Liepājas Valsts tehnikums

Mobilā aplikācija kinoteātrim “Filmu Nams”

Kvalifikācijas eksāmena praktiskās daļas dokumentācija

Darba autore:

Sofija Dišlovaja, 4PT-1

Darba vadītājs:

skolotājs, Raimonds Kristovskis

Eksāmena datums 2025. gada 16. jūnijs

Liepāja 2025

Saturs

Ievads

Mūsdienu pasaulē, kurā digitālās tehnoloģijas kļūst par neatņemamu mūsu ikdienas dzīves sastāvdaļu, kinoteātra mobilās lietotnes izveide ir ne tikai aktuāls, bet arī nepieciešams solis kinoteātra attīstībā, kas piedāvā izklaides pakalpojumus, piedāvājot gan ērtāku pieeju un papildinājumu pakalpojumiem, gan jaunas iespējas biznesa stratēģijas uzlabojumiem.

Šī projekta mērķis ir izstrādāt mobilo lietojumprogrammu kinoteātru tīklam, kas paredzēta, lai uzlabotu klientu apkalpošanas kvalitāti un optimizētu kinoteātra darba procesus.

Šīs tēmas izvēle ir balstīta uz vairākiem faktoriem. Pirmkārt, Latvijas tirgū ir ievērojams mobilā kino lietojumprogrammu trūkums. Neskatoties uz aktīvo digitālo tehnoloģiju attīstību, lielākā daļa kinoteātru apstājas ar pamata tīmekļa vietnēm, kas pilnībā neapmierina mūsdienu lietotāju ērtības un funkcionalitātes prasībām. Turklāt projekts piedāvā plašas iespējas potenciālajai sadarbībai ar lielākajiem tirgus dalībniekiem, piemēram, "Cinamon Kino" un citām kinoteātru ķēdēm. Tas rada reālas perspektīvas izstrādātā risinājuma praktiskai ieviešanai.

Izstrādājamā lietojumprogramma ir paredzēta, lai risinātu vairākas esošās problēmas un vajadzības. Pirmkārt, tā ir izstrādāta, lai nodrošināt ātru un ērtu piekļuvi kino pakalpojumiem jebkurā laikā un vietā. Otrkārt, lai nodrošināt lietotājiem ātrāko un ērtāko bezsaistes piekļuvi iegādātajām biļetēm, izmantojot aplikācijas logrīku, kuru būs iespējams pievienot mobilas ierīces sākuma ekrānā, kas ievērojami palielina pakalpojuma lietošanas ērtumu.

Treškārt, izstrādātā aplikācija piedāvās kinoteātrim papildu konkurences priekšrocību, nodrošinot lielāko zīmolu atpazīstamību, atšķirību no konkurentiem un uzlabojot klientu apkalpošanas kvalitāti.

Projektam ir ievērojams potenciāls turpmākai attīstībai un paplašināšanai. Galvenā attīstības joma ir ekskluzīvu piedāvājumu sistēmas ieviešana aplikācijas lietotājiem, kas ietver:

- īpašas akcijas un atlaides aplikācijas lietotājiem,
- priekšlaicīgu piekļuvi biļetēm uz pirmizrādēm,
- personalizētus ieteikumus, pamatojoties uz skatīšanās vēsturi,
- lojalitātes programmu ar papildu privilēģijām.

Šo funkciju ieviešana ne tikai palielinās lietotāju pievilcību aplikācijai, bet arī radīs papildus iespējas palielināt kinoteātra peļņu, palielinot apmeklējumu biežumu, klientu apkalpošanas kvalitātes uzlabojuma dēļ.

Tādējādi mobilās aplikācijas izstrāde kinoteātrim ir aktuāls un daudzsološs projekts, kam ir gan praktiska biznesa vērtība, gan ievērojams attīstības potenciāls.

Uzdevuma formulējums

1.1. Galvenais mērķis

Uzdevuma mērķis ir funkcionālas un intuitīvas kinoteātra mobilās lietotnes izstrāde, kas ļaus lietotājiem viegli apskatīt informāciju par kinofilmām, kā arī iegādāties biļetes. Lietotnei jānodrošina augstas kvalitātes lietotāja pieredze un visu komponentu uzticamība.

1.2. Projekta izstrādes plāns

Darbs pie projekta sāksies ar detalizēta lietojumprogrammas dizaina un prototipa izveidi programmā Figma. Šajā posmā tiks izstrādāti visi galvenie lietojumprogrammas ekrāni un izstrādāti lietotāju scenāriji, kas ļaus vizualizēt nākamo produktu un novērtēt tā lietojamību. Paralēli tam tiks veikts darbs pie lietojumprogrammas iekšējās struktūras projektēšanas, izveidojot nepieciešamās UML diagrammas, kas atspoguļo sistēmas struktūru un uzvedību.

Kad sagatavošanās posms būs pabeigts, darbs pāries uz tehnisko pusi. Pirmais solis būs projekta inicializācija un repozitorija izveide GitHub platformā, kas nodrošinās izstrādes procesa organizāciju un versiju kontroli.

Nākamais solis būs izstrādāt lietojumprogrammas pamatfunkcijas, tostarp divus galvenos komponentus: iespēju pārskatīt filmu sarakstu un biļešu tirdzniecības funkcionalitāti. Īpaša uzmanība tiks pievērsta integrācijai ar datubāzi, kurā tiks glabāta un apstrādāta informācija par filmām, izrādēm un rezervācijām.

Liela daļa darba tiks veltīta lietotāja saskarnes izstrādei. Šis process ietver ne tikai vizuālās komponentes īstenošanu, bet arī lietojumprogrammas lietojamības nodrošināšanu.

Būtisks izstrādes aspekts ir lietojumprogrammas integrācija ar dažādiem ārējiem pakalpojumiem. Tas ietver ne tikai maksājumu sistēmas, bet arī analītikas pakalpojumus, paziņojumu sistēmas, sociālos tīklus satura koplietošanai un citus trešo pušu API, kas var uzlabot lietojumprogrammas funkcionalitāti un lietotāja pieredzi.

Visām izstrādātajām sastāvdaļām tiks veikta rūpīga testēšana, tostarp gan koda testēšana, gan testēšana uz fiziskām ierīcēm, lai noteiktu un novērstu iespējamās veiktspējas un savietojamības problēmas.

Projekta noslēguma posmā, pamatojoties uz testēšanas rezultātiem, plānots uzlabot lietojumprogrammas dizainu un funkcionalitāti. Īpaša uzmanība tiks pievērsta veiktspējas optimizācijai un lietotāju pieredzes uzlabošanai.

1.3. Iespējamās problēmas

Paredzams, ka izstrādes procesā būs jāsaskaras ar vairākām tehniskām un organizatoriskām problēmām. Tehniskās problēmas var ietvert nepieciešamību nodrošināt uzticamu integrāciju ar maksājumu sistēmām un stabilu lietojumprogrammas darbību.

Organizatoriskās problēmas var ietvert nepieciešamību koordinēt dažādus izstrādes aspektus un ievērot termiņus. Atsevišķs izaicinājums būs nodrošināt lietotāju datu drošību un izveidot patiesi intuitīvu saskarni.

Projekta veiksme tiks vērtēta, pamatojoties uz vairākiem kritērijiem: galveno funkciju stabilitāti, lietotāja pieredzes kvalitāti, lietojumprogrammas veiktspēju un atbilstību mūsdienu mobilo lietotņu izstrādes standartiem. Īpaša uzmanība testēšanas laikā tiks pievērsta lietotāju atsauksmēm, jo viņi ir produkta galalietotāji un viņu apmierinātība ir galvenais projekta veiksmes rādītājs.

2. Programmatūras prasību specifikācija

2.1. Produkta perspektīva

Mobilai lietotnei kinoteātrim ir perspektīvas, galvenokārt, piedāvājot kinoteātrim jaunās iespējas, lai pievērstu jauno klientu uzmanību, izceltu uzņēmumu starp konkurentiem, kā arī veiktu uzlabojumus klientu apkalpošanas jomā.

Izmantotie rīki atļauj produktu pārnest uz daudzām citam platformām veidojot minimālas izmaiņas sākotnēja kodā, t.i. veidojot aplikāciju primāri balstoties uz mobilām ierīcēm, izmantojot "Flutter" ietvaru kā lietotnes pamatu, to var arī pielāgot tādām platformām kā Windows, OSX utt.

2.2. Sistēmas funkcionālās prasības

Šī nodaļā būs aprakstītas visas funkcionālās prasības kinoteātra lietotnes sistēmai. Aprakstītas funkcionālās prasības pieder gan klientu aplikācijai, gan administrācijas paneļa aplikācijai.

2.2.1. Klientu autorizācija sistēmā, izmantojot e-pastu un paroli

Mērķis:

Funkcija nepieciešama, lai kinoteātra klienti varētu ielogoties aplikācijā un lietot to

Ievaddati:

1.tabula

Klientu autorizācijas dati, ielogošana ar e-pastu

Nosaukums	Obligāts	Piezīmes
E-pasts	Jā	
Parole	Jā	Uzglabājas šifrētā formā

Apstrāde:

Nospiežot autorizēšanās pogu, sistēma pārbauda vai visi ievadlauki ir aizpildīti (Skat. 1. tabulu). Ja tie ir, tad tālāk servera pusē tiek pārbaudīti ievadītie dati ar datu bāzi, un ja ievadītie dati ir korekti - lietotājs tiek autorizēts.

Izvaddati:

1) Nepieciešamība aizpildīt visus ievadlaukus.
2) Paziņojums par nepareizu lietotājvārdu vai paroli.
3) Autorizācija sistēmā.

2.2.2. Klientu autorizācija sistēmā, izmantojot Google kontu

Mērķis:

Funkcija nepieciešama, lai kinoteātra klienti varētu ielogoties aplikācijā, izmantojot Google kontu

Apstrāde:

Nospiežot Google autorizēšanos pogu, lietotājs tiek pārsūtīts uz Google kontu sistēmu, kur lietotājam būs iespēja izvēlēties vienu no esošam kontiem ierīcē, vai ielogoties Google kontā, izmantojot e-pastu un paroli. Ja ir izvēlēts otrais variants - sistēma pārbauda vai visi ievadlauki ir aizpildīti. Ja tie ir, tad tālāk servera pusē tiek pārbaudīti ievadītie dati ar datu bāzi, un ja ievadītie dati ir korekti - lietotājs tiek autorizēts.

Izvaddati:

1) Saņemtas kļūdas izvadīšana, ja autorizācija nav veiksmīga;
2) Autorizācija sistēmā.

2.2.3. Klientu autorizācija sistēmā, izmantojot Facebook kontu

Mērķis:

Funkcija nepieciešama, lai kinoteātra klienti varētu ielogoties aplikācijā, izmantojot Facebook kontu

Apstrāde:

Nospiežot Facebook autorizēšanos pogu, lietotājs tiek pārsūtīts uz Facebook kontu sistēmu, kur lietotājam būs iespēja izvēlēties vienu no esošam kontiem ierīcē, vai ielogoties Facebook kontā, izmantojot e-pastu un paroli.

Izvaddati:

1) Saņemtas kļūdas izvadīšana, ja autorizācija nav veiksmīga;
2) Autorizācija sistēmā.

2.2.4. Klientu reģistrācijas sistēmā

Mērķis:

Funkcija nepieciešama, lai kinoteātra klienti varētu reģistrēties aplikācijā, lai turpmāk ielogoties un izmantot to

Ievaddati:

2.tabula

Klientu reģistrācijas dati

Nosaukums	Obligāts	Piezīmes
E-pasts	Jā	Jābūt unikālam
Parole	Jā	Uzglabājas šifrētā formā
Parole atkārtoti	Jā	Nepieciešams, lai klients varētu pārliecināties par paroles pareizrakstību

Apstrāde:

Nospiežot reģistrācijas pogu, sistēma pārbauda vai visi ievadlauki ir aizpildīti (Skat. 2. tabulu). Ja tie ir, tad tālāk klients tiek pārsūtīts uz nākamo reģistrācijas daļu, kurā būs jāpievieno vārds un, pēc vēlmes, bilde, lai varētu pabeigt reģistrāciju.

Izvaddati:

1) Nepieciešamība aizpildīt visus ievadlaukus;
2) Paziņojums par atkārtotas paroles nesakrišanu ar paroli;
3) Lietotāja autorizācija sistēmā.

2.2.5. Navigācija aplikācijā

Mērķis:

Funkcija nepieciešama, lai iedot lietotājam iespēju pārskatīt aplikācijā pieejamās sadaļas un funkcionalitātes.

Ievaddati:

Lietotājs nospiež uz attiecīgās navigācijas pogas.

Apstrāde:

Nospiežot kādu no navigācijas pogām, tiek atvērta attiecīgā sadaļa: piedāvājumi, saraksts, galvenā sadaļa, paziņojumi vai profils.

Izvaddati:

Attiecīgās sadaļas attēlošana.

2.2.6. Galvenās sadaļas attēlošana

Mērķis:

Funkcija dot iespēju nolasīt informāciju par jaunākām filmām un piedāvājumiem no datu bāzes.

Apstrāde:

Atvērot galveno sadaļu tiek attēlota “galerija” ar jaunākam filmām un piedāvājumiem.

Izvaddati:

“Galerijas” attēlošana galvenā sadaļā ar jaunākam filmām un piedāvājumiem no datu bāzes.

2.2.7. Galvenās sadaļas galerijas vienumu detalizēta skata attēlošana

Mērķis:

Ja kādam no galerijas vienumiem ir saite ar kādu no piedāvājumiem vai filmām, lietotājam ir jābūt iespējams atvērt detalizētu skatu.

Ievaddati:

Lietotājs pārvelc ar pirkstu no lejas uz augšu.

Apstrāde:

Ja vienumam ir pieejama saite ar kādu no piedāvājumiem vai filmām, tiek atvērts filmas vai piedāvājuma detalizēts skats.

Izvaddati:

1) Tiek attēlots filmas skats;
2) Tiek attēlots piedāvājuma skats;
3) Nekas netiek attēlots, ja vienumam nav nekādas saites.

2.2.8. Seansu saraksta attēlošana

Mērķis:

Funkcija dot iespēju pārskatīt filmu sarakstu uz attiecīgo datumu filmu saraksta sadaļā.

Ievaddati:

Lietotājs nospiež uz attiecīgas datumu pogas vai uz pogām “Šodien”, “Rīt” vai “Parīt”. Pēc noklusējuma, atvērot sadaļu, ir izvēlēts šodienas datums.

Apstrāde:

Nospiežot attiecīgo datumu pogu, tiek paradīti dati par izvēlētas dienas sarakstu no datu bāzes.

Izvaddati:

Filmu saraksta attēlošana izvēloties attiecīgo datumu filmu saraksta sadaļā.

2.2.9. Filmu saraksta attēlošana

Mērķis:

Funkcija dot iespēju lietotājiem apskatīt visas filmas no datu bāzes, kuras nav paslēptas.

Ievaddati:

Saraksta sadaļā, lietotājs nospiež uz pogas “Filmas”, kas maina skatu uz visu pieejamo filmu sarakstu.

Apstrāde:

Nospiežot uz pogas “Filmas” tiek rādīts saraksts ar visām, klientiem pieejamām, filmām.

Izvaddati:

Filmu saraksta attēlošana.

2.2.10. Biļešu pirkšana

Mērķis:

Funkcija dot iespēju lietotājiem izvēlēties un nopirkt vienu vai vairāk biļetes kādam no filmas seansiem, filmas detalizētā skatā. Pirkšanas forma (Skat. 3. tabulu) ir pieejama, tikai ja filmai ir pieejami seansi nākotnē.

Ievaddati:

3.tabula

Biļešu pirkšanas dati

Nosaukums	Obligāts	Piezīmes
Datums	Jā	Lietotājs izvēlas vienu no datumiem, kur ir pieejams izvēlētas filmas seanss. Datumi, kur nav neviena filmas seansa, nav pieejami izvēlei.
Laiks	Jā	Izvēloties datumu, tiek piedāvāti laiki, kuri norāda uz pieejamiem seansiem.
Vietas	Jā	Lietotājs izvēlas vienu vai vairāk sēdvietas zālē. Izvēlēto sēdvietu skaits ir biļešu skaits.
Promokods	Nē	Lietotājam ir iespēja ievadīt promokodu, kas aktivizēs atlaidi. Ir iespējams ievadīt tikai 1 promokodu.

Apstrāde:

Izvēloties seansu un sēdvietas, lietotājs nospiež uz pogas "Apmaksāt" un tiek radīts Stripe maksāšanas logs, kur, veicot maksājumu, tas tiek apstrādāts. Veiksmīga maksājuma gadījumā, biļetes un maksājuma vēsture tiek saglabātas datubāzē.

Izvaddati:

1) Paziņojums par obligāto lauku aizpildīšanu;
2) Paziņojums par nederīgo vai neeksistējošo promokodu;
3) Paziņojums par veiksmīgo maksājumu;
4) Paziņojums par neveiksmīgo maksājumu.

2.2.11. Piedāvājumu attēlošana piedāvājumu sadaļā

Mērķis:

Funkcija dot iespēju lietotājiem redzēt piedāvājumus attiecīgajā sadaļā.

Apstrāde:

Atverot piedāvājumu sadaļu, tiek attēloti piedāvājumi no datu bāzes.

Izvaddati:

Tiek radīta sadaļa ar piedāvājumiem.

2.2.12. Piedāvājumu detalizēta attēlošana

Mērķis:

Funkcija dot iespēju pārskatīt konkrēta piedāvājuma pilno saturu.

Ievaddati:

Lietotājs nospiež uz piedāvājuma.

Apstrāde:

Nospiežot uz piedāvājuma, tiek atvērts tā pilnais skats, lai pilnībā ietilpst savu saturu, kā arī promokodu, ja tās bijis piesaistīts.

Izvaddati:

Piedāvājuma skatā paradās pilns apraksts ar promokodu, ja tās ir piesaistīts.

2.2.13. Piedāvājumu promokoda kopēšana

Mērķis:

Funkcija dot iespēju lietotājiem nokopēt promokodu, nospiežot uz attiecīgas pogas konkrēta piedāvājumu skatā. Šī funkcija atvieglo promokodu izmantošanu un ievadīšanu.

Ievaddati:

Lietotājs nospiež uz pogas “Kopēt” blakus promokodam.

Apstrāde:

Nospiežot uz pogas ”Kopēt”, promokods tiek saglabāts ierīces starpliktuvī, un rāda paziņojumu par darbību.

Izvaddati:

1) Paziņojums par neveiksmīgu darbību;
2) Paziņojums par veiksmīgu darbību.

2.2.14. Paziņojumu attēlošana paziņojumu sadaļā

Mērķis:

Funkcija dot iespēju lietotājiem redzēt paziņojumus attiecīgajā sadaļā.

Apstrāde:

Atvērot paziņojumi sadaļu, tiek attēloti paziņojumi no datu bāzes.

Izvaddati:

Tiek radīta sadaļa ar paziņojumiem.

2.2.15. Paziņojumu detalizēta attēlošana

Mērķis:

Funkcija dot iespēju pārskatīt konkrēta paziņojuma pilno saturu, kā arī piekļūt tādām pogām kā “Atzīmēt kā izlasīto” un “Dzēst”, kuras piedāvā attiecīgas funkcijas.

Ievaddati:

Lietotājs nospiež uz paziņojuma.

Apstrāde:

Nospiežot uz saīsināta paziņojuma, tās palielinās, lai pilnībā ietilpst savu saturu.

Izvaddati:

Paziņojumā paradās pilns apraksts un divas pogas “Atzīmēt kā izlasītu” un “Dzēst”.

2.2.16. Paziņojumu atzīmēšana kā izlasītu

Mērķis:

Funkcija dot iespēju lietotājam atzīmēt paziņojumu kā izlasītu, lai mainīt izskatu un turpmāk atšķirt no jauniem paziņojumiem.

Ievaddati:

Lietotājs nospiež uz paziņojuma, palielinot to, un pēc tām nospiež uz pogas “Atzīmēt kā lasīto”.

Apstrāde:

Nospiežot uz pogas “Atzīmēt kā izlasīto” paplašinātā paziņojumā, paziņojums maina krāsu. Poga maina nosaukumu uz “Atzīmēt kā neizlasīto”.

Izvaddati:

Paziņojumam mainās krāsa, lai lietotājs varētu atšķirt lasītus paziņojumus.

2.2.17. Paziņojumu dzēšana

Mērķis:

Funkcija dot iespēju lietotājam dzēst paziņojumu. Datu bāzē, paziņojums paliek, bet tai būs piešķirts statuss “dzēsts”.

Ievaddati:

Lietotājs nospiež uz paziņojuma, palielinot to, un pēc tām nospiež uz pogas “Dzēst”.

Apstrāde:

Nospiežot uz pogas “Dzēst” paplašinātā paziņojumā, tiek paradīts “modal” logs ar iespējam atcelt darbību un apstiprināt dzēšanu. Ja lietotājs atcel dzēšanu, apstiprināšanas logs pazūd un paziņojums netiek dzēsts. Ja lietotājs apstiprina dzēšanu, paradās paziņojums par darbības rezultātu un paziņojums tiek dzēsts.

Izvaddati:

1) Apstiprināšanas logs;
2) Paziņojums par veiksmīgu dzēšanu;
3) Paziņojums par neveiksmīgu dzēšanu;
4) Dzēsts paziņojums pazūd no saraksta.

