result)): ?>
      <div class='col-md-4 mb-4'>
        <div class='card shadow' style='overflow: hidden; transition: filter 0.5s ease; filter:
brightness(95%);' onmouseover='this.style.filter="brightness(100%)"'
onmouseout='this.style.filter="brightness(95%)"' data-category='<?php echo
htmlspecialchars($row['categoryName']); ?>' data-age-limit='<?php echo
htmlspecialchars($row['ageLimit']); ?>' data-people-amount='<?php echo
htmlspecialchars($row['peopleAmount']); ?>'>
          <img src='<?php echo htmlspecialchars($row['photoPath']); ?>' class='card-img-top'
style='height: 250px; object-fit: cover;' alt='photo of <?php echo htmlspecialchars($row['name']); ?>'>
          <div class='card-body'>
            <div class='mb-2'>
              <span class='badge bg-secondary'><?php echo htmlspecialchars($row['ageLimit']);
?></span>
            </div>
            <h5 class='card-title'><?php echo htmlspecialchars($row['name']); ?></h5>
            <p class='card-text'><small><?php echo htmlspecialchars($row['peopleAmount']);
?></small></p>
          </div>
          <a href='../room/quest_info.php?ID=<?php echo htmlspecialchars($row['ID']); ?>'
class='stretched-link'></a>
        </div>
      </div>
    <?php endwhile; ?>
  </div>
</div>

<script>
function filterCards() {
  var category = document.getElementById("category").value;
  var ageLimit = document.getElementById("ageLimit").value;
  var peopleAmount = document.getElementById("peopleAmount").value;

  var cards = document.getElementsByClassName("card");

  for (var i = 0; i < cards.length; i++) {
    var card = cards[i];
    var cardCategory = card.getAttribute("data-category");
    var cardAgeLimit = parseInt(card.getAttribute("data-age-limit"));
    var cardPeopleAmount = card.getAttribute("data-people-amount");

    if (
      (category === "" || category === cardCategory) &&
      (ageLimit === "" || parseInt(ageLimit) >= cardAgeLimit) &&
      (peopleAmount === "" || peopleAmount === cardPeopleAmount)
    ) {
      card.parentElement.style.display = "block";
    } else {
      card.parentElement.style.display = "none";
    }
  }
}
</script>

<?php
include_once '../includes/footer.php';
?>

```



## //резервацию извешдоšana

```
<?php
session_start();
include '../includes/dbcon.php';
require '../vendor/autoload.php'; // Ensure this is the correct path to the autoload file

use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;
use Mpdf\Mpdf;
use Stripe\Stripe;
use Stripe\Charge;

if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

if (!isset($_SESSION['user_id'])) {
    die("Please <a href='../register_login/loginform.php'>log in as user</a> to make a reservation. " .
    mysqli_connect_error());
}

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $room_id = $_POST['quest_id'];
    $date = $_POST['date'];
    $time = $_POST['time'];
    $discount = $_POST['discount'];
    $payment = $_POST['payment_method'];
    $cost = $_POST['cost'];
    $user_id = $_SESSION['user_id'];

    // Check if a reservation already exists for the given date and time
    $query = "SELECT * FROM reservation WHERE room_id='$room_id' AND date='$date' AND
    time='$time'";
    $result = mysqli_query($conn, $query);

    if (mysqli_num_rows($result) > 0) {
        echo "A reservation already exists for this date and time. Please choose another date/time.";
        echo "<a href='../room/quest_list.php'>Back to home page</a>";
        exit();
    }

    if ($payment === 'card') {
        $stripeToken = isset($_POST['stripeToken']) ? $_POST['stripeToken'] : null;

        if (!$stripeToken) {
            echo "Payment failed: Must provide source or customer.";
            exit();
        }

        // Set your secret key. Remember to switch to your live secret key in production!
        Stripe::setApiKey('your_secret_key'); // замените 'your_secret_key' на ваш секретный ключ

        try {
            $charge = Charge::create([
                'amount' => $cost * 100, // Stripe принимает суммы в центах
                'currency' => 'eur',
                'description' => 'Payment for reservation',
                'source' => $stripeToken,
            ]);
        }
    }
}
```

```

// Save card details if checkbox is checked
if (isset($_POST['save-card']) && $_POST['save-card'] == 'on') {
    $cardDate = $_POST['cardDate'];
    $cardNumber = $_POST['cardNumber'];
    $cardName = $_POST['cardName'];