2.2.18. Maksājumu vēstures attēlošana

Mērķis:

Funkcija dot iespēju lietotājam apskatīties maksājumu vēsturi, kas glabājas datu bāzē.

Ievaddati:

Lietotājs pāriet uz maksājumu sadaļu no profila sadaļas.

Apstrāde:

Nospiežot attiecīgo pogu profila sadaļā tiek atvērta maksājumu sadaļa.

Izvaddati:

Maksājumu vēstures ierakstu attēlošana.

2.2.19. Maksājuma datu kopēšana

Mērķis:

Funkcija dot iespēju lietotājam nokopēt maksājuma ID vai seansa ID mobilās ierīces starpliktuvē. Tas varētu būt nepieciešams lietotāju saziņai ar administrāciju, jautājuma vai problēmas gadījumā saistībā ar maksājumu.

Ievaddati:

Lietotājs nospiež uz attiecīgas pogas blakus ID.

Apstrāde:

Nospiežot kopēšanas pogu, attiecīgais ID tiek saglabāts mobilās ierīces starpliktuvē, kā arī tiek paradīts paziņojums par darbību.

Izvaddati:

1) Paziņojums par veiksmīgo darbību;
2) Paziņojums par neveiksmīgo darbību.

2.2.20. Lietotāja biļešu attēlošana

Mērķis:

Funkcija dot iespēju pāriet no profila sadaļas uz biļešu sadaļu un piekļūt nopirkto biļešu sarakstām.

Ievaddati:

Lietotājs nospiež uz biļešu sadaļas pogas profila sadaļā.

Apstrāde:

Nospiežot attiecīgo pogu profila sadaļā tiek atvērta biļešu sadaļa.

Izvaddati:

Biļešu saraksta attēlošana.

2.2.21. Biļešu detalizēts skats ar QR koda ģenerāciju

Mērķis:

Funkcija dot iespēju lietotājiem apskatīties datus par biļeti detalizēti, kā arī redzēt biļešu QR kodu, lai noradīt to kinoteātra darbiniekam.

Ievaddati:

Lietotājs nospiež uz biļetes biļešu sarakstā.

Apstrāde:

Nospiežot uz biļetes, tiek atvērts detalizēts skats ar biļetes datiem un QR kodu.

Izvaddati:

1) Biļetes dati (seanss, datums, filma, utt.);
2) Biļetes statuss (izmantots, beidzies termiņš, nav izmantots);
3) Biļetes QR kods.

2.2.22. Profila detalizētais skats

Mērķis:

Funkcija dot iespēju pāriet no profila sadaļas uz profila detalizētu skatu, kur ir iespējams mainīt vārdu, bildi vai e-pastu.

Ievaddati:

Lietotājs nospiež uz attiecīgas pogas profila sadaļā.

Apstrāde:

Nospiežot attiecīgo pogu profila sadaļā tiek atvērta profila detaļu skats.

Izvaddati:

Lietotāja vārda, attēla un e-pasta attēlošana rediģēšanas.

2.2.23. Lietotāja bildes maiņa

Mērķis:

Funkcija dot iespēju lietotājam mainīt bildi savam profilam, profila detaļu sadaļā.

Ievaddati:

Nospiežot uz lietotāja bildes profila detaļu sadaļā, tiek atvērta attēlu izvēlē aplikācijā pēc iestatījumiem. Lietotājs var izvēlēties bildi, vai atcelt darbību. Kā bildi drīkst izvēlēties tikai attēlu ņemot vērā faila izmēru ierobežojumu.

Apstrāde:

Nospiežot uz lietotāja bildes, atvērsies ierīces attēlu aplikācija pēc iestatījumiem, kurā lietotājs izvēlēsies attēlu vai atcels darbību. Pēc attēla izvēles bilde tiek saglabāta serverī un datu bāzē saglabāsies attiecīga URL adrese.

Izvaddati:

1) Paziņojums par veiksmīgo bildes maiņu;
2) Paziņojums par neveiksmīgo bildes maiņu validācijas kļūdas dēļ;
3) Paziņojums par neveiksmīgo bildes maiņu darbības atcelšanas dēļ;
4) Paziņojums par neveiksmīgo vārdu maiņu servera kļūdas dēļ.

2.2.24. Lietotāja vārda maiņa

Mērķis:

Funkcija dot iespēju lietotājam mainīt vārdu savam profilam, profila detaļu sadaļā.

Ievaddati:

4.tabula

Lietotāja vārda maiņas dati

Nosaukums	Obligāts	Piezīmes
Vārds	Jā	Ievaddati ir ierobežoti līdz 25 simboliem un nedrīkst būt tukši. No sākuma, lauks ir aizpildīts ar eksistējošo vārdu.

Apstrāde:

Nospiežot apstiprināšanas pogu, tiek pārbaudīts laukā ievadītais vārds (Skat. 4. tabulu). Ja ievadītais vārds trūkst, tiek radīts paziņojums par validācijas kļūdu. Ja vārds ir ievadīts, tās tiek saglabāts datu bāzē, mainot attiecīgajām lietotājam vārdu uz ievadīto.

Izvaddati:

5) Paziņojums par veiksmīgo vārdu maiņu;
6) Paziņojums par neveiksmīgo vārdu maiņu validācijas kļūdas dēļ;
7) Paziņojums par neveiksmīgo vārdu maiņu servera kļūdas dēļ.

2.2.25. Lietotāja e-pasta maiņa

Mērķis:

Funkcija dot iespēju lietotājam mainīt e-pastu savam kontam, profila detaļu sadaļā.

Ievaddati:

5.tabula

Lietotāja e-pasta maiņas dati

Nosaukums	Obligāts	Piezīmes
E-pasts	Jā	Ievaddati tiek pārbaudīti, lai sakritu ar e-pasta formātu. No sākuma, lauks ir aizpildīts ar eksistējošo e-pastu. Nedrīkst būt tukšs.

Apstrāde:

Nospiežot apstiprināšanas pogu, tiek pārbaudīts laukā ievadītais e-pasts (Skat. 5. tabulu). Ja ievadītais e-pasts trūkst vai neatbilst e-pasta formātam, tiek radīts paziņojums par validācijas kļūdu. Ja e-pasts ir ievadīts un atbilsts formāta kritērijiem, tas tiek saglabāts datu bāzē, mainot attiecīgajām lietotājam e-pastu uz ievadīto.

Izvaddati:

8) Paziņojums par veiksmīgo e-pasta maiņu;
9) Paziņojums par neveiksmīgo e-pasta maiņu validācijas kļūdas dēļ;
10) Paziņojums par neveiksmīgo e-pasta maiņu servera kļūdas dēļ.

2.2.26. Lietotāja paroles maiņa

Mērķis:

Funkcija dot iespēju lietotājam mainīt paroli savam kontam, profila detaļu sadaļā.

Ievaddati:

6.tabula

Klientu paroles maiņas dati

Nosaukums	Obligāts	Piezīmes
Vecā parole	Jā	Ievaddati tiek pārbaudīti, lai sakritu ar paroli pirms maiņas. Nedrīkst būt tukšs.
Jaunā parole	Jā	Minimālais simbolu garums ir 8. Ir jābūt vismaz 1 lielām burtam un vismaz 1 ciparam. Nedrīkst būt tukšs. Saglabāts šifrētā veidā.
Jaunā parole atkārtoti	Jā	Ir jābūt vienādam ar ievaddatiem laukā "Jaunā parole". Nepieciešams, lai palīdzēt lietotājam atcerēties paroli.

Apstrāde:

Nospiežot apstiprināšanas pogu, tiek pārbaudīti ievadlauki (Skat. 6. tabulu). Ja kāds lauks ir tukšs vai jaunā parole neatbilsts nosacījumiem, tiek radīts paziņojums par validācijas kļūdu. Ja vecā parole nesakrīt ar datu bāzē saglabāto, tiek radīts paziņojums par nepareizo paroli. Ja visi lauki ir ievadīti un atbilst kritērijiem, jaunā parole tiek saglabāta datu bāzē, mainot attiecīgajām lietotājam paroli uz šifrēto jauno.

Izvaddati:

1) Paziņojums par veiksmīgo paroles maiņu;
2) Paziņojums par neveiksmīgo paroles maiņu validācijas kļūdas dēļ;
3) Paziņojums par neveiksmīgo paroles maiņu nepareizas vecās paroles dēļ;
4) Paziņojums par neveiksmīgo paroles maiņu servera kļūdas dēļ.

2.2.27. Ielogošanās administrāciju panelī

Mērķis:

Funkcija nepieciešama, lai kinoteātra administrācijas paneļa lietotāji varētu ielogoties aplikācijā un lietot to.

Ievaddati:

7.tabula

Klientu autorizācijas dati, ielogošana ar e-pastu

Nosaukums	Obligāts	Piezīmes
E-pasts	Jā	
Parole	Jā	Uzglabājas šifrētā formā

Apstrāde:

Nospiežot autorizēšanos pogu, sistēma pārbauda vai visi ievadlauki ir aizpildīti (Skat. 7. tabulu). Ja tie ir, tad tālāk servera pusē tiek pārbaudīti ievadītie dati ar datu bāzi, un ja ievadītie dati ir korekti, un lietotājam ir administratora tiesības - lietotājs tiek autorizēts.

Izvaddati:

1) Nepieciešamība aizpildīt visus ievadlaukus;
2) Paziņojums par nepareizu lietotājvārdu vai paroli;
3) Paziņojums par administratora tiesību trūkumu;
4) Autorizācija sistēmā.

2.2.28. Administrācijas paneļa navigācija

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu pārvietoties starp dažādām administrācijas sadaļām.

Ievaddati:

Lietotājs nospiež uz attiecīgas navigācijas pogas kreisajā malā.

Apstrāde:

Nospiežot kādu no navigācijas pogām, tiek atvērta attiecīga sadaļa: sākums, sākuma elementi, filmas, saraksts, lietotāji, piedāvājumi, promokodi vai maksājumi.

Izvaddati:

Attiecīgās sadaļas attēlošana administrācijas panelī.

2.2.29. Statiskas informācijas attēlošana sākuma ekrānā

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu ātri pārskatīt sistēmas galveno statistiku un aktuālo informāciju.

Apstrāde:

Atveroties sākuma ekrānam, sistēma automātiski ielādē un parāda šādu informāciju:

1) Sveiciena ziņojums ar administratora vārdu un pašreizējo datumu.
2) Filmu, lietotāju, piedāvājumu un promokodu kopējais skaits.
3) Šodienas ieņēmumi un pārdoto biļešu skaits.
4) Šodienas seansu saraksts.
5) Gaidāmās jaunās filmas.
6) Pēdējie maksājumi.
7) Populārākā filma (ar visvairāk pārdotajām biļetēm, no aktuāliem seansiem).

Izvaddati:

Tiek attēlota statistikas informācija kartītēs, kā arī saraksti ar galvenajiem datiem katrā kategorijā.

2.2.30. Filmu saraksta attēlošana administrācijas panelī

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu pārskatīt visas sistēmā esošās filmas.

Apstrāde:

Atvērot sadaļu "Filmas", sistēma automātiski ielādē un parāda visu filmu sarakstu no datu bāzes, katra filma tiek attēlota kā kartīte ar pamatinformāciju.

Izvaddati:

Tiek attēlots filmu saraksts ar katras filmas plakātu, nosaukumu, aprakstu, žanru, režisoru, ilgumu, reitingu un pirmizrādes datumu.

2.2.31. Jaunas filmas pievienošana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu pievienot jaunu filmu sistēmā.

Ievaddati:

8.tabula

Filmas pievienošanas dati

Nosaukums	Obligāts	Piezīmes
Nosaukums	Jā	
Apraksts	Jā	
Žanrs	Jā	
Režisors	Jā	
Pirmizrādes datums	Jā	
Ilgums (min)	Jā	
Vecuma ierobežojums	Jā	
Aktieri	Jā	
Plakāta attēls	Jā	
Vāka attēls	Jā	

Apstrāde:

Nospiežot pogu "Pievienot" filmu sadaļā, atveras filmas pievienošanas forma. Aizpildot visus obligātos laukus (Skat. 8. tabulu) un nospiežot pogu "Saglabāt", sistēma saglabā jauno filmu datu bāzē. Attēli tiek augšupielādēti servera glabātuvē.

Izvaddati:

1) Paziņojums par nepieciešamību aizpildīt visus obligātos laukus;
2) Paziņojums par veiksmīgu filmas pievienošanu;
3) Paziņojums par kļūdu filmas pievienošanā, ja tāda radusies.

2.2.32. Filmas pievienošana, izmantojot ārējo API

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu ātrāk pievienot filmu, izmantojot datus no TMDB API.

Ievaddati:

Lietotājs nospiež uz pogas "Meklēt, lai pievienot" filmas pievienošanas formā.

Apstrāde:

Atveras filmu meklēšanas forma, kurā lietotājs var meklēt filmas no TMDB datubāzes. Pēc filmas izvēles, sistēma automātiski aizpilda filmas pievienošanas formu ar datiem no TMDB, ieskaitot nosaukumu, aprakstu, žanru, ilgumu, pirmizrādes datumu un attēlus. Administratoram ir iespēja rediģēt šos datus pirms saglabāšanas.

Izvaddati:

1) Filmu saraksts no meklēšanas rezultātiem;
2) Automātiski aizpildīta filmas pievienošanas forma;
3) Paziņojums par veiksmīgu filmas pievienošanu pēc saglabāšanas;
4) Paziņojums par kļūdu, ja tāda radusies.

2.2.33. Filmas rediģēšana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu rediģēt esošās filmas datus.

Ievaddati:

Lietotājs nospiež uz filmas kartītes filmu saraksta sadaļā.

Apstrāde:

Atveras filmas rediģēšanas forma ar esošajiem filmas datiem. Lietotājs var mainīt jebkuru filmas lauku, ieskaitot attēlus. Pēc pogas "Saglabāt" nospiešanas, izmaiņas tiek saglabātas datu bāzē.

Izvaddati:

1) Filmas rediģēšanas forma ar esošajiem datiem;
2) Paziņojums par veiksmīgu filmas atjaunināšanu;
3) Paziņojums par kļūdu, ja tāda radusies.

2.2.34. Filmas dzēšana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu dzēst filmu no sistēmas.

Ievaddati:

Lietotājs atver filmas rediģēšanas formu un nospiež pogu "Dzēst".

Apstrāde:

Parādās apstiprinājuma logs. Ja lietotājs apstiprina dzēšanu, filma tiek dzēsta no datu bāzes, kā arī tiek dzēsti visi ar to saistītie seansi un attēli no glabātuves.

Izvaddati:

1) Apstiprinājuma logs pirms dzēšanas;
2) Paziņojums par veiksmīgu filmas dzēšanu;
3) Paziņojums par kļūdu, ja tāda radusies.

2.2.35. Filmu meklēšana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu ātri atrast noteiktu filmu TMDB datubāzē.

Ievaddati:

Lietotājs ievada filmas nosaukumu meklēšanas laukā un nospiež meklēšanas pogu.

Apstrāde:

Sistēma veic vaicājumu TMDB API un attēlo atrastās filmas režģa skatā. Lietotājs var izvēlēties vienu no atrastajām filmām, lai to pievienotu sistēmai.

Izvaddati:

1) Meklēšanas rezultātu attēlošana režģa skatā;
2) Paziņojums, ja nekas nav atrasts;
3) Paziņojums par kļūdu, ja tāda radusies meklēšanas procesā.

2.2.36. Sākuma ekrāna elementa izveidošana no filmas

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu ātri izveidot sākuma ekrāna galerijas elementu, balstoties uz esošu filmu.

Ievaddati:

Lietotājs atver filmas rediģēšanas formu un nospiež pogu "Uztaisīt sākuma elementu".

Apstrāde:

Atveras sākuma ekrāna elementa pievienošanas forma, kura jau ir daļēji aizpildīta ar filmas datiem. Lietotājs var pielāgot elementa nosaukumu un aprakstu, un saglabāt to kā jaunu sākuma ekrāna elementu.

Izvaddati:

1) Daļēji aizpildīta sākuma ekrāna elementa forma;
2) Paziņojums par veiksmīgu elementa pievienošanu;
3) Paziņojums par kļūdu, ja tāda radusies.

2.2.37. Sākuma ekrāna elementu saraksta attēlošana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu pārskatīt visus sistēmā esošos sākuma ekrāna elementus.

Apstrāde:

Atvērot sadaļu "Sākuma elementi", sistēma automātiski ielādē un parāda visu sākuma ekrāna elementu sarakstu no datu bāzes.

Izvaddati:

Tiek attēlots sākuma ekrāna elementu saraksts ar attēlu, nosaukumu, aprakstu un saistīto elementu (filmu vai piedāvājumu) informāciju.

2.2.38. Jauna sākuma ekrāna elementa pievienošana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu pievienot jaunu sākuma ekrāna elementu sistēmā.

Ievaddati:

9.tabula

Sākuma ekrāna elementa pievienošanas dati

Nosaukums	Obligāts	Piezīmes
Nosaukums	Jā	Elementa virsraksts
Apraksts	Jā	Elementa apraksts
Attēls	Jā	Elementa attēls
Saistītais elements	Nē	Saite uz filmu vai piedāvājumu (nav obligāti)

Apstrāde:

Nospiežot pogu "Pievienot +" sākuma elementu sadaļā, atveras elementa pievienošanas forma. Aizpildot obligātos laukus (Skat. 9. tabulu) un nospiežot pogu "Saglabāt", sistēma saglabā jauno elementu datu bāzē. Attēls tiek augšupielādēts servera glabātuvē.

Izvaddati:

1) Paziņojums par nepieciešamību aizpildīt obligātos laukus;
2) Paziņojums par veiksmīgu elementa pievienošanu;
3) Paziņojums par kļūdu, ja tāda radusies.

2.2.39. Sākuma ekrāna elementa rediģēšana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu rediģēt esošos sākuma ekrāna elementus.

Ievaddati:

Lietotājs nospiež uz sākuma ekrāna elementa kartītes sarakstā.

Apstrāde:

Atveras elementa rediģēšanas forma ar esošajiem datiem. Lietotājs var mainīt jebkuru lauku, ieskaitot attēlu un saistīto elementu. Pēc pogas "Saglabāt" nospiešanas, izmaiņas tiek saglabātas datu bāzē.

Izvaddati:

1) Elementa rediģēšanas forma ar esošajiem datiem;
2) Paziņojums par veiksmīgu elementa atjaunināšanu;
3) Paziņojums par kļūdu, ja tāda radusies.

2.2.40. Sākuma ekrāna elementa dzēšana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu dzēst sākuma ekrāna elementu no sistēmas.

Ievaddati:

Lietotājs atver elementa rediģēšanas formu un nospiež pogu "Dzēst".

Apstrāde:

Parādās apstiprinājuma logs. Ja lietotājs apstiprina dzēšanu, elements tiek dzēsts no datu bāzes, kā arī tiek dzēsts tā attēls no glabātuves.

Izvaddati:

1) Apstiprinājuma logs pirms dzēšanas;
2) Paziņojums par veiksmīgu elementa dzēšanu;
3) Paziņojums par kļūdu, ja tāda radusies.

2.2.41. Lietotāju saraksta attēlošana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu pārskatīt visus sistēmā reģistrētos lietotājus.

Apstrāde:

Atvērot sadaļu "Lietotāji", sistēma automātiski ielādē un parāda visu lietotāju sarakstu no datu bāzes.

Izvaddati:

Tiek attēlots lietotāju saraksts ar profila attēlu, vārdu, e-pastu, reģistrācijas datumu un lomu (administrators vai parasts lietotājs).

2.2.42. Lietotāju filtrēšana pēc lomas

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu filtrēt lietotāju sarakstu pēc lomas.

Ievaddati:

Lietotājs izvēlas filtru no nolaižamā saraksta: "Visi lietotāji", "Tikai administratori" vai "Tikai lietotāji".

Apstrāde:

Atkarībā no izvēlētā filtra, sistēma attēlo tikai tos lietotājus, kas atbilst izvēlētajai lomai.

Izvaddati:

Filtrēts lietotāju saraksts pēc izvēlētās lomas.

2.2.43. Jauna lietotāja pievienošana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu pievienot jaunu lietotāju sistēmā.

Ievaddati:

10.tabula

Lietotāja pievienošanas dati

Nosaukums	Obligāts	Piezīmes
Vārds	Jā	Lietotāja vārds
E-pasts	Jā	Unikāls e-pasta adrese
Parole	Jā	Vismaz 6 simboli
Loma	Jā	Administrators vai parasts lietotājs
Profila attēls	Nē	Lietotāja profila attēls

Apstrāde:

Nospiežot pogu "Pievienot +" lietotāju sadaļā, atveras lietotāja pievienošanas forma. Aizpildot obligātos laukus (Skat. 10. tabulu) un nospiežot pogu "Pievienot", sistēma reģistrē jaunu lietotāju. Ja tiek pievienots profila attēls, tas tiek augšupielādēts servera glabātuvē.

Izvaddati:

1) Paziņojums par nepieciešamību aizpildīt obligātos laukus;
2) Paziņojums par veiksmīgu lietotāja pievienošanu;
3) Paziņojums par kļūdu, ja tāda radusies (piemēram, e-pasts jau eksistē).

2.2.44. Lietotāja rediģēšana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu rediģēt esošo lietotāju datus.

Ievaddati:

Lietotājs nospiež uz lietotāja kartītes lietotāju sarakstā.

Apstrāde:

Atveras lietotāja rediģēšanas forma ar esošajiem datiem. Lietotājs var mainīt vārdu, e-pastu, lomu un profila attēlu. Pašreizējais lietotājs nevar mainīt savu lomu. Pēc pogas "Saglabāt" nospiešanas, izmaiņas tiek saglabātas datu bāzē.

Izvaddati:

1) Lietotāja rediģēšanas forma ar esošajiem datiem;
2) Paziņojums par veiksmīgu lietotāja atjaunināšanu;
3) Paziņojums par kļūdu, ja tāda radusies.

2.2.45. Lietotāja dzēšana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu dzēst lietotāju no sistēmas.

Ievaddati:

Lietotājs atver lietotāja rediģēšanas formu un nospiež pogu "Dzēst". Pašreizējais lietotājs nevar dzēst savu kontu.

Apstrāde:

Parādās apstiprinājuma logs. Ja lietotājs apstiprina dzēšanu, lietotājs tiek dzēsts no datu bāzes, kā arī tiek dzēsts tā profila attēls no glabātuves.

Izvaddati:

1) Apstiprinājuma logs pirms dzēšanas;
2) Paziņojums par veiksmīgu lietotāja dzēšanu;
3) Paziņojums par kļūdu, ja tāda radusies.

2.2.46. Seansu saraksta attēlošana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu pārskatīt visus sistēmā esošos seansus.

Ievaddati:

Lietotājs izvēlas datumu, par kuru vēlas redzēt seansus, izmantojot datumu izvēles pogas.

Apstrāde:

Atvērot sadaļu "Saraksts", sistēma automātiski ielādē un parāda izvēlētās dienas seansu sarakstu no datu bāzes. Pēc noklusējuma tiek parādīti šodienas seansi.

Izvaddati:

Tiek attēlots seansu saraksts ar filmas attēlu, nosaukumu, žanru, ilgumu, laiku un zāles numuru.

2.2.47. Jauna seansa pievienošana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu pievienot jaunu seansu sistēmā.

Ievaddati:

11.tabula

Seansa pievienošanas dati

Nosaukums	Obligāts	Piezīmes
Datums	Jā	Seansa datums
Laiks	Jā	Seansa laiks
Zāle	Jā	Zāles numurs (1-5)
Filma	Jā	Izvēlētā filma no saraksta

Apstrāde:

Nospiežot pogu "Pievienot +" seansu sadaļā, atveras seansa pievienošanas forma. Aizpildot obligātos laukus (Skat. 11. tabulu) un nospiežot pogu "Pievienot", sistēma saglabā jauno seansu datu bāzē.

Izvaddati:

1) Paziņojums par nepieciešamību aizpildīt obligātos laukus;
2) Paziņojums par veiksmīgu seansa pievienošanu;
3) Paziņojums par kļūdu, ja tāda radusies.

2.2.48. Seansa rediģēšana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu rediģēt esošā seansa datus.

Ievaddati:

Lietotājs nospiež uz seansa kartītes seansu sarakstā.

Apstrāde:

Atveras seansa rediģēšanas forma ar esošajiem datiem. Lietotājs var mainīt datumu, laiku, zāli un filmu. Pēc pogas "Saglabāt" nospiešanas, izmaiņas tiek saglabātas datu bāzē.

Izvaddati:

1) Seansa rediģēšanas forma ar esošajiem datiem;
2) Paziņojums par veiksmīgu seansa atjaunināšanu;
3) Paziņojums par kļūdu, ja tāda radusies.

2.2.49. Seansa dzēšana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu dzēst seansu no sistēmas.

Ievaddati:

Lietotājs atver seansa rediģēšanas formu un nospiež pogu "Dzēst".

Apstrāde:

Parādās apstiprinājuma logs. Ja lietotājs apstiprina dzēšanu, seanss tiek dzēsts no datu bāzes.

Izvaddati:

1) Apstiprinājuma logs pirms dzēšanas;
2) Paziņojums par veiksmīgu seansa dzēšanu;
3) Paziņojums par kļūdu, ja tāda radusies.

2.2.50. Piedāvājumu saraksta attēlošana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu pārskatīt visus sistēmā esošos piedāvājumus.

Apstrāde:

Atvērot sadaļu "Piedāvājumi", sistēma automātiski ielādē un parāda visu piedāvājumu sarakstu no datu bāzes.

Izvaddati:

Tiek attēlots piedāvājumu saraksts ar attēlu, nosaukumu, aprakstu un saistīto promokodu (ja tāds ir).

2.2.51. Jauna piedāvājuma pievienošana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu pievienot jaunu piedāvājumu sistēmā.

Ievaddati:

12.tabula

Piedāvājuma pievienošanas dati

Nosaukums	Obligāts	Piezīmes
Nosaukums	Jā	Piedāvājuma virsraksts
Apraksts	Jā	Piedāvājuma apraksts
Attēls	Jā	Piedāvājuma attēls
Promokods	Nē	Saite uz promokodu (nav obligāti)

Apstrāde:

Nospiežot pogu "Pievienot +" piedāvājumu sadaļā, atveras piedāvājuma pievienošanas forma. Aizpildot obligātos laukus (Skat. 12. tabulu) un nospiežot pogu "Pievienot", sistēma saglabā jauno piedāvājumu datu bāzē. Attēls tiek augšupielādēts servera glabātuvē.

Izvaddati:

1) Paziņojums par nepieciešamību aizpildīt obligātos laukus;
2) Paziņojums par veiksmīgu piedāvājuma pievienošanu;
3) Paziņojums par kļūdu, ja tāda radusies.

2.2.52. Piedāvājuma rediģēšana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu rediģēt esošā piedāvājuma datus.

Ievaddati:

Lietotājs nospiež uz piedāvājuma kartītes piedāvājumu sarakstā.

Apstrāde:

Atveras piedāvājuma rediģēšanas forma ar esošajiem datiem. Lietotājs var mainīt nosaukumu, aprakstu, attēlu un saistīto promokodu. Pēc pogas "Saglabāt" nospiešanas, izmaiņas tiek saglabātas datu bāzē.

Izvaddati:

1) Piedāvājuma rediģēšanas forma ar esošajiem datiem;
2) Paziņojums par veiksmīgu piedāvājuma atjaunināšanu;
3) Paziņojums par kļūdu, ja tāda radusies.

2.2.53. Piedāvājuma dzēšana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu dzēst piedāvājumu no sistēmas.

Ievaddati:

Lietotājs atver piedāvājuma rediģēšanas formu un nospiež pogu "Dzēst".

Apstrāde:

Parādās apstiprinājuma logs. Ja lietotājs apstiprina dzēšanu, piedāvājums tiek dzēsts no datu bāzes, kā arī tiek dzēsts tā attēls no glabātuves.

Izvaddati:

1) Apstiprinājuma logs pirms dzēšanas;
2) Paziņojums par veiksmīgu piedāvājuma dzēšanu;
3) Paziņojums par kļūdu, ja tāda radusies.

2.2.54. Maksājumu saraksta attēlošana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu pārskatīt visus sistēmā veiktos maksājumus.

Apstrāde:

Atvērot sadaļu "Maksājumi", sistēma automātiski ielādē un parāda visu maksājumu sarakstu no datu bāzes, sakārtotu pēc datuma.

Izvaddati:

Tiek attēlots maksājumu saraksts ar ID, datumu, lietotāja informāciju, summu, statusu un seansa informāciju. Augšpusē tiek rādīta maksājumu statistika – veiksmīgo un neveiksmīgo maksājumu kopējā summa un skaits.

2.2.55. Maksājumu filtrēšana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu filtrēt maksājumu sarakstu pēc dažādiem kritērijiem.

Ievaddati:

Lietotājs var izvēlēties filtru no nolaižamā saraksta: "Visi maksājumi", "Veiksmīgi maksājumi" vai "Neveiksmīgi maksājumi". Papildus var izvēlēties konkrētu datumu, par kuru redzēt maksājumus.

Apstrāde:

Atkarībā no izvēlētajiem filtriem, sistēma attēlo tikai tos maksājumus, kas atbilst izvēlētajiem kritērijiem.

Izvaddati:

Filtrēts maksājumu saraksts atbilstoši izvēlētajiem kritērijiem.

2.2.56. Maksājumu meklēšana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu meklēt maksājumus pēc ID, lietotāja ID vai seansa ID.

Ievaddati:

Lietotājs ievada meklējamo tekstu meklēšanas laukā.

Apstrāde:

Sistēma meklē maksājumus, kas satur ievadīto tekstu maksājuma ID, lietotāja ID vai seansa ID laukos.

Izvaddati:

Meklēšanas rezultāti – maksājumi, kas atbilst meklēšanas kritērijiem.

2.2.57. Maksājumu detaļu kopēšana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu ātri kopēt maksājumu ID, seansa ID vai lietotāja ID.

Ievaddati:

Lietotājs nospiež uz kopēšanas ikonas blakus attiecīgajam ID maksājuma detaļu skatā.

Apstrāde:

Izvēlētais ID tiek kopēts starpliktuvē.

Izvaddati:

Paziņojums par veiksmīgu ID kopēšanu.

2.2.58. Promokodu saraksta attēlošana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu pārskatīt visus sistēmā esošos promokodus.

Apstrāde:

Atvērot sadaļu "Promokodi", sistēma automātiski ielādē un parāda visu promokodu sarakstu no datu bāzes.

Izvaddati:

Tiek attēlots promokodu saraksts ar nosaukumu, atlaides tipu (procentuālā vai fiksētā summa) un atlaides vērtību.

2.2.59. Jauna promokoda pievienošana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu pievienot jaunu promokodu sistēmā.

Ievaddati:

13.tabula

Promokoda pievienošanas dati

Nosaukums	Obligāts	Piezīmes
Promokoda nosaukums	Jā	Unikāls promokoda nosaukums, lielajiem burtiem
Atlaides tips	Jā	Fiksēta summa (€) vai procentuāla (%)
Atlaides vērtība	Jā	Summa eiro vai procentu vērtība

Apstrāde:

Nospiežot pogu "Pievienot +" promokodu sadaļā, atveras promokoda pievienošanas forma. Aizpildot obligātos laukus (Skat. 13. tabulu) un nospiežot pogu "Pievienot", sistēma saglabā jauno promokodu datu bāzē. Ja lietotājs ievada promokodu, kas jau eksistē, sistēma parāda kļūdas paziņojumu.

Izvaddati:

1) Paziņojums par nepieciešamību aizpildīt obligātos laukus;
2) Paziņojums par veiksmīgu promokoda pievienošanu;
3) Paziņojums par kļūdu, ja tāda radusies (piemēram, promokods jau eksistē).

2.2.60. Promokoda rediģēšana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu rediģēt esošā promokoda datus.

Ievaddati:

Lietotājs nospiež uz promokoda kartītes promokodu sarakstā.

Apstrāde:

Atveras promokoda rediģēšanas forma ar esošajiem datiem. Lietotājs var mainīt nosaukumu, atlaides tipu un vērtību. Pēc pogas "Saglabāt" nospiešanas, izmaiņas tiek saglabātas datu bāzē.

Izvaddati:

1) Promokoda rediģēšanas forma ar esošajiem datiem;
2) Paziņojums par veiksmīgu promokoda atjaunināšanu;
3) Paziņojums par kļūdu, ja tāda radusies (piemēram, jaunais nosaukums jau eksistē).

2.2.61. Promokoda dzēšana

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu dzēst promokodu no sistēmas.

Ievaddati:

Lietotājs atver promokoda rediģēšanas formu un nospiež pogu "Dzēst".

Apstrāde:

Parādās apstiprinājuma logs. Ja lietotājs apstiprina dzēšanu, promokods tiek dzēsts no datu bāzes. Ja promokods ir piesaistīts kādam piedāvājumam, dzēšana nav iespējama, un lietotājs saņem attiecīgu kļūdas paziņojumu.

Izvaddati:

1) Apstiprinājuma logs pirms dzēšanas;
2) Paziņojums par veiksmīgu promokoda dzēšanu;
3) Paziņojums par kļūdu, ja promokods ir piesaistīts piedāvājumam.

2.2.62. Izlogošanās no administrācijas paneļa

Mērķis:

Funkcija nepieciešama, lai administrācijas paneļa lietotāji varētu droši izlogoties no sistēmas.

Ievaddati:

Lietotājs nospiež uz navigācijas pogas "Izlogoties" vai apstiprina izlogošanos apstiprinājuma logā.

Apstrāde:

Parādās apstiprinājuma logs "Vai tiešām vēlaties izlogoties?". Ja lietotājs apstiprina, tiek veikta izlogošanās no Firebase autentifikācijas, un lietotājs tiek novirzīts uz administrācijas pieslēgšanās ekrānu.

Izvaddati:

1) Apstiprinājuma logs pirms izlogošanās;
2) Atgriešanās administrācijas pieslēgšanās ekrānā pēc veiksmīgas izlogošanās.

2.3. Sistēmas nefunkcionālās prasības

1) Klientu aplikācijai jāatbalsta mobilās ierīces operētājsistēmas Android un IOS;
2) Administrācijas aplikācijai jāatbalsta ierīces operētājsistēmas Windows un macOS;
3) Sistēmas lietotāja saskarnei jābūt saprotamai un viegli pārskatāmai;
4) Saskarne, skatu struktūra, fonti, burtu izmēri u.c. ievēroti vienādi visos aplikācijas skatos;
5) Pievienot, rediģēt un dzēst informāciju no aplikācijas sadaļām var rediģēt tikai administrācija vai attiecīgi norādīts darbinieks, izņemot katra lietotāja konta informāciju, kuru arī spēj rediģēt katrs lietotājs tikai savam kontam;
6) Datu apstrādes vidējais lauks ne ilgāk par 3 sekundēm;
7) Aplikācijai ir jābūt pielāgotai visbiežāk sastopamiem ekrānu izmēriem mobilām ierīcēm;
8) Administrācijas aplikācijai ir jābūt minimālais limits aplikācijas loga izmēram;
9) Lietotāju dati, īpaši paroles un maksājumu informācija, ir jāuzglabā drošā veidā, izmantojot atbilstošus šifrēšanas algoritmus. Paroles nekad netiek glabātas atklātā veidā;
10) Sistēmai jānodrošina stabila darbība arī vienlaicīgu lietotāju pieslēgšanās gadījumā, īpaši filmu pirmizrāžu laikā, kad lietotāju aktivitāte ir paaugstināta;
11) Aplikācijai jādarbojas arī sliktos interneta savienojuma apstākļos, saglabājot pamat funkcionalitāti un nodrošinot iespēju apskatīt jau iegādātās biļetes bezsaistē, izmantojot lokālo kešatmiņu;
12) Jānodrošina regulāri sistēmas atjauninājumi, lai novērstu drošības problēmas un pievienotu jaunu funkcionalitāti, ar minimālu ietekmi uz lietotāju pieredzi;
13) Sistēmai jābūt modulārai un viegli paplašināmai, ļaujot nākotnē pievienot jaunas funkcijas, piemēram, lojalitātes programmu vai personalizētus filmu ieteikumus;
14) Administrācijas panelim jānodrošina lietotāja darbību auditācijas žurnāls ar visām būtiskajām izmaiņām, kas veiktas sistēmā, lai būtu iespējams izsekot, kurš un kad veicis konkrētas darbības.

2.4. Gala lietotāja raksturiezīmes

Mobilās aplikācijas “Filmu Nams” lietotāju klāsts ir apjomīgs un ir paredzēta nākošajiem lietotājiem:

- Lietotājiem, kuri vēlās apmeklēt kinoteātri. Mobilā lietotne “Filmu Nams” sniedz iespēju lietotājiem noskatīties pieejamo filmu sarakstu, kā arī attiecīgas dienas sarakstu un iespēju nopirkt biļeti uz izvēlēto seansu, apmaksājot to ar internet banku vai bankas karti.
- Lietotājiem kuri vēlās apskatīt jauno filmu sarakstu. Aplikācijā ir pieejama sadaļa ar filmām un jebkurš autentificēts lietotājs drīkst pārskatīt to.

3. Izstrādes līdzekļu, rīku apraksts un izvēles pamatojums

Aplikācijas “Filmu Nams” izstrādei, kā bāzi, tika izvēlēti rīki Flutter un Firebase, jo tās ir spēcīga kombinācija starpplatformu risinājumu izveidei ar servera puses atbalstu.

Flutter nodrošina elastīgu un ātru lietotāja interfeisa izstrādi Android un iOS operētājsistēmām, savukārt Firebase nodrošina rīkus drošai datu glabāšanai un lietotāju autentifikācijai.

Turpmāk tika detalizēti aprakstīti šo tehnoloģiju izvēles pamatojums.

3.1. Izvēlēto risinājuma līdzekļu un valodu apraksts

3.1.1. Flutter un Dart programmēšanas valoda – saskarne

Flutter tika izvēlēts projekta izstrādei, pateicoties tā starpplatformu izstrādes iespējām, kas ļauj izveidot vienu lietotni Android un iOS. Turklāt lietotne atbalsta tīmekļa versiju, kurā būs pieejams administrācijas panelis.

Viena no galvenajām priekšrocībām ir vienkārša stilizēšana un interfeisa pārvaldība, pateicoties Dart valodai.

Dart nodrošina augstu veiktspēju, tīru sintaksi un iespēju efektīvi pārvaldīt lietojumprogrammas stāvokli, kas paātrina izstrādi un vienkāršo projekta atbalstu.

3.1.2. Firebase – servera puses izstrāde

Firebase tika izvēlēta šim projektam, jo tā piedāvā jaudīgas iespējas servera puses izstrādei bez nepieciešamības pārvaldīt savu serveri (datubāzes pārvaldība, autorizācijas un autentifikācijas iespējas, analītika utt.).

Tā nodrošina ērtus rīkus lietotāju autentifikācijai, glabāšanai mākonī un paziņojumu pārvaldībai. Firebase izmantošana ļauj viegli mērogot lietojumprogrammu un nodrošina augstu veiktspēju, pateicoties reāllaika integrācijai.

Šīs funkcijas padara to par ideālu risinājumu kino lietojumprogrammas funkcionalitātes un administrēšanas paneļa atbalstam.

3.1.3. Visual Studio Code – izstrādes vide

Visual Studio Code tika izvēlēta kā galvenā izstrādes vide projekta izstrādei, jo tā ir viegla, augstas veiktspējas un plašu pielāgošanas iespēju dēļ.

Tā nodrošina spēcīgu Flutter un Dart atbalstu, izmantojot paplašinājumus, nodrošinot projekta ērto palaišanu, atkļūdošanas rīkus un sintakses pārbaudi, kā arī integrāciju ar versiju kontroles sistēmām, piemērām, GitHub, un lietotājam draudzīgs terminālis.

Šīs daudzas priekšrocības padara to par daudzpusīgu rīku efektīvai izstrādei un projektu pārvaldībai.

3.1.4. Xcode - IOS emulators

Xcode emulators tiek izmantots, lai testētu un atkļūdotu lietojumprogrammu iOS ierīcēs. Tā ir Apple oficiālā izstrādes vide un emulators, kas nodrošina precīzu lietojumprogrammas atveidošanu dažādos iPhone un iPad modeļos.

Izmantojot Xcode, ir iespējams pārbaudīt savas lietotnes saderību un veiktspēju iOS platformā, kā arī pirms publicēšanas pārbaudīt lietotāja saskarni un funkcijas.

3.1.5. Android Studio – Android emulators

Android Studio emulators tiek izmantots, lai testētu un atkļūdošanas programmas Android ierīcēs.

Tas ir jaudīgs rīks, kas ļauj simulēt dažādus viedtālruņu modeļus, Android versijas un darbības apstākļus, tostarp tīkla iestatījumus un aparatūras specifikācijas.

Android Studio emulators nodrošina precīzu lietojumprogrammas uzvedības atveidošanu Android ierīcēs, kas palīdz identificēt un novērst kļūdas pirms produkta izlaišanas.

3.1.6. GitHub – projekta versionēšana

GitHub tiek izmantots versiju kontrolei un projektu sadarbībai. Šī platforma ļauj droši uzglabāt pirmkodu, sekot izmaiņām un efektīvi pārvaldīt izstrādes uzdevumus, izmantojot Git versiju kontroles sistēmu.

GitHub izmantošana nodrošina izstrādes procesa pārredzamību, vienkāršo komandas sadarbību un nodrošina ērtus rīkus lietojumprogrammu izveides un izvietošanas automatizēšanai.

3.2. Iespējamo (alternatīvo) risinājuma līdzekļu un valodu apraksts

3.2.1. React Native - saskarne

React Native var izmantot kā alternatīvu Flutter. Šī platforma ļauj arī izstrādāt starpplatformu mobilās lietotnes Android un iOS, izmantojot JavaScript un React.

Galvenās React Native priekšrocības ir iespēja izmantot esošās React ekosistēmas bibliotēkas un ātra izstrāde, pateicoties pazīstamiem “frontend” rīkiem.

Tomēr, salīdzinot ar Flutter, veiktspēja un interfeisa stilizācija var būt mazāk elastīga, tāpēc tā nav tik ieteicama grafiski intensīvām lietojumprogrammām.