    $stmt = $conn->prepare("INSERT INTO card (user_id, cardDate, cardNumber, cardName)
VALUES (?, ?, ?, ?)");
    $stmt->bind_param("iss", $user_id, $cardDate, $cardNumber, $cardName);
    $stmt->execute();
    $stmt->close();
}

} catch (\Stripe\Exception\CardException $e) {
    echo 'Payment failed: ' . $e->getError()->message;
    exit();
}

}

// Insert reservation into the database
$insert_query = "INSERT INTO reservation (date, time, cost, payment, room_id, client_id,
creation_date)
VALUES ('$date', '$time', '$cost', '$payment', '$room_id', '$user_id', CURDATE())";
mysqli_query($conn, $insert_query);

// Fetch user details
$user_query = "SELECT name, surname, email FROM users WHERE id='$user_id'";
$user_result = mysqli_query($conn, $user_query);
$user_row = mysqli_fetch_assoc($user_result);

// Fetch quest room and address details using JOIN
$quest_query = "
SELECT q.name AS quest_name, a.buildingAddress AS quest_address
FROM quests q
JOIN address a ON q.adress_id = a.id
WHERE q.id='$room_id'";
$quest_result = mysqli_query($conn, $quest_query);
$quest_row = mysqli_fetch_assoc($quest_result);

$user_name = $user_row['name'];
$user_surname = $user_row['surname'];
$user_email = $user_row['email'];
$quest_name = $quest_row['quest_name'];
$quest_address = $quest_row['quest_address'];

// Ensure the receipts directory exists
$receipts_dir = './receipts/';
if (!is_dir($receipts_dir)) {
    mkdir($receipts_dir, 0777, true);
}

// Generate PDF receipt using MPDF
$mpdf = new Mpdf();
$receipt_content = "
<style>
    body { font-family: Arial, sans-serif; }
    .invoice-box { max-width: 800px; margin: auto; padding: 30px; border: 1px solid #eee; box-
shadow: 0 0 10px rgba(0, 0, 0, 0.15); }
    .invoice-box table { width: 100%; line-height: inherit; text-align: left; border-collapse: collapse; }
    .invoice-box table td { padding: 5px; vertical-align: top; }

```

```

.invoice-box table tr td:nth-child(2) { text-align: right; }
.invoice-box table tr.top table td { padding-bottom: 20px; }
.invoice-box table tr.information table td { padding-bottom: 40px; }
.invoice-box table tr.heading td { background: #eee; border-bottom: 1px solid #ddd; font-
weight: bold; }
.invoice-box table tr.details td { padding-bottom: 20px; }
.invoice-box table tr.item td { border-bottom: 1px solid #eee; }
.invoice-box table tr.item.last td { border-bottom: none; }
.invoice-box table tr.total td:nth-child(2) { border-top: 2px solid #eee; font-weight: bold; }
</style>
<div class='invoice-box'>
  <table cellpadding='0' cellspacing='0'>
    <tr class='top'>
      <td colspan='2'>
        <table>
          <tr>
            <td class='title'>
              <h2>Quest Room Reservation</h2>
            </td>
            <td>
              Invoice #: " . uniqid() . "<br>
              Created: " . date('Y-m-d') . "<br>
              Due: " . date('Y-m-d', strtotime('+30 days')) . "
            </td>
          </tr>
        </table>
      </td>
    </tr>
    <tr class='information'>
      <td colspan='2'>
        <table>
          <tr>
            <td>
              Kirill Quest Room<br>
              1234 Main St<br>
              Anytown, CA 12345
            </td>
            <td>
              $user_name $user_surname<br>
              $user_email
            </td>
          </tr>
        </table>
      </td>
    </tr>
    <tr class='heading'>
      <td>
        Payment Method
      </td>
      <td>
        $payment
      </td>
    </tr>
    <tr class='details'>
      <td>
        Payment Method
      </td>
      <td>
        $payment
      </td>
    </tr>
    <tr class='heading'>

```

```

        <td>
            Item
        </td>
        <td>
            Price
        </td>
    </tr>
    <tr class='item'>
        <td>
            Quest Room Reservation - $quest_name
        </td>
        <td>
            $$cost
        </td>
    </tr>
    <tr class='item'>
        <td>
            Date and Time
        </td>
        <td>
            $date at $time
        </td>
    </tr>
    <tr class='item last'>
        <td>
            Location
        </td>
        <td>
            $quest_address
        </td>
    </tr>
    <tr class='total'>
        <td></td>
        <td>
            Total: $$cost
        </td>
    </tr>
</table>
</div>
";
";