3.2.2. Laravel un MySQL datu bāze – servera daļa

Kā alternatīvu Firebase projekta servera pusē ir iespējams izmantot Laravel un MySQL kombināciju.

Laravel ir populārs PHP karkass, kas piedāvā ērtus rīkus backend izstrādei, tostarp maršrutēšanu, sesiju pārvaldību, lietotāju autentifikāciju un vaicājumu apstrādi. Tas ietver arī spēcīgu Eloquent datubāzes ORM ērtai mijiedarbībai ar MySQL. Savukārt MySQL ir viena no populārākajām relāciju datubāzēm, kas ir ideāli piemērota strukturētu datu glabāšanai un sarežģītu vaicājumu veikšanai.

Lai gan Laravel un MySQL izmantošana prasa servera konfigurāciju un datubāzes pārvaldību, šī kombinācija nodrošina lielāku elastību un datu kontroli, kā arī ļauj īstenot sarežģītāku loģiku un biznesa procesus, padarot to par lielisku alternatīvu Firebase mērogojamākiem projektiem.

3.2.3. Android Studio kā izstrādes vide

Kā alternatīvu Visual Studio Code Flutter izstrādei ir iespējams izmantot Android Studio. Android Studio ir jaudīga integrētā izstrādes vide, kas nodrošina visus nepieciešamos rīkus Flutter lietojumprogrammu izveidei, tostarp Dart atbalstu, iebūvētu emulatoru, atkļūdošanas programmu un lietotāja saskarnes iespējas.

Tomēr, neraugoties uz tās priekšrocībām, Android Studio prasa vairāk sistēmas resursu, kas var palēnināt veiktspēju vecākās vai mazāk jaudīgās ierīcēs, kā arī neatbalsta Flutter projektu palaišanu uz IOS emulatoru vai pieslēgto ierīci.

Šie faktori padara VS Code piemērotāku ātrai prototipu izveidei un ikdienas darbam, savukārt Android Studio būs noderīga sarežģītākiem uzdevumiem un izstrādāšanai Android vidē.

3.2.4. Īstas ierīces izmantošana emulatoru vietā

Flutter un Visual Studio Code piedāvā iespēju palaist projektu uz mobilajām ierīcēm, kas pieslēgtas izstrādātāja datoram, ļaujot pārbaudīt saskarni un funkcionalitāti uz reālām ierīcēm. Tomēr izstrādātājam ir nepieciešamas vismaz divas ierīces, viena ar Android un otra ar iOS operētājsistēmu, lai varētu testēt aplikāciju abās platformās.

Emulatoriem ir priekšrocība, jo tie ļauj ātri un vienkārši pielāgot dažādus parametrus, piemēram, ekrāna izmēru vai operētājsistēmas versiju, kas nav iespējams mainīt reālajām ierīcēm.

Ideāls risinājums būtu izmantot gan emulatorus, gan reālas ierīces aplikācijas izstrādē.

3.2.5. GitLab - versionēšana

GitLab ir viena no populārākajām alternatīvām GitHub, kas piedāvā līdzīgas funkcijas kā koda versiju kontrole, projekta pārvaldība un sadarbība.

Tā ir atvērtā koda platforma, kas ļauj izstrādātājiem pārvaldīt Git repozitorijus, veikt komandas sadarbību, kā arī automatizēt izstrādes procesus.

Tāpat GitLab piedāvā spēcīgu saskarni un integrāciju ar citiem izstrādes rīkiem, kas padara to par ērtu alternatīvu GitHub, īpaši, ja ir nepieciešama iekšēja kontrole pār jūsu infrastruktūru.

Izvēle starp GitLab un GitHub ir atkarīga no izstrādātāja vajadzībām, piemēram, vai ir nepieciešama pilnīga platformas kontrole vai priekšrocības publiskajā kopienā un atvērtajā koda ekosistēmā.

4. Sistēmas modelēšana un projektēšana

4.1. Sistēmas struktūras modelis

Filmu Nams mobilās aplikācijas struktūra ir izstrādāta, izmantojot MVC (Model-View-Controller) arhitektūras principus, kas nodrošina kodu organizāciju, vieglu uzturēšanu un funkcionalitāti. Sistēma sastāv no vairākiem savstarpēji saistītiem komponentiem, kas organizēti hierarhiskā struktūrā.

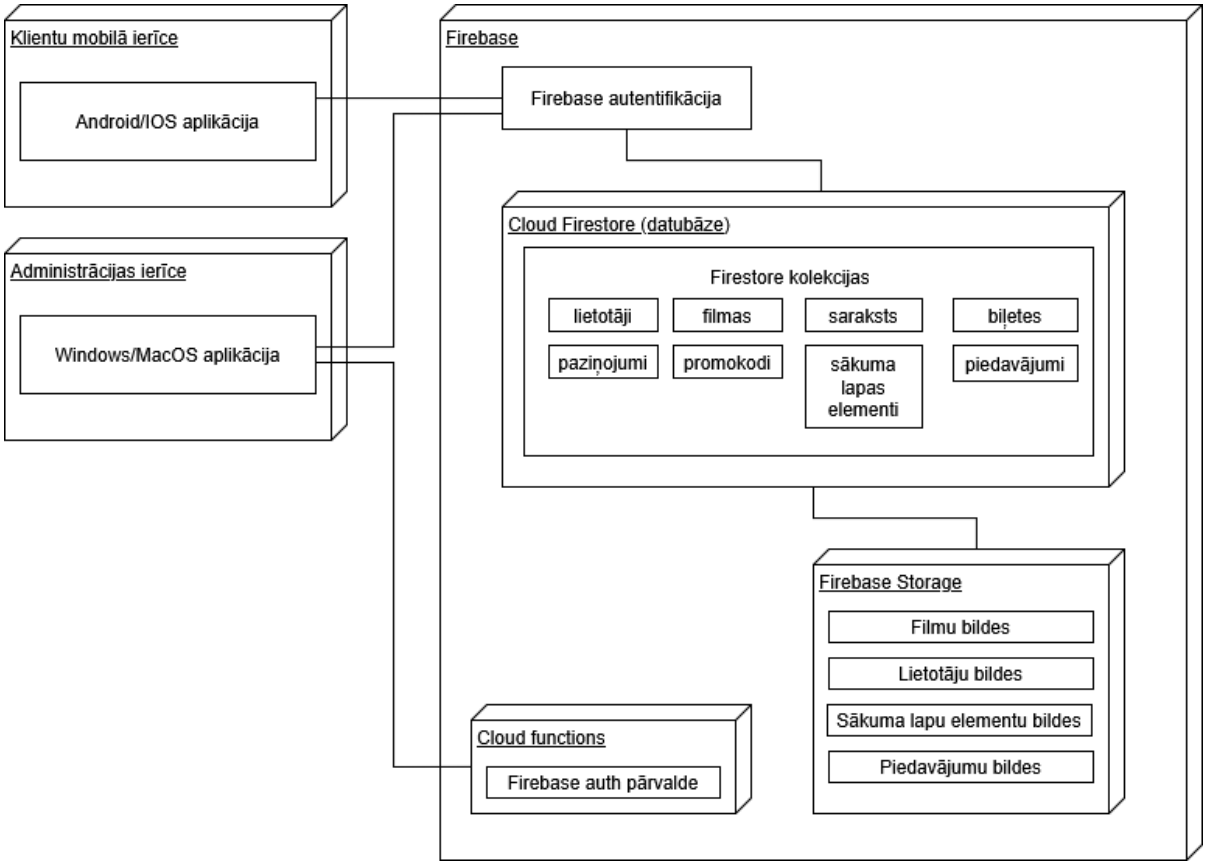
Aplikācijas pamata komponentes ir:

- Datu modeļi (Models) - definē datu struktūras;
- Skati (Views) - definē lietotāja saskarnes elementus;
- Kontrolleri (Controllers) - apstrādā loģiku un datu pārvaldību.
- Servisi - nodrošina savienojumu ar Firebase pakalpojumiem (Firestore, Authentication, Storage).

Aplikācija izmanto Firebase platformu kā pamata infrastruktūru, nodrošinot autentifikāciju, datu glabāšanu un mediju pārvaldību, kas ļauj visiem lietotājiem piekļūt vieniem un tiem pašiem datiem, neatkarīgi no ierīces.

4.1.1. Sistēmas struktūra (izvietojuma diagramma)

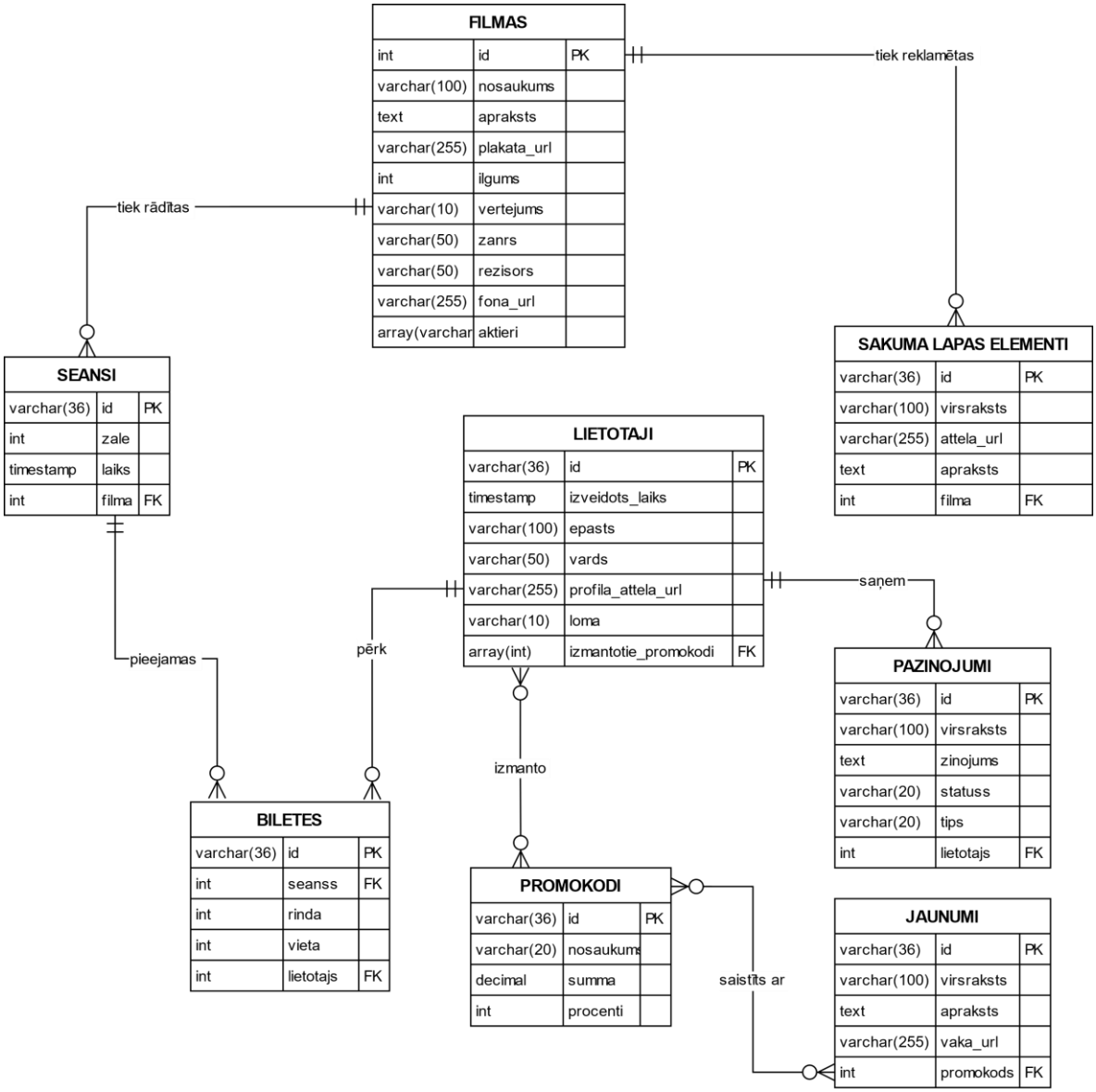
Šī diagramma atspoguļo "Filmu Nams" sistēmas arhitektūru un komponenšu izvietojumu. Tā parāda divas galvenās lietotāju saskarnes - klientu mobilo ierīci (Android/iOS) un administrācijas ierīci (Windows/MacOS), kas abas mijiedarbojas ar Firebase pakalpojumiem. Firebase pakalpojumi ietver autentifikāciju, Firestore datubāzi ar dažādām kolekcijām (lietotāji, filmas, saraksts, biļetes, utt.), Firebase Storage mediju glabāšanai un Cloud Functions servisa funkcijām. Diagramma (Skat. 1. attēlu.) uzskatāmi parāda, kā dažādās sistēmas komponentes ir savienotas un kā tās mijiedarbojas savā starpā.



1. attēls. Abu aplikāciju struktūra

4.1.2. ER diagramma

Šī diagramma (Skat. 2. attēlu.) atspoguļo "Filmu Nams" datubāzes struktūru, parādot galvenās datu entītijas un to savstarpējās attiecības. Tajā attēlotas tabulas FILMAS, SEANSI, BIĻETES, LIETOTĀJI, PAZIŅOJUMI, PROMOKODI un citas, ar norādītiem lauku tipiem (varchar, int, utt.) un attiecību veidiem (viens-pret-daudziem, daudzi-pret-daudziem). Diagramma skaidri parāda primārās un ārējās atslēgas, kā arī obligātos atribūtus. Šī struktūra ilustrē, kā dati tiek organizēti un saistīti sistēmā, nodrošinot pilnvērtīgu kinoteātra funkcionalitāti.



2. attēls. Datu bāzes relāciju diagramma

4.2. Funkcionālais un dinamiskais sistēmas modelis

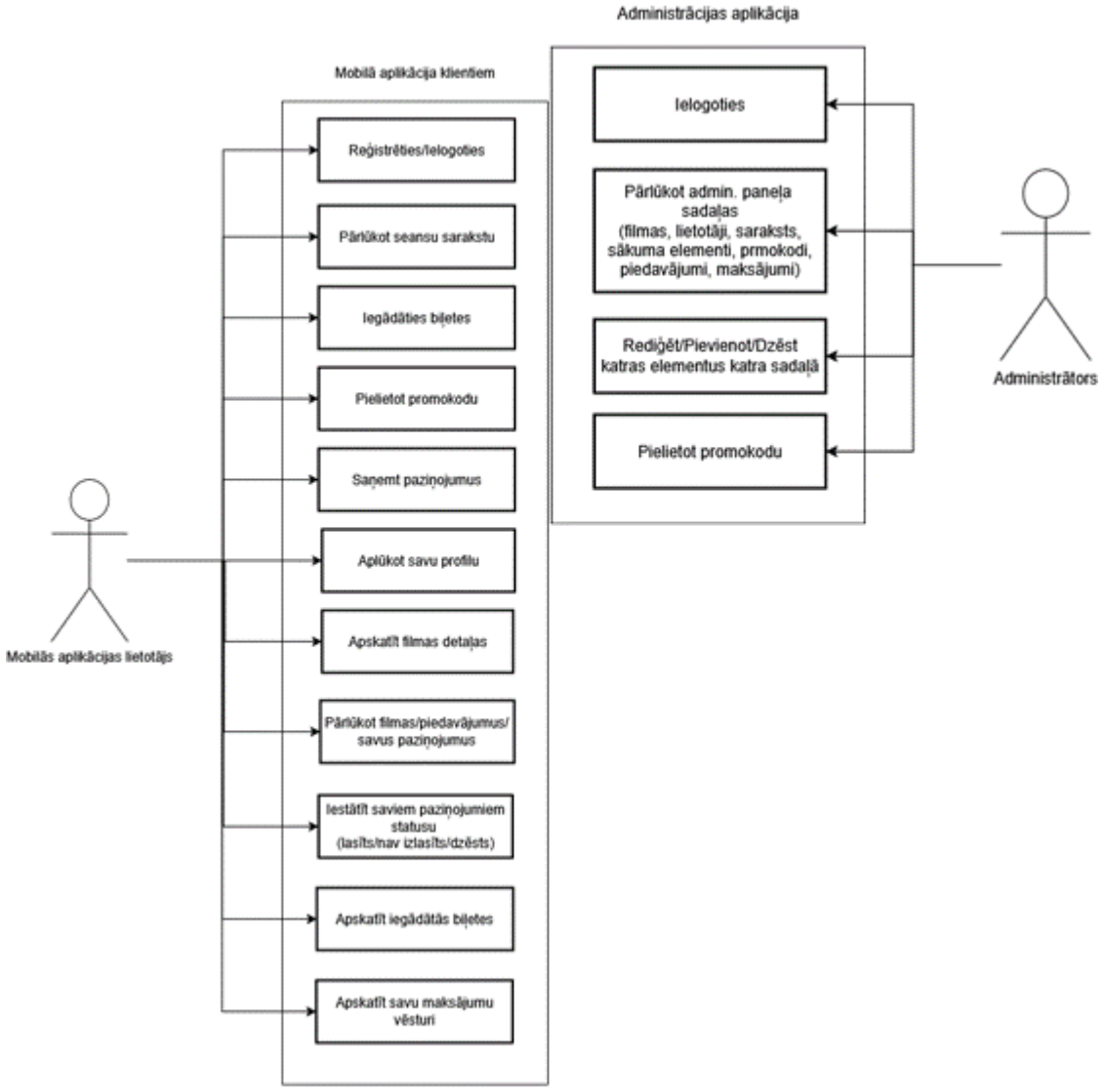
"Filmu Nams" aplikācija darbojas gan ar klientu sadaļu, gan kā administrācijas panelis atkarībā no lietotāja lomas. Sistēmas galvenie lietojumgadījumi ietver:

- Lietotāja autentifikācija (reģistrācija, pieslēgšanās, e-pasta verifikācija);
- Filmu pārlūkošana;
- Seansa izvēle un biļešu iegāde;
- Lietotāja profila pārvaldība;
- Administrācijas panelī – datubāzes satura pārvaldība.

Galvenās aktivitātes klientu saskarnē ietver filmu pārlūkošanu, seansu izvēli, sēdvietu rezervēšanu un maksājumu apstrādi, savukārt administrācijas saskarne ļauj pārvaldīt sistēmas saturu un lietotājus.

4.2.1. Lietojumgadījumu diagramma (Use Case)

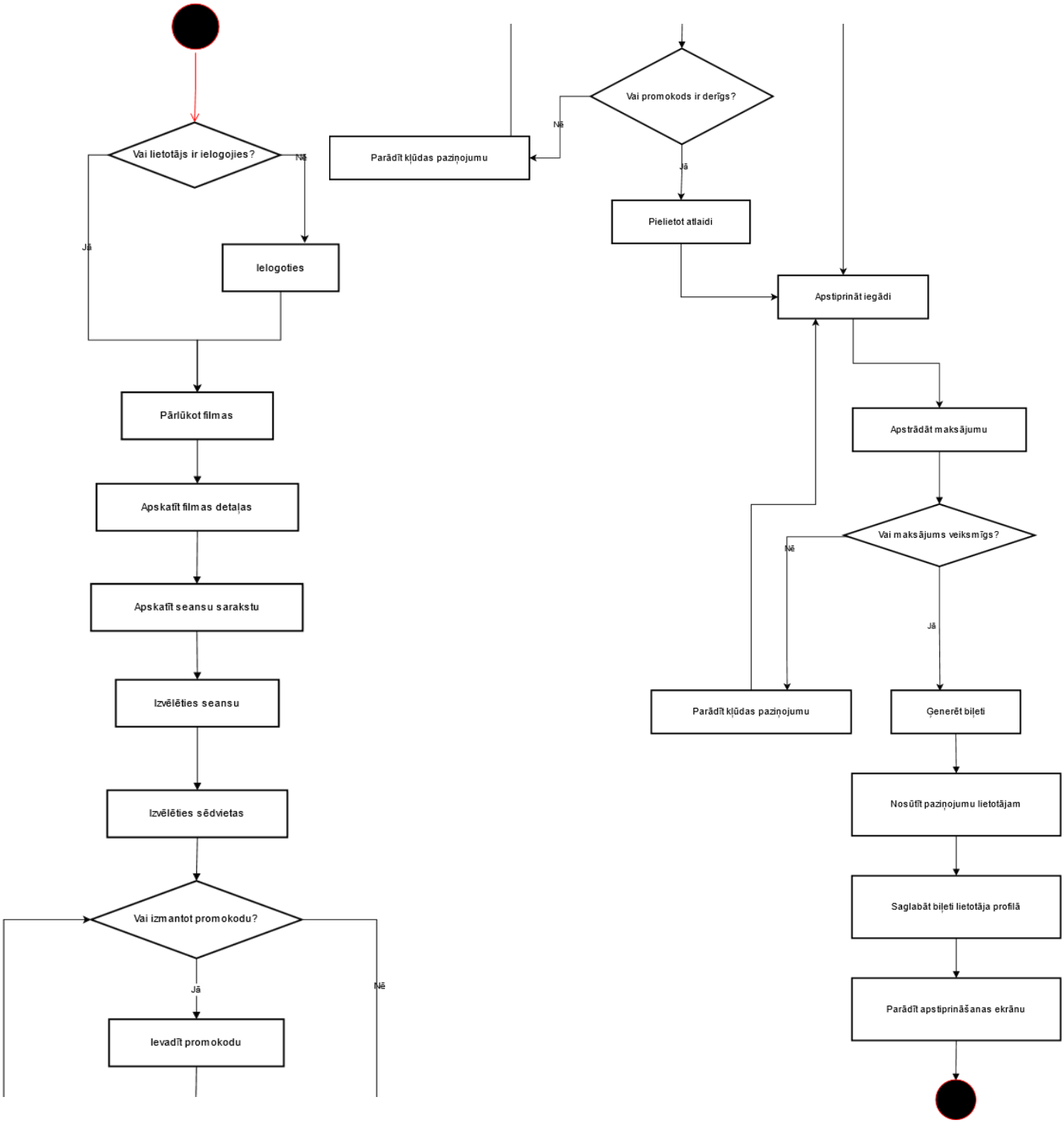
Šī diagramma (Skat. 3. attēlu.) parāda sistēmas funkcionalitāti no lietotāju perspektīvas, ilustrējot, ko dažādi lietotāju tipi var darīt ar sistēmu. Kreisajā pusē redzamas mobilās aplikācijas lietotāju iespējas (reģistrēties/ielogoties, pārlūkot seansu sarakstu, iegādāties biļetes, pielietot promokodu, utt.), bet labajā pusē - administratoru iespējas (ielogoties, pārvaldīt administrācijas paneļa saturu, rediģēt dažādus elementus). Diagramma sniedz skaidru priekšstatu par sistēmas lietojuma robežām un lietotāju lomu atšķirībām.