$mpdf->WriteHTML($receipt_content);
$pdf_path = $receipts_dir . uniqid('receipt_', true) . '.pdf';
$mpdf->Output($pdf_path, 'F'); // Save PDF to a file

// Send email with receipt as attachment using PHPMailer
$mail = new PHPMailer(true);
try {
    $mail->isSMTP();
    $mail->Host = 'smtp.gmail.com';
    $mail->SMTPAuth = true;
    $mail->Username = 'kirillquestroom@gmail.com';
    $mail->Password = 'tdiscwhdffittzmz';
    $mail->SMTPSecure = 'tls';
    $mail->Port = 587;

    $mail->setFrom('kirillquestroom@gmail.com', 'KirillQuestRoom');
    $mail->addAddress($user_email, $user_name);

    $mail->isHTML(true); // Set email format to HTML
    $mail->Subject = 'Reservation Successful';
    $mail->Body = "Hello, $user_name $user_surname!<br><br>

```

Your reservation has been successfully made for the escape room \"\$quest\_name\"  
 at \$time on \$date, located at: \$quest\_address.<br><br>  
 Please find your official invoice attached.<br><br>  
 Best regards, Team Kirill Quest Room.";

```
// Attach PDF
$mail->addAttachment($pdf_path);

$mail->send();
echo 'Reservation successful. Confirmation email sent.';
} catch (Exception $e) {
    echo "Reservation successful, but email could not be sent. Mailer Error: {$mail->ErrorInfo}";
}

// Redirect to the home page or confirmation page
header('Location: ../room/quest_list.php');
exit();
} else {
    echo "Invalid request method.";
}
?>
```

### // redīgēt rezervāciju

```
<?php
session_start();

include '../includes/dbcon.php';

if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {

    $room_id = $_POST['room_id'];
    $date = $_POST['date'];
    $time = $_POST['time'];
    $cost = $_POST['cost'];
    $reserv_id = $_POST['reserv_id'];

    // Check if a reservation already exists for the given date and time
    $query = "SELECT * FROM reservation WHERE room_id='$room_id' AND date='$date' AND
time='$time'";
    $result = mysqli_query($conn, $query);

    if (mysqli_num_rows($result) > 0) {
        echo "A reservation already exists for this date and time. Please choose another date/time.";
        echo "<a href='../profile/profile_info.php'>Back to info page</a>";
        exit();
    }

    // Check if the logged-in user is an admin or a regular user
    $query = "UPDATE reservation
SET date = '$date', time = '$time', cost = '$cost'
WHERE ID = '$reserv_id'";

    if (mysqli_query($conn, $query)) {
        echo "Reservation updated successfully!";
    }
}
```

```

        echo "<a href='../profile/profile_info.php'>Back to info page</a>";
        header("Location: ../profile/profile_info.php");
    } else {
        echo "Error saving reservation: " . mysqli_error($conn);
    }
}

```

```

// Close database connection
mysqli_close($conn);
?>

```

#### // parbaude uz rezervāciju kalendārā

```

<?php
include '../includes/dbcon.php'; // Подключение к базе данных

if(isset($_GET['date']) && isset($_GET['quest_id'])) {
    $date = $_GET['date'];
    $quest_id = $_GET['quest_id'];
    $query = "SELECT time FROM reservation WHERE date = '$date' AND room_id = '$quest_id'";
    $result = mysqli_query($conn, $query);
    if (!$result) {
        http_response_code(500); // Внутренняя ошибка сервера
        echo json_encode(['error' => 'Database query failed']);
        exit;
    }

    $bookedTimes = [];
    while($row = mysqli_fetch_assoc($result)) {
        $bookedTimes[] = $row['time']; // Убедитесь, что 'time' это правильное название столбца
    }
    echo json_encode($bookedTimes);
    mysqli_close($conn);
} else {
    http_response_code(400); // Ошибка запроса
    echo json_encode(['error' => 'Date or quest_id parameter missing']);
}
?>

```

#### // rediģēt kvestu istabu

```

<?php
$pageTitle = 'Quest Room update form';
include_once '../includes/header.php';
?>

<?php
// Check if the user is not an admin
if (!isset($_SESSION['admin_id']) || empty($_SESSION['admin_id'])) {
    // Redirect the user to the admin login page
    header("Location: ../register_login/loginform.php");
    exit();
}

include '../includes/dbcon.php';

if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}

// Check if form is submitted
if(isset($_POST['update_room'])){

```

```

// Get room ID
$room_id = $_POST['room_id'];