3. attēls. Aplikāciju Lietojumgadījumi

4.2.2. Aktivitāšu diagramma (Activity)

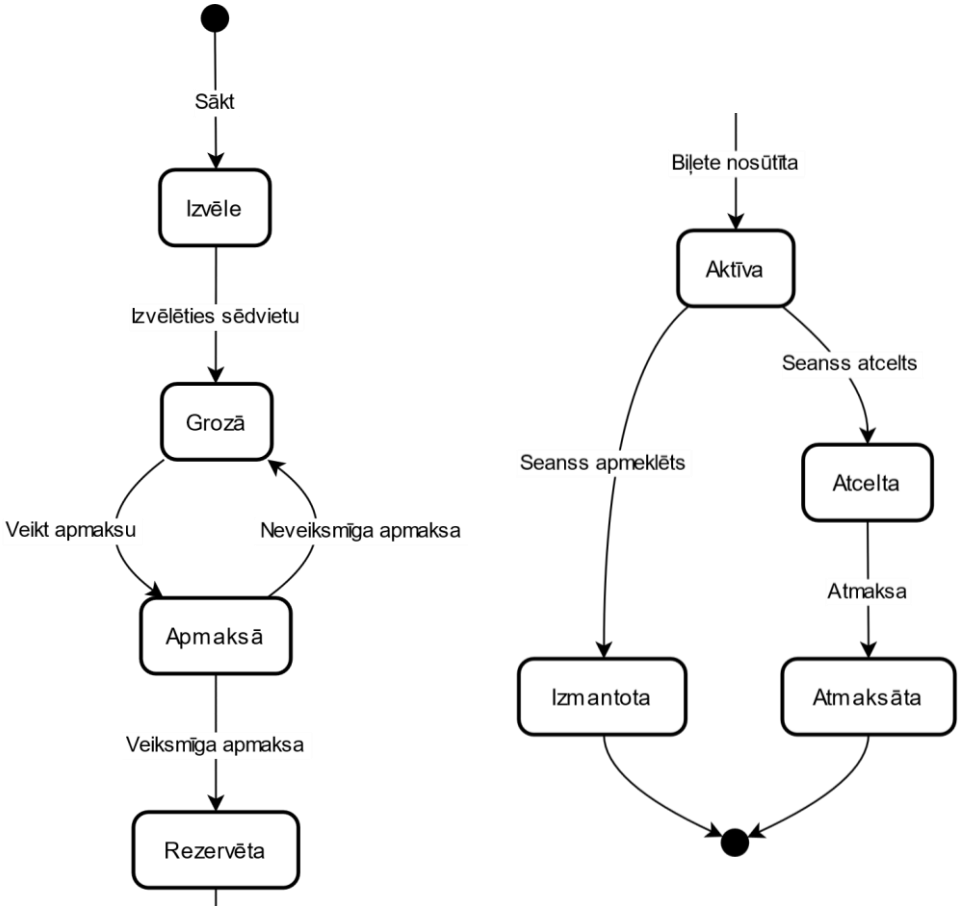
Šī diagramma (Skat. 4. attēlu.) ilustrē maksājumu un biļešu ģenerēšanas procesu. Diagramma skaidri parāda alternatīvās plūsmas veiksmīga un neveiksmīga maksājuma gadījumā.



4. attēls. Maksājuma aktivitāšu diagramma

4.2.3. Stāvokļu diagramma (State)

Šī diagramma (Skat. 5. attēlu.) atspoguļo biļetes dzīves ciklu "Filmu Nams" sistēmā. Sākot ar biļetes izveidi, tā pāriet uz grozā ievietotu stāvokli, tad uz apmaksas stāvokli (ar iespējamiem veiksmīgas vai neveiksmīgas apmaksas rezultātiem), rezervācijas stāvokli pēc veiksmīgas apmaksas, un aktīvu stāvokli, kad biļete ir derīga. No aktīva stāvokļa biļete var pāriet uz izmantota stāvokli (ja seansa apmeklējums notiek) vai atcelta stāvokli (ja seansa apmeklējums nenotiek). Diagramma sniedz skaidru priekšstatu par to, kā biļete maina savus stāvokļus dažādos tās dzīves cikla posmos.



5. attēls. Maksājuma stāvokļu diagramma

4.3. Datu struktūru apraksts

Mobilas aplikācijas izstrādei ir izvēlēta NoSQL datu glabāšanas pieeja, izmantojot Firebase Firestore, kas nodrošina elastīgu un mērogojamu risinājumu. Šī izvēle ir pamatota ar vairākiem būtiskiem faktoriem:

Firebase Firestore piedāvā dokumentu bāzētu datu struktūru, kas ļauj organizēt saistīto informāciju kolekcijās, piemēram, filmas, seansi, lietotāji, promokodi, utt. Šāda pieeja atvieglo datu pārvaldību un ļauj efektīvi veidot vaicājumus, kas ir īpaši svarīgi, kad nepieciešams, piemēram, filtrēt seansus pēc datuma vai atlasīt filmas pēc žanra.

4.3.1. Galvenās datu modeļu klases

Aplikācijā tiek izmantoti strukturēti datu modeļi, kas atspoguļo biznesa loģiku un nodrošina konsekventu datu apstrādi, piemēram:

1) **Filmas modelis**- detalizēta informācija par filmām:
    a. Pamatinformācija (nosaukums, apraksts, žanrs);
    b. Tehniskie parametri (ilgums, pirmizrādes datums);
    c. Kreatīvā informācija (režisors, aktieri);
    d. Mediju faili (plakāta un vāka attēlu URL);
2) **Seansa modelis** - saista filmas ar konkrētiem seansiem:
    a. Saite uz filmu;
    b. Laiks un datums;
    c. Zāles numurs;
    d. Biļetes cena;
3) **Biļetes modelis** - reprezentē lietotāju iegādātās biļetes:
    a. Saite uz seansu;
    b. Saite uz lietotāju;
    c. Iegādes datums;
    d. Sēdvietas informācija;
    e. Biļetes statuss;

Firebase Firestore gadījumā, kur nav tiešas relāciju atbalsta kā tradicionālajās SQL datubāzēs, tiek izmantota dokumentu atsauču pieeja, lai veidotu saites starp dažādiem datu objektiem. Dokumentu atsauces tiek izmantotas, lai veidotu saites starp saistītiem objektiem, piemēram, starp biļeti un seansu vai starp piedāvājumu un promokodu. "Viena pret vienu" attiecības tiek pārvaldītas ar tiešām dokumentu atsaucēm, bet "viena pret daudzām" attiecības tiek risinātas, izmantojot masīvus vai apakškolekcijas.

4.3.2. Datu cietības stratēģija

Sistēma izmanto vairākas datu glabāšanas stratēģijas optimālai darbībai:

Firebase Authentication - autentifikācijas risinājums, kas ciešā integrācijā ar Firestore nodrošina vienkāršu lietotāju piekļuves kontroli un datu drošību. Tas ļauj viegli īstenot lomu bāzētu piekļuvi, nodalot klientu un administratoru skatījumus un funkcionalitāti.

Firebase Firestore - primāra datu glabātuve, kas nodrošina reāllaika sinhronizāciju starp visām ierīcēm un serveri.

Firebase Storage - mediju failu glabāšanai, kas efektīvi apstrādā liela apjoma failus un nodrošina optimizētu piekļuvi no mobilās aplikācijas.

4.3.3. MVC arhitektūras pielietojums datu apstrādē

Aplikācija implementē Model-View-Controller (MVC) arhitektūras principus, kas nodrošina skaidru atbildības sadali datu pārvaldībā:

- **Modeļi (Models)** - datu struktūras un bizness loģika;
- **Skati (Views)** - lietotāja saskarnes komponentes;
- **Kontrolleri (Controllers)** - starpnieki starp modeļiem un skatiem.

Piemēram, filmu kontrolleris, nodrošina vairākas kritiski svarīgas funkcijas datu pārvaldībai:

- Datu ielādi no Firebase Firestore;
- Datu sinhronizāciju ar lokālo stāvokli;
- Datu validāciju un apstrādi;
- Kļūdu apstrādi un paziņošanu.

Šāda arhitektūra nodrošina koda modularitāti, vieglu testēšanu un uzturēšanu, kā arī ļauj efektīvi pārvaldīt sarežģītas datu plūsmas.

4.3.4. Drošība un datu integritāte

Firebase Firestore drošības noteikumi nodrošina granulāru piekļuves kontroli datiem. Sistēmā implementēti šādi drošības principi:

- Lietotāju autentifikācijas pārbaude pirms piekļuves sensitīviem datiem;
- Lomu bāzēta piekļuves kontrole (RBAC), kas nodrošina, ka lietotāji var piekļūt tikai tiem datiem, kas atbilst viņu lomai;
- Validācijas noteikumi, kas garantē datu integritāti gan klienta, gan servera pusē;

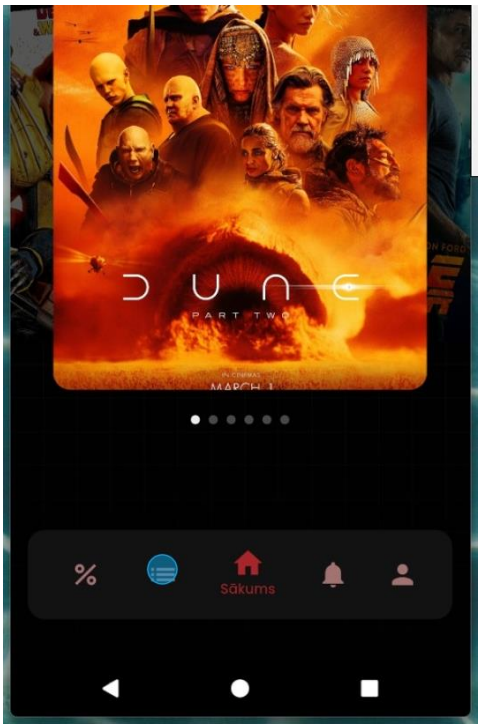
- Transakciju mehānismi, kas nodrošina atomāras datu izmaiņas, piemēram, biļešu pirkumu reģistrācijā.

Šie principi kopā veido daudzslāņainu drošības arhitektūru, kas aizsargā lietotāju datus un nodrošina sistēmas stabilu darbību.

5. Lietotāju ceļvedis

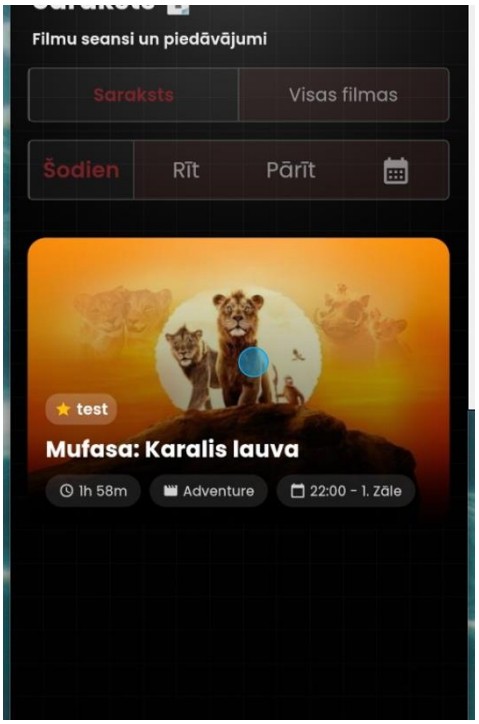
5.1. Biļešu iegāde

1) Nospiediet uz sadaļas "Saraksts";



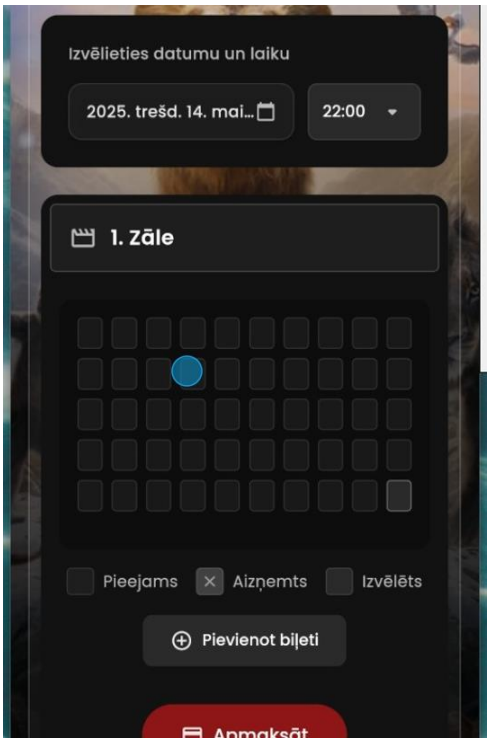
6. attēls. Pāreja uz saraksta sadaļu

2) Izvēlējieties seansu;



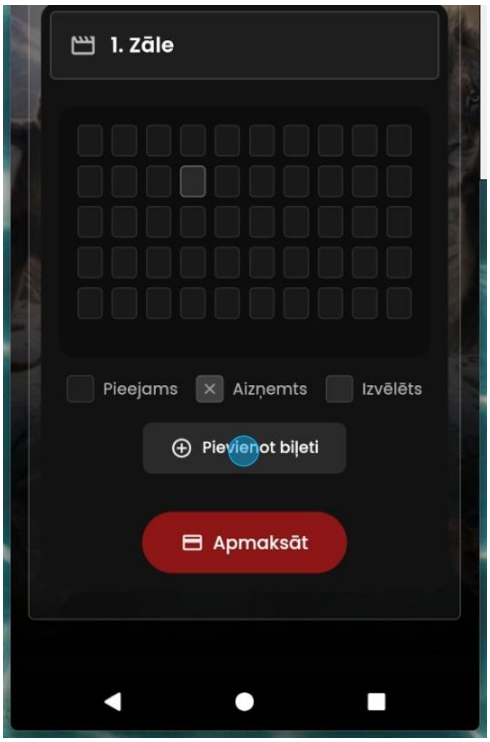
7. attēls. Filmu un sarakstu skats

3) Izvēlējieties sēdvietu;



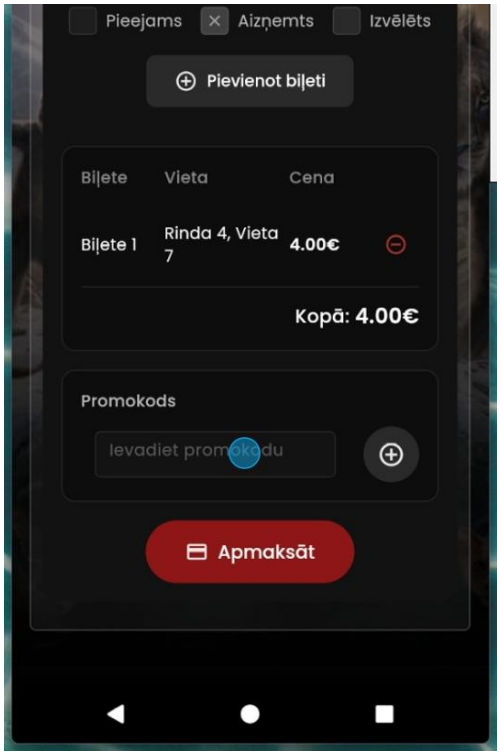
8. attēls. Sēdvietas izvēle

4) Pievienojiet izvēlēto sēdvietu;



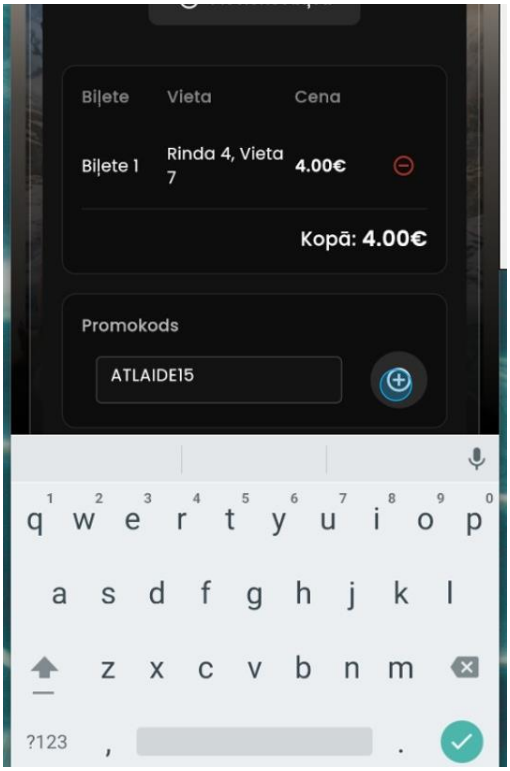
9. attēls. Biļetes pievienošana

5) Ievadiet promokodu;



10. attēls. Promokoda ievadīšana

6) Aktivizējiet promokodu;

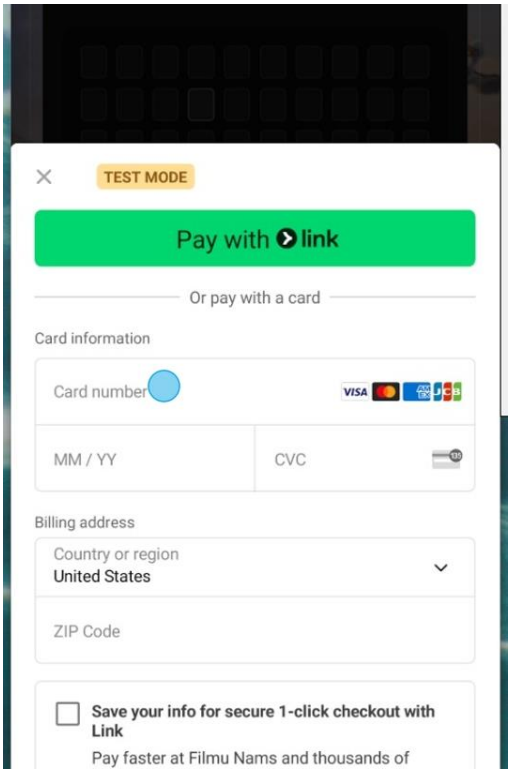


11. attēls. Promokoda pievienošana

7) Nospiediet "Apmaksāt";

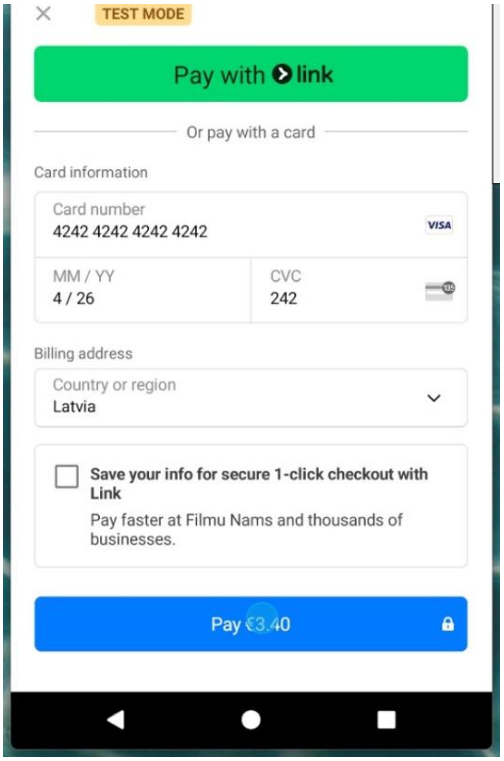
12. attēls. Maksājuma poga

8) Ievadiet datus maksāšanai;



13. attēls. Stripe maksājuma forma

9) Apstipriniet maksājumu;

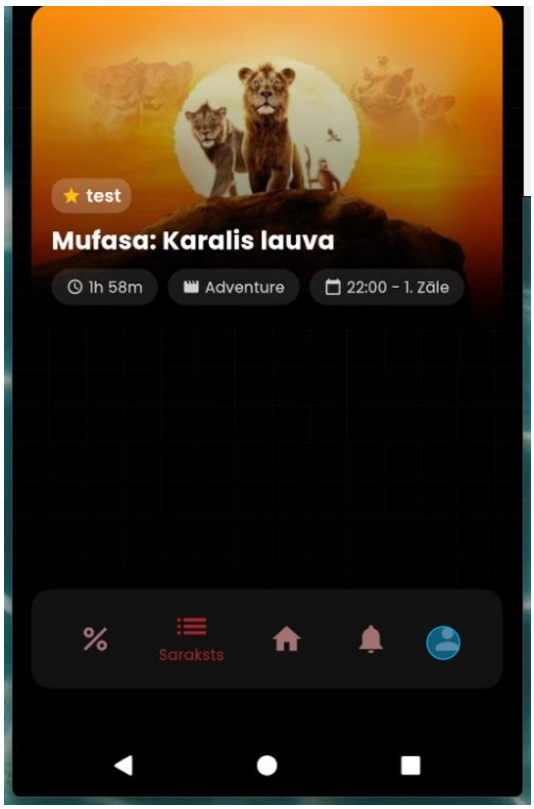


14. attēls. Aizpildīta Stripe forma

10) Izejiet ārā no filmas skata;

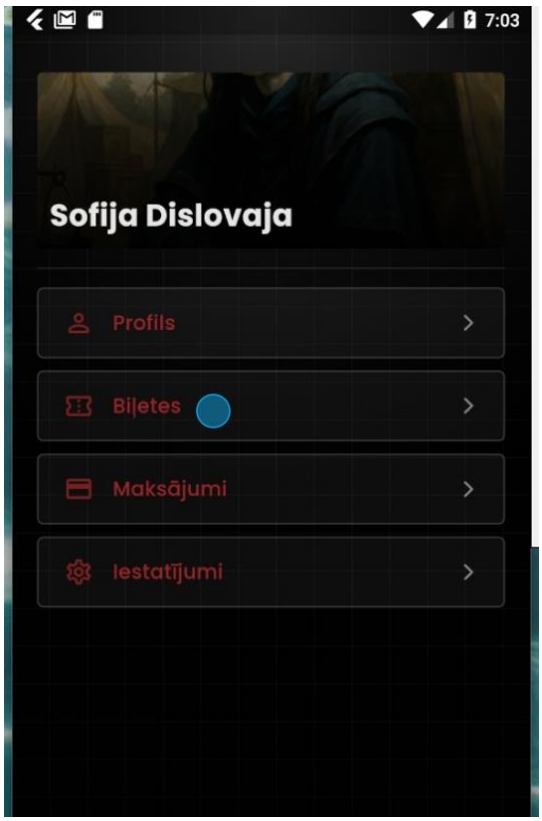
15. attēls. Pāreja no filmas skata uz sarakstu

11) Nospiediet uz profila sadaļas;



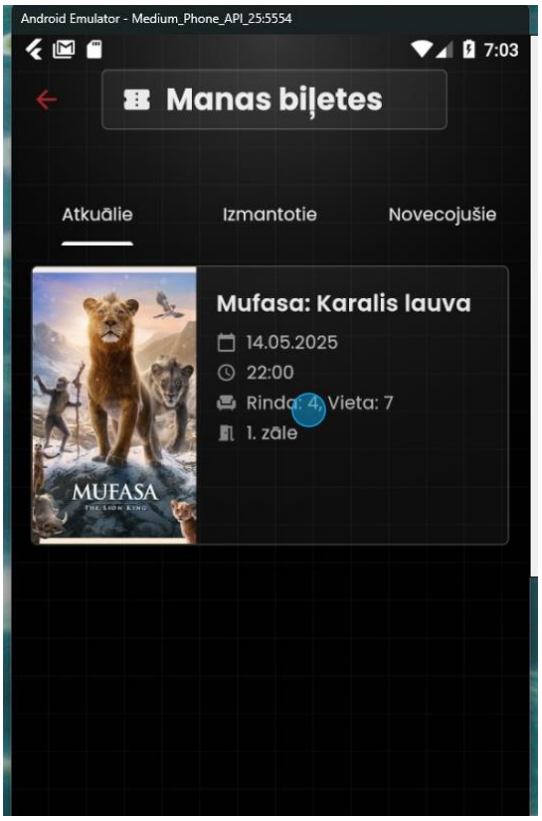
16. attēls. Pāreja uz profila skatu

12) Nospiediet uz biļešu sadaļas;



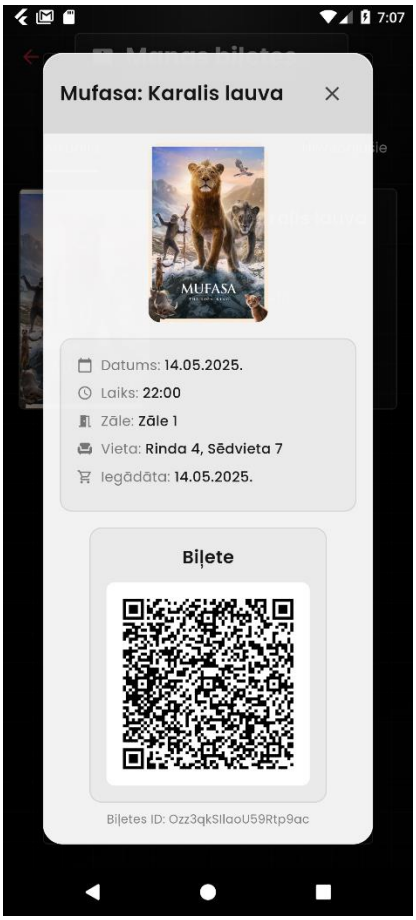
17. attēls. Pāreja uz biļešu skatu

13) Nospiediet uz biļetes;



18. attēls. Biļešu skats

14) Biļete atvērās ar detalizētu skatu, kur arī ir pieejams QR kods.



19. attēls. Biļetes detalizēts skats

5.2. Piedāvājumu attēlošana un promokoda kopēšana

Šeit būs paskaidrots:

- Kā var pāriet uz piedāvājumu sadaļu;
- Kā var atvērt kādu no piedāvājumiem detalizēti;
- Kā var nokopēt promokodu, ja tās ir piesaistīts pie piedāvājuma.

1) Nospiediet uz piedāvājumu sadaļas pogas;

20. attēls. Galvenais skats, pāreja uz piedāvājumu skatu

2) Nospiediet uz pogas "Lasīt vairāk", lai atvērt piedāvājuma pilno skatu;

21. attēls. Piedāvājumu saraksts

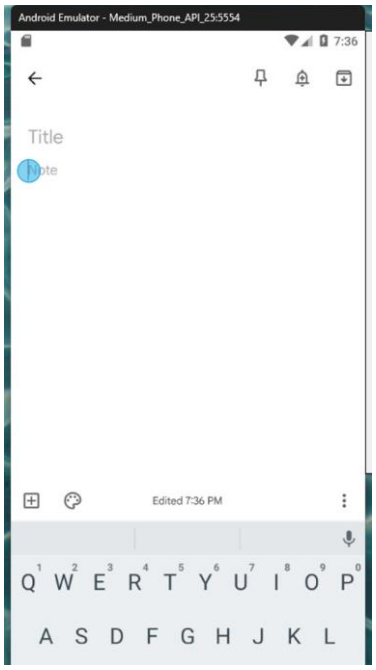
3) Ja piedāvājumam ir piesaistīts promokods, ir jābūt redzamai pogai "Kopēt". Nospiediet uz tās;

22. attēls. Piedāvājumu detalizēts skats

4) Tiek radīts paziņojums par veiksmīgo darbību. Nospiedām "Ok!"

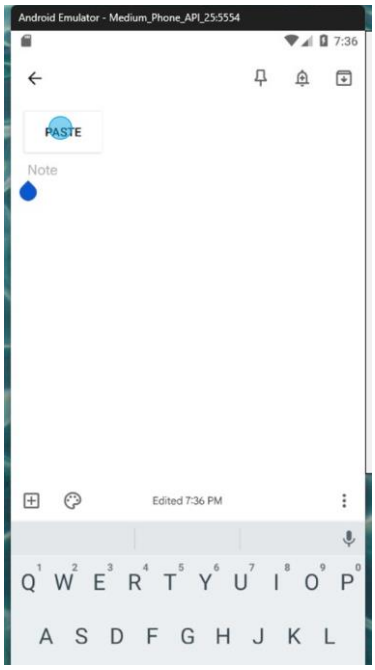
23. attēls. Paziņojums par promokoda kopēšanu

5) Tagad, jebkurā vietā, kur ir iespējams ievadīt tekstu, var pārbaudīt vai promokods bija saglabāts ierīces starpliktuvē. Nospiediet un turiet pirkstu līdz paradās opcija "Ievietot";



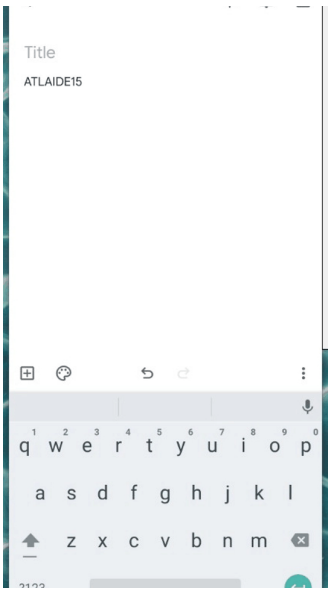
24. attēls. Teksta redaktors

6) Nospiedām uz opcijas "Ievietot";



25. attēls. Promokoda ievietošana teksta redaktorā

7) Nokopētais promokods tiek ierakstīts teksta laukā, kas liecina par to, ka promokods bija veiksmīgi saglabāts ierīces starpliktuvē.



26. attēls. Promokoda kopēšanas gala rezultāts

5.3. Darbības ar paziņojumiem

Šeit būs paskaidrots:

- Kā var pāriet uz paziņojumu sadaļu;
- Kā var atvērt kādu no paziņojumiem detalizēti;
- Kā var mainīt paziņojuma statusu;
- Kā var izdzēst paziņojumu.

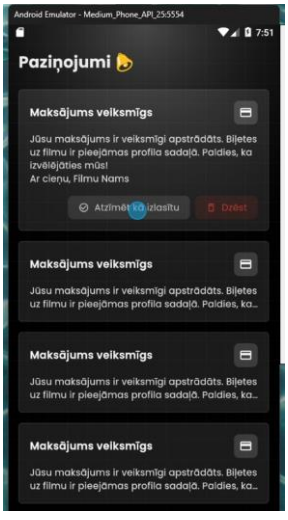
1) Nospiediet uz paziņojumu sadaļas pogas, lai pāriet uz attiecīgo skatu;

27. attēls. Galvenais skats, pāreja uz paziņojumu skatu

2) Nospiediet uz paziņojuma, lai atvērtu to detalizēto skatu;

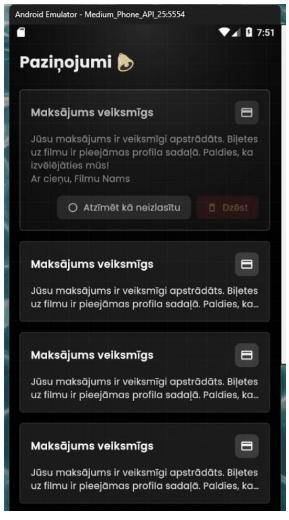
28. attēls. Paziņojumu saraksts

3) Lai mainīt status uz izlasītu vai neizlasītu, ir jānospiež pirmā poga no paziņojuma piedāvātām darbībām;



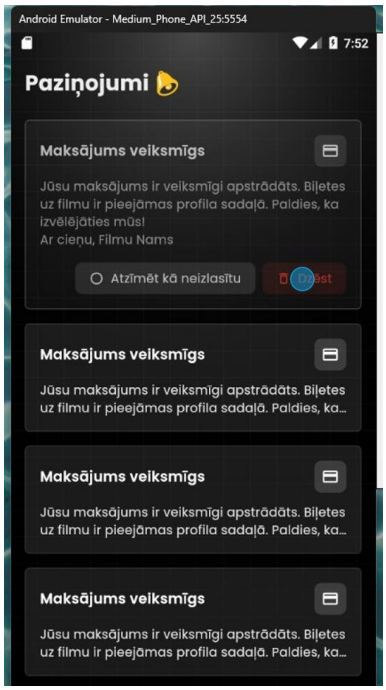
29. attēls. Paziņojuma detalizēts skats

4) Paziņojuma krāsa mainās uz tumšo, kad tās ir izlasīts, un uz gaišo, kad tās nav izlasīts;



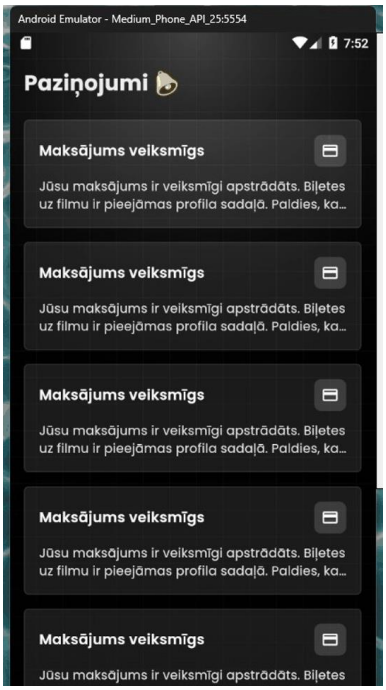
30. attēls. Paziņojums ar statusu "Izlasīts"

5) Lai nodzēst paziņojumu, nospiediet pogu "Dzēst";



31. attēls. Paziņojuma poga "Dzēst"

6) Paziņojums pazūd no saraksta, kas liecina par veiksmīgo dzēšanu.



32. attēls. Saraksts bez izdzēsta paziņojuma

5.4. Lietotāju profila datu maiņa

Šeit būs paskaidrots:

- Kā var pāriet uz profila sadaļu;
- Kā var atvērt profila detalizētu skatu;
- Kā var nomainīt profila bildi;
- Kā var nomainīt vārdu vai e-pastu.

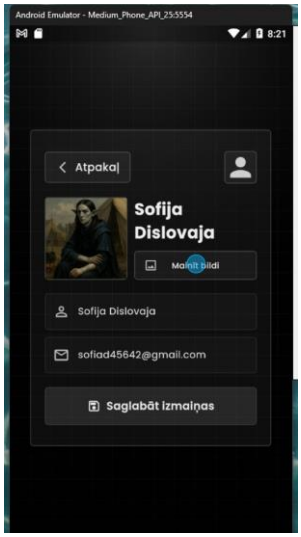
1) Nospiediet uz profila sadaļas pogas;

33. attēls. Galvenais skats, pāreja uz profila sadaļu

2) Nospiediet uz pogas "Profils", lai atvērt detalizētu skatu;

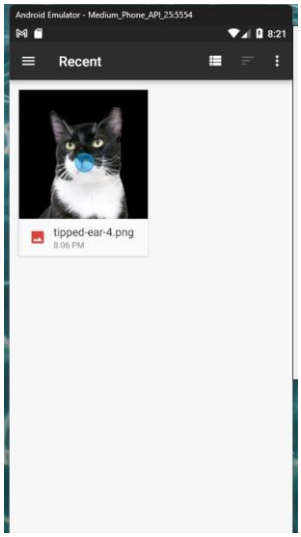
34. attēls. Profila sadaļa

3) Tiek radīti lietotāja profila dati. Nospiediet "Mainīt bildi";



35. attēls. Profila detalizētais skats

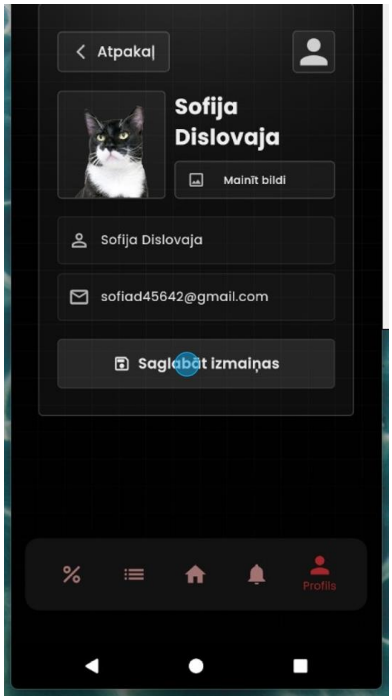
4) Izvēlējieties bildi no ierīces galerijas;



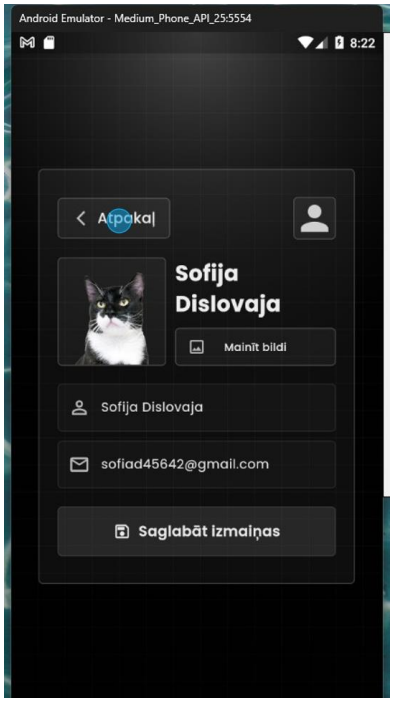
36. attēls. Bildes izvēle

5) Nospiediet saglabāt izmaiņas;

6) Izejiet atpakaļ uz profila skatu, lai pārliecināties, ka bilde tika saglabāta;



37. attēls. Bildes saglabāšana

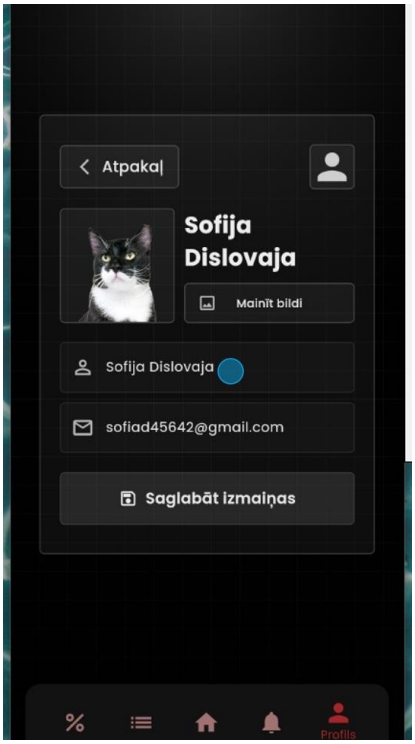


38. attēls. Poga "Atpakaļ"

7) Bilde mainījies. Atgriežamies profila rediģēšanas formā;

8) Ievadiet jaunus datus (vārds un/vai e-pasts);

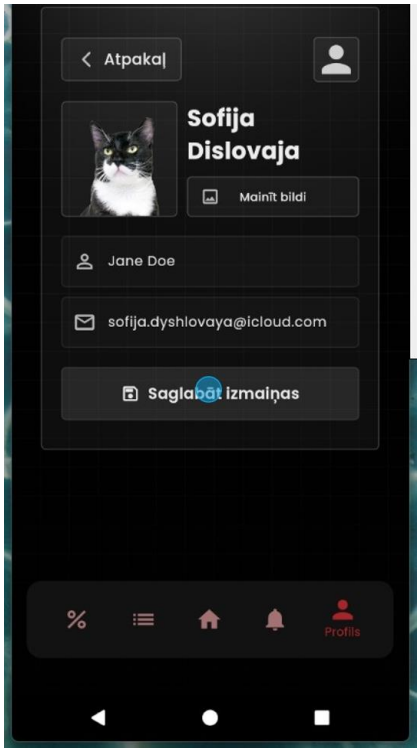
39. attēls. Jaunā profila bilde



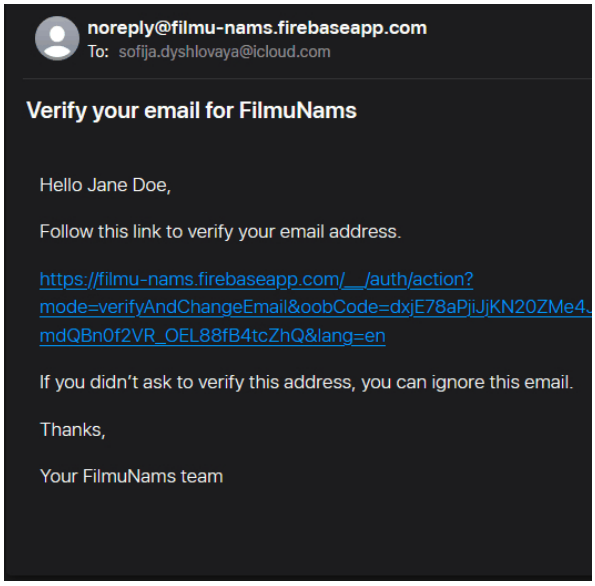
40. attēls. Vārda un e-pasta maiņa

9) Nospiediet saglabāt izmaiņas. Vārds ir saglabājies, bet e-pastam ir vajadzīga apstiprināšana;

10) Apstipriniet jauno e-pastu, izmantojot saiti vēstulē;

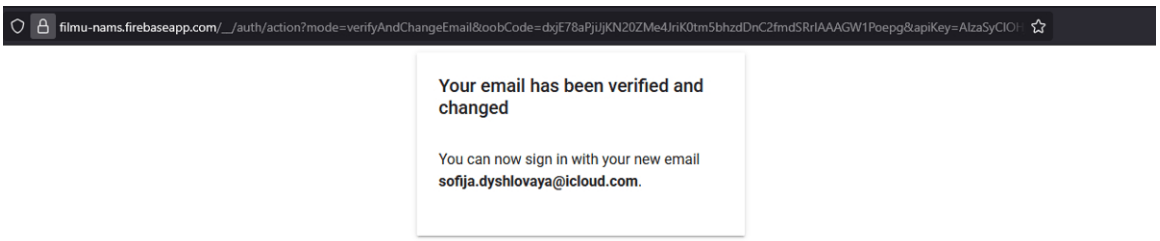


41. attēls. Vārda un e-pasta saglabāšana



42. attēls. Vēstule jauno e-pastu aktivizēšanai

11) E-pasts ir saglabāts.