// Get form data
$name = mysqli_real_escape_string($conn, $_POST['name']);
$category = mysqli_real_escape_string($conn, $_POST['category']);
$address = mysqli_real_escape_string($conn, $_POST['address']);
$peopleAmount = mysqli_real_escape_string($conn, $_POST['peopleAmount']);
$ageLimit = mysqli_real_escape_string($conn, $_POST['ageLimit']);
$description = mysqli_real_escape_string($conn, $_POST['description']);

// Validate form data

if (empty($name) || empty($category) || empty($address) || empty($peopleAmount) ||
empty($ageLimit) || empty($description)) {
    echo "Please fill in all fields!";
    exit();
}

if (strlen($name) > 50 || strlen($category) > 30 || strlen($address) > 50 ||
strlen($peopleAmount) > 30 || strlen($ageLimit) > 30 || strlen($description) > 500) {
    echo "Too long text, maximum 30 symbols for the name, category, address, discount
and peopleAmount and maximum 500 symbols for the description!";
    exit();
}

if (!preg_match("/^[a-zA-Z ]*$/", $category)) {
    echo "Category should only contain letters and spaces!";
    exit();
}

if (!preg_match('/^([1-9])-([1-9])$/', $peopleAmount, $matches)) {
    echo "People Amount should be in the format 'x-y', where x and y are numbers greater than 0!";
    exit();
}

if ($matches[1] >= $matches[2]) {
    echo "The first number in People Amount should be less than the second number!";
    exit();
}

if (!is_numeric($ageLimit) || $ageLimit <= 0 || $ageLimit >= 99) {
    echo "Age Limit should be a number greater than 0 and less than 99!";
    exit();
}

function generateRandomFilename() {
    return uniqid() . '.jpg';
}

function saveImage($file, $oldFilePath) {
    $allowed_types = ['image/jpeg', 'image/png', 'image/gif', 'image/jpg'];
    $filetype = $file['type'];

    if (!in_array($filetype, $allowed_types)) {
        die('Invalid file type.');
```

```

    }

    $target_dir = '../images/';

    $filename = generateRandomFilename();

    while (file_exists($target_dir . $filename)) {
        $filename = generateRandomFilename();
    }

    $target_file = $target_dir . $filename;

    if (!empty($oldFilePath) && file_exists($oldFilePath)) {
        unlink($oldFilePath);
    }

    if (move_uploaded_file($file['tmp_name'], $target_file)) {
        return $target_file;
    } else {
        return null;
    }
}

if (isset($_FILES['photo']) && $_FILES['photo']['error'] === UPLOAD_ERR_OK) {
    $photoPath = saveImage($_FILES['photo'], $_POST['photoPath']);
    if ($photoPath === null) {
        die('Error uploading photo.');
```



```

        header("Location: ../room/quest_form.php");
        exit();
    }

    // Get current room data

    $id = $_POST['id'];
    $query = "SELECT q.ID, q.name, qc.categoryName, a.buildingAdress, q.peopleAmount, qc.ageLimit,
    q.description, q.photoPath
    FROM quests q
    LEFT JOIN questcategory qc ON q.questCategory_id = qc.ID
    LEFT JOIN address a ON q.adress_id = a.ID
    WHERE q.ID='$id'";
    $result = mysqli_query($conn, $query);

    // Check if room was found
    if(mysqli_num_rows($result) == 1){
        $row = mysqli_fetch_assoc($result);

        ?>

        <div class="container mt-5">
        <h2>Edit Room</h2>
        <form method="post" action="" enctype="multipart/form-data">
            <div class="form-group">
                <label for="name">Name:</label>
                <input type="text" class="form-control" name="name" value="<?php echo $row['name'];
?>">
            </div>
            <div class="form-group">
                <label for="category">Category:</label>
                <input type="text" class="form-control" name="category" value="<?php echo
$row['categoryName']; ?>">
            </div>
            <div class="form-group">
                <label for="address">Address:</label>
                <input type="text" class="form-control" name="address" value="<?php echo
$row['buildingAdress']; ?>">
            </div>
            <div class="form-group">
                <label for="peopleAmount">People Amount:</label>
                <input type="text" class="form-control" name="peopleAmount" value="<?php echo
$row['peopleAmount']; ?>">
            </div>
            <div class="form-group">
                <label for="ageLimit">Age Limit:</label>
                <input type="text" class="form-control" name="ageLimit" value="<?php echo
$row['ageLimit']; ?>">
            </div>
            <div class="form-group">
                <label for="description">Description:</label>
                <textarea class="form-control" name="description"><?php echo $row['description'];
?></textarea>
            </div>
            <div class="form-group">
                <label>Current photo:</label><br>
                