43. attēls. Jauna e-pasta aktivizēšana

5.5. Maksājumu sadaļas attēlošana un maksājuma ID kopēšana

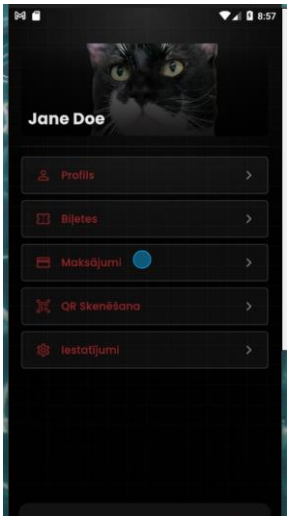
Šeit būs paskaidrots:

- Kā var pāriet uz maksājumu sadaļu;
- Kā var atvērt kādu no maksājumiem detalizēti;
- Kā var nokopēt maksājuma ID.

1) Nospiediet profila sadaļas pogu;

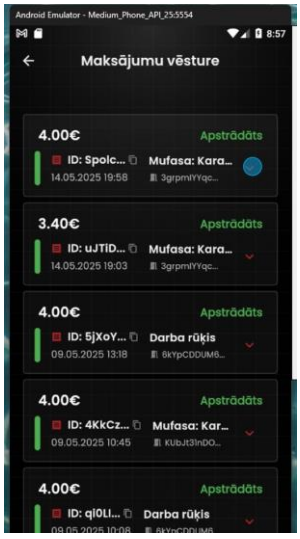
44. attēls. Galvenais skats, pāreja uz profila sadaļu

2) Nospiediet uz maksājumu sadaļas pogu;



45. attēls. Pāreja uz maksājumu sadaļu

3) Tiek radīts saraksts ar maksājumiem. Nospiediet bultiņu, lai atvērt maksājumu detalizētāk;



46. attēls. Maksājumu sadaļa

4) Tiek radīts maksājumā detalizēts skats. Nospiediet uz pogas "Kopēt";

47. attēls. Maksājuma detalizēts skats

5) Lai pārbaudīt vai ID tiešām ir nokopēts, varat ievietot to kāda teksta redaktorā (Skat. 24., 25., 26. attēlu).

5.6. Paroles maiņa

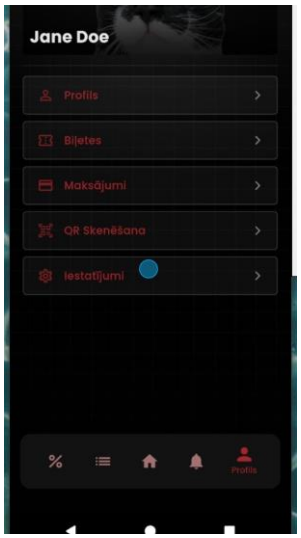
Šeit būs paskaidrots:

- Kā var pāriet uz iestatījumu sadaļu;
- Kā var nomainīt paroli.

1) Nospiediet uz profila sadaļas pogas;

48. attēls. Galvenais skats, pāreja uz profila sadaļu

2) Nospiediet uz iestatījumu sadaļas pogas;



49. attēls. Pāreja uz iestatījumu sadaļu

3) Nospiediet "Mainīt paroli";

50. attēls. Pāreja uz paroles maiņas skatu

4) Ievadiet datus un nospiediet "Mainīt paroli". Parole tiek atjaunota.

51. attēls. Paroles maiņa

6. Testēšanas dokumentācija

6.1. Izvēlētās testēšanas metodes, rīku apraksts un pamatojums

Testēšana ir būtisks mobilās aplikācijas "Filmu Nams" izstrādes posms, kas nodrošina produkta kvalitāti un funkcionalitātes atbilstību izvirzītajām prasībām. Projekta ietvaros tika izmantotas vairākas testēšanas metodes, kas kopā veidoja visaptverošu testēšanas stratēģiju.

Manuālā testēšana tika izvēlēta kā primārā testēšanas metode, jo tā ļauj novērtēt lietotāja pieredzi un saskarnes ergonomiku reālos lietošanas apstākļos. Manuālās testēšanas procesā tika izmantota strukturēta pieeja:

- Testpiemēru izstrāde (Skat. 14. tabulu.) katrai funkcionālajai prasībai;
- Sistemātiska testpiemēru izpilde dažādās vidēs un ierīcēs;
- Detalizēta kļūdu dokumentēšana (Skat. 15. tabulu.) trūkumu novēršanai;
- Atkārtota testēšana pēc kļūdu labojumiem.

Šī metode ļāva efektīvi identificēt vizuālās un funkcionalitātes problēmas, kas nebūtu viegli atklājamas automātiskajos testos.

Ņemot vērā aplikācijas starpplatformu raksturu, testēšana tika veikta uz dažādām fiziskām ierīcēm:

- Android ierīcēm;
- iOS ierīcēm;
- Windows un macOS datoriem administratoru paneļa testēšanai.

Testēšana uz reālām ierīcēm bija nepieciešama, lai pārliecinātos par aplikācijas korektu darbību dažādos ekrāna izmēros, izšķirtspējās un operētājsistēmu versijās, kā arī identificētu ar ierīci specifiskās problēmas, kas var netikt atklātas emulatorā.

6.2. Testpiemēru kopa

14.tabula

Fragments no testpiemēru kopas

ID	Nosaukums	Izpildes nosacījumi	Apraksts	Izpildes soļi	Ievades dati	Sagaidāmais rezultāts	Prasības ID
Black Box							
TP.KL.01	Klientu aplikācijas palaišana	Lietotājam ir uzinstalēta klientu aplikācija uz iOS ierīces	Tiek pārbaudīts, vai aplikācija tiek palaista, nospiežot uz tās ikonas	Ar pirkstu pieskarties pie aplikācijas ikonas	Pieskāriens uz aplikācijas ikonas	Klientu aplikācijas palaišana, redzams ielogošanas vai sākuma skats	PR.KL.01

TP.KL.02	Klientu aplikācijas palaišana	Lietotājam ir uzinstalēta klientu aplikācija uz Android ierīces	Tiek pārbaudīts, vai aplikācija tiek palaista, nospiežot uz tās ikonas	Ar pirkstu pieskarties pie aplikācijas ikonas	Pieskāriens uz aplikācijas ikonas	Klientu aplikācijas palaišana, redzams ielogošanas vai sākuma skats	PR.KL.01
TP.KL.03	Klientu aplikācijas palaišana	Lietotājam ir uzinstalēta klientu aplikācija uz iOS ierīces	Tiek pārbaudīts, vai aplikācija tiek palaista, nospiežot un turot pirksti uz tās ikonas	Ar pirkstu pieskarties un turēt vismaz 1,5 sekundes uz aplikācijas ikonas	Ilgstošs pieskāriens uz aplikācijas ikonas	Klientu aplikācija netiek palaista, tiek radīts aplikācijas opcijas	PR.KL.01
TP.KL.04	Klientu aplikācijas palaišana	Lietotājam ir uzinstalēta klientu aplikācija uz Android ierīces	Tiek pārbaudīts, vai aplikācija tiek palaista, nospiežot un turot pirksti uz tās ikonas	Ar pirkstu pieskarties un turēt vismaz 1,5 sekundes uz aplikācijas ikonas	Ilgstošs pieskāriens uz aplikācijas ikonas	Klientu aplikācija netiek palaista, tiek radīts aplikācijas opcijas	PR.KL.01
TP.AD.01	Administratoru aplikācijas palaišana	Lietotājam ir uzinstalēta klientu aplikācija uz macOS ierīces	Tiek pārbaudīts, vai aplikācija tiek palaista, nospiežot uz tās ikonas	Ar peles kreiso klikšķi nospiest uz aplikācijas ikonas	Peles kreisais klikšķis uz aplikācijas ikonas	Administratoru aplikācijas palaišana, redzams ielogošanas vai sākuma skats	PR.AD.01
TP.AD.02	Administratoru aplikācijas palaišana	Lietotājam ir uzinstalēta klientu aplikācija uz Windows ierīces	Tiek pārbaudīts, vai aplikācija tiek palaista, nospiežot uz tās ikonas	Ar peles kreiso klikšķi nospiest uz aplikācijas ikonas	Peles kreisais klikšķis uz aplikācijas ikonas	Administratoru aplikācijas palaišana, redzams ielogošanas vai sākuma skats	PR.AD.01
TP.AD.03	Administratoru aplikācijas palaišana	Lietotājam ir uzinstalēta klientu aplikācija uz macOS ierīces	Tiek pārbaudīts, vai aplikācija tiek palaista, nospiežot un turot pirksti uz tās ikonas	Ar peles labo klikšķi nospiest uz aplikācijas ikonas	Peles labais klikšķis uz aplikācijas ikonas	Administratoru aplikācija netiek palaista, tiek radīts aplikācijas opcijas	PR.AD.01
TP.AD.04	Administratoru aplikācijas palaišana	Lietotājam ir uzinstalēta klientu aplikācija uz Windows ierīces	Tiek pārbaudīts, vai aplikācija tiek palaista, nospiežot un turot pirksti uz tās ikonas	Ar peles labo klikšķi nospiest uz aplikācijas ikonas	Peles labais klikšķis uz aplikācijas ikonas	Administratoru aplikācija netiek palaista, tiek radīts aplikācijas opcijas	PR.AD.01
TP.KL.LOG.01	Veiksmīga ielogošanās klientu aplikācijā, izmantojot e-pastu un paroli	Lietotājam ir uzinstalēta klientu aplikācija uz iOS vai Android ierīces	Tiek pārbaudīts, vai lietotājs var ielogoties aplikācijā izmantojot eksistējošo kontu e-pastu un paroli	Atvērt klientu aplikāciju, ielogošanas formā ievadīt e-pastu un paroli, apstiprināt ar pieskārienu uz pogas "Ielogoties"	Pieskāriens formas laukiem, datu (e-pasts, parole) ievadīšana ar ierīces pieejamo tastatūru, pieskāriens uz pogas "Ielogoties"	Klients tiek ielogots iekšā un aplikācijas skats mainās uz galveno sadaļu	PR.KL.LOG.01

TP.KL.LOG.02	Neveiksmīga ielogošanās klientu aplikācijā, izmantojot e-pastu un paroli	Lietotājam ir uzinstalēta klientu aplikācija uz iOS vai Android ierīces	Tiek pārbaudīts, vai lietotājs nedrīkst ielogoties aplikācijā izmantojot nederīgo e-pastu un/vai paroli	Atvērt klientu aplikāciju, ielogošanas formā ievadīt e-pastu un paroli, apstiprināt ar pieskārienu uz pogas "Ielogoties"	Pieskāriens formas laukiem, datu (e-pasts, parole) ievadīšana ar ierīces pieejamo tastatūru, pieskāriens uz pogas "Ielogoties"	Klients netiek ielogots iekšā un radās paziņojums par kļūdu	PR.KL.LOG.01
TP.KL.LOG.03	Veiksmīga ielogošanās klientu aplikācijā, izmantojot Google kontu	Lietotājam ir uzinstalēta klientu aplikācija uz iOS vai Android ierīces	Tiek pārbaudīts, vai lietotājs var ielogoties aplikācijā izmantojot Google kontu	Atvērt klientu aplikāciju, ielogošanas formā nospiest uz pogas "Google", ielogoties ar Google kontu	Pieskaries pogai "Google"	Klients tiek ielogots iekšā un aplikācijas skats mainās uz galveno sadaļu	PR.KL.LOG.02
TP.KL.LOG.04	Neveiksmīga ielogošanās klientu aplikācijā, izmantojot Google kontu	Lietotājam ir uzinstalēta klientu aplikācija uz iOS vai Android ierīces	Tiek pārbaudīts, vai lietotājs nedrīkst ielogoties aplikācijā atceļot Google ielogošanās darbību	Atvērt klientu aplikāciju, ielogošanas formā nospiest uz pogas "Google", ielogoties ar Google kontu	Pieskaries pogai "Google"	Klients netiek ielogots iekšā un radās paziņojums par darbības atcelšanu	PR.KL.LOG.02
TP.KL.LOG.05	Veiksmīga ielogošanās klientu aplikācijā, izmantojot Facebook kontu	Lietotājam ir uzinstalēta klientu aplikācija uz iOS vai Android ierīces	Tiek pārbaudīts, vai lietotājs var ielogoties aplikācijā izmantojot Facebook kontu	Atvērt klientu aplikāciju, ielogošanas formā nospiest uz pogas "Facebook", ielogoties ar Facebook kontu	Pieskaries pogai "Facebook"	Klients tiek ielogots iekšā un aplikācijas skats mainās uz galveno sadaļu	PR.KL.LOG.03
TP.KL.LOG.06	Neveiksmīga ielogošanās klientu aplikācijā, izmantojot Facebook kontu	Lietotājam ir uzinstalēta klientu aplikācija uz iOS vai Android ierīces	Tiek pārbaudīts, vai lietotājs var ielogoties aplikācijā izmantojot Facebook kontu	Atvērt klientu aplikāciju, ielogošanas formā nospiest uz pogas "Facebook", ielogoties ar Facebook kontu	Pieskaries pogai "Facebook"	Klients netiek ielogots iekšā un radās paziņojums par darbības atcelšanu	PR.KL.LOG.03
TP.KL.REG.01	Veiksmīga reģistrēšanās klientu aplikācijā	Lietotājam ir uzinstalēta klientu aplikācija uz iOS vai Android ierīces	Tiek pārbaudīts, vai lietotājs var reģistrēties aplikācijā	Aplikācijā pāriet uz reģistrācijas formu, ievadīt e-pastu un paroli, apstiprināt ar pieskārienu uz pogas "Reģistrēties"	Pieskāriens formas laukiem, datu (e-pasts, parole) ievadīšana ar ierīces pieejamo tastatūru, pieskāriens uz pogas "Reģistrēties"	Klients tiek ielogots iekšā un aplikācijas skats mainās uz galveno sadaļu	PR.KL.REG.01

TP.KL.REG.02	Neveiksmīga reģistrēšanās klientu aplikācijā	Lietotājam ir uzinstalēta klientu aplikācija uz iOS vai Android ierīces	Tiek pārbaudīts, vai lietotājs nedrīkst reģistrēties aplikācijā izmantojot nederīgo e-pastu un/vai paroli	Atvērt klientu aplikāciju, pāriet uz reģistrācijas formu, reģistrēšanās formā ievadīt e-pastu un paroli, apstiprināt ar pieskārienu uz pogas "Reģistrēties"	Pieskāriens formas laukiem, datu (e-pasts, parole) ievadīšana ar ierīces pieejamo tastatūru, pieskāriens uz pogas "Reģistrēties"	Klients netiek ielogots iekšā un radās paziņojums par kļūdu	PR.KL.REG.01
TP.AD.LOG.01	Veiksmīga ielogošanās administratoru aplikācijā	Lietotājam ir uzinstalēta administratoru aplikācija uz macOS vai Windows ierīces	Tiek pārbaudīts, vai lietotājs var ielogoties aplikācijā izmantojot eksistējošo kontu e-pastu un paroli, kuram ir administratora tiesības	Atvērt administratoru aplikāciju, ielogošanas formā ievadīt e-pastu un paroli, apstiprināt ar pieskārienu uz pogas "Ielogoties"	Pieskāriens formas laukiem, datu (e-pasts, parole) ievadīšana ar ierīces pieejamo tastatūru, pieskāriens uz pogas "Ielogoties"	Administrators tiek ielogots iekšā un aplikācijas skats mainās uz galveno sadaļu	PR.AD.LOG.01
TP.AD.LOG.02	Neveiksmīga ielogošanās administratoru aplikācijā	Lietotājam ir uzinstalēta administratoru aplikācija uz macOS vai Windows ierīces	Tiek pārbaudīts, vai lietotājs nedrīkst ielogoties aplikācijā izmantojot nederīgo (parasta lietotāja) kontu, vai nepareizo e-pastu un/vai paroli	Atvērt administratoru aplikāciju, ielogošanas formā ievadīt e-pastu un paroli, apstiprināt ar pieskārienu uz pogas "Ielogoties"	Pieskāriens formas laukiem, datu (e-pasts, parole) ievadīšana ar ierīces pieejamo tastatūru, pieskāriens uz pogas "Ielogoties"	Administrators netiek ielogots iekšā un radās paziņojums par kļūdu	PR.AD.LOG.01

TP.AD.FIL.IZV.01	Veiksmīga filmas pievienošana administrācijas sadaļā	Lietotājs ir ielogots administratoru aplikācijā uz macOS vai Windows ierīces	Tiek pārbaudīts, vai lietotājs var pievienot jauno filmu administrācijas panelī	Aplikācijā pāriet uz filmu sadaļu, nospiest uz pogas "Pievienot", formā ievadīt vajadzīgus datus, apstiprināt	Pieskāriens formas laukiem, datu ievadīšana ar ierīces pieejamo tastatūru, pieskāriens uz pogas "Saglabāt"	Jaunās filmas dati tiek saglabāti datu bāze, izveides forma pazūd un atkal ir redzams filmu saraksts, kur pievienojies jaunā filma	PR.AD.FIL.IZV.01
TP.AD.FIL.IZV.02	Neveiksmīga filmas pievienošana administrācijas sadaļā	Lietotājs ir ielogots administratoru aplikācijā uz macOS vai Windows ierīces	Tiek pārbaudīts, vai lietotājs nedrīkst pievienot jauno filmu administrācijas panelī, ja ievadītie dati nav derīgi	Aplikācijā pāriet uz filmu sadaļu, nospiest uz pogas "Pievienot", formā ievadīt vajadzīgus datus, apstiprināt	Pieskāriens formas laukiem, datu ievadīšana ar ierīces pieejamo tastatūru, pieskāriens uz pogas "Saglabāt"	Paradās paziņojums par nederīgiem datiem, filma netiek saglabāta	PR.AD.FIL.IZV.01

6.3. Testēšanas žurnāls

15.tabula

Fragments no testēšanas žurnāla

ID	Datums	Testpiemēra ID	Testpiemēra nosaukums	Testētājs	Statuss	Kļūdas paziņojums	Kļūdas ziņojuma Nr.
Black Box							
TZ.01	2025.05.13.	TP.KL.01	Klientu aplikācijas palaišana	Sofija Dišlovaja	Veiksmīgs		
TZ.02	2025.05.13.	TP.KL.02	Klientu aplikācijas palaišana	Sofija Dišlovaja	Veiksmīgs		
TZ.03	2025.05.13.	TP.KL.03	Klientu aplikācijas palaišana	Sofija Dišlovaja	Veiksmīgs		
TZ.04	2025.05.13.	TP.KL.04	Klientu aplikācijas palaišana	Sofija Dišlovaja	Veiksmīgs		
TZ.05	2025.05.14.	TP.AD.01	Administratoru aplikācijas palaišana	Sofija Dišlovaja	Veiksmīgs		
TZ.06	2025.05.14.	TP.AD.02	Administratoru aplikācijas palaišana	Sofija Dišlovaja	Veiksmīgs		
TZ.07	2025.05.15.	TP.AD.03	Administratoru aplikācijas palaišana	Sofija Dišlovaja	Veiksmīgs		
TZ.08	2025.05.15.	TP.AD.04	Administratoru aplikācijas palaišana	Sofija Dišlovaja	Veiksmīgs		
TZ.09	2025.05.15.	TP.KL.LOG.01	Veiksmīga ielogošanās klientu aplikācijā, izmantojot e-pastu un paroli	Sofija Dišlovaja	Veiksmīgs		
TZ.10	2025.05.15.	TP.KL.LOG.02	Neveiksmīga ielogošanās klientu aplikācijā, izmantojot e-pastu un paroli	Sofija Dišlovaja	Veiksmīgs		
TZ.11	2025.05.15.	TP.KL.LOG.03	Veiksmīga ielogošanās klientu aplikācijā, izmantojot Google kontu	Sofija Dišlovaja	Veiksmīgs		
TZ.12	2025.05.15.	TP.KL.LOG.04	Neveiksmīga ielogošanās klientu aplikācijā, izmantojot Google kontu	Sofija Dišlovaja	Veiksmīgs		
TZ.13	2025.05.15.	TP.KL.LOG.05	Veiksmīga ielogošanās klientu aplikācijā, izmantojot Facebook kontu	Sofija Dišlovaja	Veiksmīgs		

TZ.14	2025.05.15.	TP.KL.LOG.06	Neveiksmīga ielogošanās klientu aplikācijā, izmantojot Facebook kontu	Sofija Dišlovaja	Veiksmīgs		
TZ.15	2025.05.16.	TP.KL.REG.01	Veiksmīga reģistrēšanās klientu aplikācijā	Sofija Dišlovaja	Veiksmīgs		
TZ.16	2025.05.16.	TP.KL.REG.02	Neveiksmīga reģistrēšanās klientu aplikācijā	Sofija Dišlovaja	Veiksmīgs		
TZ.17	2025.05.16.	TP.AD.LOG.01	Veiksmīga ielogošanās administratoru aplikācijā	Sofija Dišlovaja	Veiksmīgs		
TZ.18	2025.05.16.	TP.AD.LOG.02	Neveiksmīga ielogošanās administratoru aplikācijā	Sofija Dišlovaja	Veiksmīgs		
TZ.19	2025.05.16.	TP.AD.FIL.IZV.01	Veiksmīga filmas pievienošana administrācijas sadaļā	Sofija Dišlovaja	Veiksmīgs		
TZ.20	2025.05.16.	TP.AD.FIL.IZV.02	Neveiksmīga filmas pievienošana administrācijas sadaļā	Sofija Dišlovaja	Veiksmīgs		

7. Secinājumi

Projekta "Filmu Nams" izstrādes procesā ir izdevies izveidot pilnvērtīgu un funkcionālu mobilo aplikāciju kinoteātra darbības nodrošināšanai, kas atbilst sākotnēji izvirzītajām prasībām un mērķiem. Izstrādes gaitā tika iegūta vērtīga pieredze un izdarīti vairāki secinājumi.