            </div>
            <div class="form-group">
                <label for="photo">New Photo:</label>
                <input type="file" class="form-control-file" name="photo">

```

```

        </div>
        <input type="hidden" name="photoPath" value="<?php echo $row['photoPath']; ?>">
        <input type="hidden" name="room_id" value="<?php echo $row['ID']; ?>">
        <button type="submit" class="btn btn-primary" name="update_room">Update</button>
    </form>
</div>

<?php
} else{
    echo "Room not found.";
}

// Close database connection
mysqli_close($conn);
?>

```

### // pdf failu ģenerēšana

```

<?php
include '../includes/dbcon.php';
require_once '../vendor/autoload.php';

$mpdf = new \Mpdf\Mpdf();

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Получаем выбранные месяц и год из POST-запроса
$month = isset($_POST['selectedMonth']) ? $_POST['selectedMonth'] : date('m');
$year = isset($_POST['selectedYear']) ? $_POST['selectedYear'] : date('Y');

// Company header and title
$html = '
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <style>
        body { font-family: Arial, sans-serif; }
        .header { text-align: center; margin-bottom: 20px; }
        .header img { max-width: 150px; }
        h1 { color: #333; }
        h2 { color: #555; margin-top: 40px; }
        table { width: 100%; border-collapse: collapse; margin-top: 20px; }
        th, td { border: 1px solid #ddd; padding: 10px; text-align: left; }
        th { background-color: #f2f2f2; }
        .total { font-weight: bold; }
    </style>
</head>
<body>
    <div class="header">
        
        <h1>Reservation Report</h1>
        <p>For the month of ' . $month . ' ' . $year . '</p>
    </div>
';

// First query - Total cost by date
$sql1 = "SELECT SUM(cost) as total_cost, DATE(date) as date
FROM reservation

```

```

        WHERE MONTH(date) = '$month' AND YEAR(date) = '$year'
        GROUP BY date";
$result1 = $conn->query($sql1);
$html .= '<h2>Total Cost by Date</h2>';
$html .= '<table><thead><tr><th>Date</th><th>Total Cost</th></tr></thead><tbody>';
$total_month_cost = 0;
while ($row = $result1->fetch_assoc()) {
    $html .= '<tr><td>' . $row['date'] . '</td><td>' . number_format($row['total_cost'], 2) . '</td></tr>';
    $total_month_cost += $row['total_cost'];
}
$html .= '<tr class="total"><td>Total</td><td>' . number_format($total_month_cost, 2) . '</td></tr>';
$html .= '</tbody></table>';

// Second query - Number of reservations by day of the week
$days_of_week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'];
$sql2 = "SELECT WEEKDAY(date) AS day_of_week, COUNT(*) AS reservation_count
        FROM reservation
        WHERE MONTH(date) = '$month' AND YEAR(date) = '$year'
        GROUP BY day_of_week";
$result2 = $conn->query($sql2);

$reservations_by_day = array_fill(0, 7, 0);
while ($row = $result2->fetch_assoc()) {
    $reservations_by_day[$row['day_of_week']] = $row['reservation_count'];
}

$html .= '<h2>Number of Reservations by Day of the Week</h2>';
$html .= '<table><thead><tr><th>Day of the Week</th><th>Number of
Reservations</th></tr></thead><tbody>';
foreach ($days_of_week as $index => $day_name) {
    $html .= '<tr><td>' . $day_name . '</td><td>' . $reservations_by_day[$index] . '</td></tr>';
}
$html .= '</tbody></table>';

// Third query - Visitors by rooms and quests
$sql3 = "SELECT q.name, COUNT(r.room_id) as visitors
        FROM reservation r
        JOIN quests q ON r.room_id = q.id
        WHERE MONTH(r.date) = '$month' AND YEAR(r.date) = '$year'
        GROUP BY q.name";
$result3 = $conn->query($sql3);
$html .= '<h2>Visitors by Rooms and Quests</h2>';
$html .= '<table><thead><tr><th>Quest</th><th>Visitors</th></tr></thead><tbody>';
while ($row = $result3->fetch_assoc()) {
    $html .= '<tr><td>' . $row['name'] . '</td><td>' . $row['visitors'] . '</td></tr>';
}
$html .= '</tbody></table>';

// Footer
$html .= '
    <p>Report generated on ' . date('Y-m-d') . '</p>
</body>
</html>';
$pdf->WriteHTML($html);
$pdf->Output('reservation_report_' . $month . '_' . $year . '.pdf', 'D');
// Close connection
$conn->close();
?>

```