Izvēlētā tehniskā risinājuma kombinācija – Flutter, Dart un Firebase – izrādījās vispiemērotākā šāda veida projekta realizācijai. Flutter nodrošināja vienas koda bāzes izmantošanu gan iOS, gan Android platformām, kas būtiski samazināja izstrādes laiku un izmaksas. Savukārt Firebase nodrošināja visas nepieciešamās servera puses funkcionalitātes, tostarp datu glabāšanu, autentifikāciju un failu pārvaldību, bez nepieciešamības veidot atsevišķu back-end infrastruktūru.

Izstrādes procesa laikā tika sastapti vairāki izaicinājumi:

1) Biļešu rezervēšanas procesa izveide, nodrošinot korektus sēdvietu izvēles un rezervācijas mehānismus.
2) Maksājumu sistēmas integrācija, veidojot drošu un uzticamu maksājumu apstrādi.
3) Dažādu autentifikācijas metožu (e-pasts/parole, Google, Facebook) implementācija un testēšana.
4) Saskarnes elementu pielāgošana dažāda izmēra un proporciju ekrāniem.
5) Administrācijas paneļa funkcionalitāte, kas prasa stingru piekļuves kontroli un daudzveidīgas datu pārvaldības iespējas.

Šīs problēmas tika veiksmīgi atrisinātas, un rezultātā lietotne piedāvā visas nepieciešamās funkcijas gan klientiem, gan administrācijai:

- Klientiem: filmu kataloga pārlūkošana, seansu saraksta apskatīšana, biļešu iegāde, profila pārvaldība, paziņojumu saņemšana, piedāvājumu izmantošana

- Administrācijai: filmu, seansu, lietotāju, piedāvājumu un promokodu pārvaldība, statistikas pārskati, maksājumu uzskaite

Aplikācijas struktūra ir veidota atbilstoši MVC (Model-View-Controller) arhitektūras principiem, kas nodrošina koda modularitāti un vieglu uzturēšanu. Datu modelēšanai tika izvēlēta NoSQL pieeja, izmantojot Firebase Firestore, kas sniegs elastīgumu turpmākai funkcionalitātes paplašināšanai.

Aplikācijas testēšana uz dažādām ierīcēm apliecināja, ka lietotne darbojas stabili un nodrošina konsekventu lietotāja pieredzi neatkarīgi no ierīces modeļa vai operētājsistēmas versijas.

Projekta turpmākās attīstības iespējas ir plašas un varētu ietvert:

1) Personalizētu filmu ieteikumu algoritmus, balstoties uz lietotāja skatīšanās vēsturi;
2) Lojalitātes programmas ieviešanu ar punktu uzkrāšanu un papildus bonusiem;

3) Paplašinātu analītikas funkcionalitāti biznesam, ļaujot analizēt apmeklējumu tendences un finansiālos rādītājus;

Šis projekts ir demonstrējis, ka mūsdienu izstrādes rīki un tehnoloģijas, piemēram, Flutter un Firebase, ļauj relatīvi īsā laikā izveidot profesionālu un funkcionālu mobilo lietotni. Projekta izstrāde ir sniegusi neatsveramu pieredzi visās mobilās izstrādes jomās – no UI/UX projektēšanas līdz servera integrācijai un maksājumu apstrādei.

"Filmu Nams" aplikācija ir ne tikai tehnisks risinājums, bet arī potenciāls biznesa instruments, kas var uzlabot kinoteātra pakalpojumu pieejamību, palielināt klientu apmierinātību un radīt jaunas ienākumu iespējas.

8. Lietoto terminu un saīsinājumu skaidrojumi

15. tabula

Terminu un saīsinājumu skaidrojumi

Saīsinājums vai termins	Paskaidrojums
API	Application Programming Interface - programmēšanas saskarne, kas ļauj dažādām programmatūras komponentēm savstarpēji mijiedarboties. Projektā izmantota TMDB API filmu informācijas iegūšanai.
Cloud Firestore	Google Firebase platformas NoSQL dokumentu orientēta datubāze, kas izmantota lietotnes datu glabāšanai.
Cloud Functions	Serverless skripti, kas tiek izpildīti Google mākoņvides infrastruktūrā. Projektā izmantoti specifiskiem uzdevumiem, piemēram, administratora lomu piešķiršanai.
Dart	Programmēšanas valoda, ko izmanto Flutter aplikāciju izstrādei.
ER diagramma	Entity Relationship diagramma - datu modelēšanas rīks, kas parāda entītijas (tabulas) un to savstarpējās attiecības.
Firebase	Google mākoņa platforma, kas piedāvā dažādus rīkus un pakalpojumus mobilajām lietotnēm, ieskaitot datubāzi, autentifikāciju un failu glabātuvi.
Firebase Authentication	Autentifikācijas pakalpojums, kas ļauj lietotājiem ielogoties lietotnē ar e-pastu/paroli, Google, Facebook vai citiem kontiem.
Firebase Storage	Failu glabāšanas pakalpojums, kas tiek izmantots attēlu un citu mediju failu uzglabāšanai.
Flutter	Google izstrādāta atvērtā pirmkoda UI izstrādes komplekts, kas ļauj veidot starpplatformu lietotnes no viena koda bāzes.
IOS	Apple Inc. izstrādāta mobilā operētājsistēma, kas darbojas iPhone ierīcēs.
Administrācijas panelis	Lietotnes daļa, kas paredzēta kinoteātra darbiniekiem un administratoriem, ļaujot pārvaldīt filmas, seansus, lietotājus un citu saturu.
Logrīks (Widget)	Mazs lietotnes elements, kas tiek novietots ierīces sākuma ekrānā un parāda aktuālo informāciju bez nepieciešamības atvērt lietotni. Projektā izmantots biļešu attēlošanai.
MVC	Model-View-Controller - programmatūras arhitektūras modelis, kas sadala lietotni trīs savstarpēji saistītās daļās, lai nodalītu iekšējo loģiku no lietotāja saskarnes.
NoSQL	"Not Only SQL" - datubāzu kategorija, kas piedāvā mehānismu datu glabāšanai un iegūšanai, atšķirīgu no tradicionālajām relāciju datubāzēm. Projektā izmantota Firebase Firestore.
Promokods	Īpašs kods, ko lietotāji var ievadīt, lai saņemtu atlaidi biļešu pirkumam.
RBAC	Role-Based Access Control - piekļuves kontroles pieeja, kas balstās uz lietotāju lomām. Projektā izmantota, lai nodalītu administratoru un parasto lietotāju iespējas.
Stripe	Maksājumu apstrādes platforma, kas projektā izmantota biļešu pirkumu apstrādei.
TMDB	The Movie Database - filmu datubāze, kuras API izmanto sistēma, lai iegūtu informāciju par filmām.
UI/UX	User Interface/User Experience - lietotāja saskarne un lietotāja pieredze, elementi, kas attiecas uz lietotnes vizuālo izskatu un lietošanas ērtumu.
Karūseļa elementi	Sākuma ekrāna rotējošie attēli, kas parāda aktuālās filmas vai piedāvājumus.
Push paziņojums	Paziņojums, kas tiek nosūtīts mobilajai ierīcei, pat ja lietotne nav aktīvi izmantota.
Testpiemērs	Konkrēts testēšanas scenārijs ar noteiktiem ievaddatiem un sagaidāmiem rezultātiem.

9. Literatūras un informācijas avotu saraksts

1. Flutter dokumentācija [tiešsaiste]. [Skatīts 03.05.2025.]. Pieejams: https://docs.flutter.dev/
2. Firebase dokumentācija [tiešsaiste]. [Skatīts 03.05.2025.]. Pieejams: https://firebase.google.com/docs
3. Kas ir MVC [tiešsaiste]. [Skatīts 07.05.2025.]. Pieejams: https://developer.mozilla.org/en-US/docs/Glossary/MVC
4. Kas ir RBAC [tiešsaiste]. [Skatīts 08.05.2025.]. Pieejams: https://auth0.com/docs/manage-users/access-control/rbac

Pielikumi

1. pielikums

Populāra aktuāla seansa iegūšana no datu bāzes

```dart
// Funkcija, kas atgriež populārāko seansu (to, kam pārdots visvairāk biļešu) un biļešu skaitu
Future<Map<String, Object>?> getMostPopularSchedule() async {
  try {
    final now = DateTime.now(); // Pašreizējais datums un laiks
    final currentTimestamp = Timestamp.fromDate(
        now); // Konvertējam to uz Firestore Timestamp formātu

    // Iegūst visus biļešu dokumentus no Firestore
    final ticketsSnapshot = await _firestore.collection('tickets').get();

    Map<String, int> scheduleCountMap =
        {}; // Izveido objektu, lai skaitītu biļetes katram seansam
    //(String - seansa ID, int - biļešu skaits)

    List<Future<void>> scheduleFutures =
        []; // Saraksts, kurā glabāt nākotnes uzdevumus (Futures) datu apstrādei

    for (var ticketDoc in ticketsSnapshot.docs) {
      // Izējam cauri visām biļetēm

      // Iegūst atsauci uz seansa dokumentu, kas saistīts ar šo biļeti
      final scheduleRef = ticketDoc.data()['schedule'] as DocumentReference;

      final scheduleId = scheduleRef.id; // Iegūst seansa dokumenta ID

      // Pievieno jaunu nepilno objektu sarakstam, lai apstrādātu šo seansu
      scheduleFutures.add(scheduleRef.get().then((scheduleDoc) {
        final scheduleData = scheduleDoc.data()
            as Map<String, dynamic>?; // Iegūst seansa datus no dokumenta
        final scheduleTime =
            scheduleData?['time'] as Timestamp?; // Iegūst seansa laiku

        if (scheduleTime != null &&
            scheduleTime.compareTo(currentTimestamp) > 0) {
          // Pārbauda, vai seansa laiks ir nākotnē

          // Palielina skaitu šim seansam kartē (vai iestata 1, ja tas nav vēl kartē)
          scheduleCountMap[scheduleId] =
              (scheduleCountMap[scheduleId] ?? 0) + 1;
        }
      }));
    }
    await Future.wait(
        scheduleFutures); // Gaida, līdz visi seansa datu iegūšanas uzdevumi ir pabeigti
    if (scheduleCountMap.isEmpty) {
      // Ja nav atrasts neviens nākotnes seanss ar biļetēm, atgriež null
      return null;
    }

    String mostPopularScheduleId =
        scheduleCountMap.entries // Atrod seansa ID ar lielāko biļešu skaitu
            .reduce((a, b) => a.value > b.value ? a : b)
            .key;

    // Iegūst populārākā seansa dokumentu no Firestore
    final scheduleDoc = await _firestore
        .collection('schedule')
        .doc(mostPopularScheduleId)
        .get();

    if (!scheduleDoc.exists) return null; // Pārbauda, vai dokuments eksistē

    final schedule = await ScheduleModel.fromMapAsync(
      scheduleDoc.data()!,
      scheduleDoc.id,
    );

    // Atgriež objektu ar populārāko seansu un biļešu skaitu
    return {
      'schedule': schedule,
      'count': scheduleCountMap[mostPopularScheduleId] ?? 0,
    };
  } catch (e) {
    debugPrint(
        'Error getting most popular schedule: $e'); // Kļūdas gadījumā izvada kļūdas paziņojumu un atgriež null
    return null;
  }
}
```

2. pielikums

Logrīka atjaunošana ar jaunām biļetēm

```dart
static Future<void> updateTicketsWidget() async {  // Statiska funkcija, kas atjaunina biļešu logrīku (widget)
  try {  // Sāk kļūdu apstrādes bloku
    final ticketController = TicketController();  // Izveido biļešu kontrolieri
    final tickets = await ticketController.getUserTickets();  // Iegūst lietotāja biļetes
    debugPrint('TicketWidgetManager: Found ${tickets.length} tickets total');  // Izvada atrasto biļešu skaitu atkļūdošanai
    if (tickets.isEmpty) {  // Pārbauda, vai biļešu saraksts ir tukšs
      await HomeWidget.saveWidgetData(ticketsDataKey, "[]");  // Saglabā tukšu masīvu logrīkam
      await HomeWidget.saveWidgetData(ticketsCountKey, 0);  // Iestatīta biļešu skaitu uz 0
      await HomeWidget.saveWidgetData(lastUpdatedKey, DateTime.now().millisecondsSinceEpoch);  // Saglabā pēdējo atjaunināšanas laiku
      debugPrint('TicketWidgetManager: Saved empty ticket data to widget');  // Izdrukā paziņojumu par tukšu datu saglabāšanu
      await HomeWidget.updateWidget(  // Atjaunina logrīku abām platformām
        iOSName: iOSWidgetName,  // iOS logrīka nosaukums
        androidName: androidWidgetName,  // Android logrīka nosaukums
      );
      return;  // Beidz funkciju, ja nav biļešu
    }
    tickets.sort((a, b) => b.purchaseDate.compareTo(a.purchaseDate));  // Sakārto biļetes pēc pirkuma datuma dilstošā secībā
    final today = DateTime.now();  // Iegūst pašreizējo datumu
    final startOfDay = DateTime(today.year, today.month, today.day, 0, 0, 0);  // Iegūst šodienas sākuma datumu un laiku
    final recentTickets = tickets.where((ticket) => ticket.schedule.time.toDate().isAfter(startOfDay)).take(3).toList();  // Atlasa 3 nākotnes biļetes
    final ticketsData = recentTickets.map((ticket) => {  // Pārveido biļetes par JSON struktūru
          'id': ticket.id,  // Pievieno biļetes ID
          'movieTitle': ticket.schedule.movie.title,  // Pievieno filmas nosaukumu
          'posterUrl': ticket.schedule.movie.posterUrl,  // Pievieno plakāta URL
          'date': ticket.schedule.time.toDate().millisecondsSinceEpoch,  // Pievieno datumu milisekundēs
          'hall': ticket.schedule.hall,  // Pievieno zāles numuru
          'seat': '${ticket.seat['row'] + 1}-${ticket.seat['seat'] + 1}',  // Pievieno sēdvietas informāciju
          'formattedTime': ticket.getFormattedShowTime(),  // Pievieno formatētu laiku
          'formattedDate': ticket.getFormattedShowDate(),  // Pievieno formatētu datumu
        }).toList();

    final ticketsJson = jsonEncode(ticketsData);  // Konvertē datus uz JSON tekstu
    debugPrint('TicketWidgetManager: JSON data to save: $ticketsJson');  // Izdrukā saglabājamos JSON datus

    await HomeWidget.saveWidgetData(ticketsDataKey, ticketsJson);  // Saglabā biļešu datus
    await HomeWidget.saveWidgetData(ticketsCountKey, recentTickets.length);  // Saglabā biļešu skaitu
    await HomeWidget.saveWidgetData(lastUpdatedKey, DateTime.now().millisecondsSinceEpoch);  // Saglabā pēdējo atjaunināšanas laiku

    await HomeWidget.updateWidget(  // Atjaunina logrīku
      iOSName: iOSWidgetName,  // iOS logrīka nosaukums
      androidName: androidWidgetName,  // Android logrīka nosaukums
    );

    debugPrint('TicketWidgetManager: Widget updated with ${recentTickets.length} tickets');  // Izvada paziņojumu par veiksmīgu atjaunināšanu
  } catch (e, stackTrace) {  // Uztver kļūdas un steka izsekojumu
    debugPrint('Failed to update tickets widget: $e');  // Izvada kļūdas ziņojumu
    debugPrint('Stack trace: $stackTrace');  // Izvada steka izsekojumu
  }
}
```

3. pielikums.

Maksājuma apstrāde

```
Future<bool> processPayment({  // Funkcija maksājuma apstrādei, atgriež boolean vērtību par maksājuma statusu
  required BuildContext context,  // Prasītais konteksts UI elementiem
  required double amount,  // Maksājuma summa
  required String currency,  // Valūta (piem., EUR)
  required String description,  // Maksājuma apraksts
  String? customerEmail,  // Klienta e-pasts
  required String scheduleId,  // Seansu ID, ar kuru saistīts maksājums
}) async {
  final schedule = _firestore.collection('schedule').doc(scheduleId);  // Iegūst atsauci uz seansu Firestore datubāzē
  try {  // Sāk kļūdu apstrādes bloku
    debugPrint("Processing payment of $amount $currency for: $description");  // Izvada atkļūdošanas informāciju
    final paymentIntentResult = await _createPaymentIntent(  // Izveido maksājuma nodomu Stripe platformā
      amount: (amount * 100).toInt(),  // Pārvērš summu centus (Stripe prasība)
      currency: currency,  // Nodod valūtu
      description: description,  // Nodod aprakstu
      customerEmail: customerEmail,  // Nodod klienta e-pastu, ja tas ir norādīts
    );
    if (paymentIntentResult == null) {  // Pārbauda, vai maksājuma nodomu izdevās izveidot
      _showPaymentError(context, "Neizdevās izveidot maksājumu");  // Parāda kļūdas paziņojumu lietotājam
      generateUnsuccessfulHistory(  // Izveido neveiksmīga maksājuma vēsturi datubāzē
        amount: amount,  // Nodod summu
        schedule: schedule,  // Nodod seansu
        reason: 'Failed to create payment intent',  // Norāda kļūdas iemeslu
        product: description,  // Nodod produkta aprakstu
      );
      return false;  // Atgriež false, norādot, ka maksājums neizdevās
    }
    final clientSecret = paymentIntentResult['client_secret'] as String;  // Iegūst klienta slepeno atslēgu no maksājuma nodoma rezultāta
    await Stripe.instance.initPaymentSheet(  // Inicializē Stripe maksājumu lapu
      paymentSheetParameters: SetupPaymentSheetParameters(  // Iestata maksājumu lapas parametrus
        merchantDisplayName: 'Filmu Nams',  // Iestata tirgotāja nosaukumu
        paymentIntentClientSecret: clientSecret,  // Klineta atslēga
        style: ThemeMode.dark,  // Iestata tumšo tēmu
      ),
    );
    await Stripe.instance.presentPaymentSheet();  // Parāda maksājumu lapu lietotājam un gaida darbības rezultātu
    _showPaymentSuccess(context);  // Parāda veiksmīga maksājuma paziņojumu
    return true;  // Atgriež true, norādot, ka maksājums izdevās
  } catch (e) {  // Uztver kļūdas maksājuma apstrādes laikā
    debugPrint("Payment error: $e");  // Izvada kļūdu
    if (e is StripeException) {  // Pārbauda, vai tā ir Stripe kļūda
      generateUnsuccessfulHistory(  // Izveido neveiksmīga maksājuma vēsturi
        amount: amount,  // Nodod summu
        schedule: schedule,  // Nodod seansu
        reason: 'Stripe: ${e.error.localizedMessage}',  // Norāda Stripe kļūdas iemeslu
        product: description,  // Nodod produkta aprakstu
      );
      _showPaymentError(context, "Stripe kļūda: ${e.error.localizedMessage}");  // Parāda Stripe kļūdas paziņojumu
    } else {  // Ja tā ir cita kļūda
      generateUnsuccessfulHistory(  // Izveido neveiksmīga maksājuma vēsturi
        amount: amount,  // Nodod summu
        schedule: schedule,  // Nodod seansu
        reason: "Payment error: $e",  // Norāda vispārīgu kļūdas iemeslu
        product: description,  // Nodod produkta aprakstu
      );
      _showPaymentError(context, "Maksājuma kļūda: $e");  // Parāda vispārīgu kļūdas paziņojumu
    }
    return false;  // Atgriež false, norādot, ka maksājums neizdevās
  }
}
```

4. pielikums

Ielogošanās, izmantojot Facebook Android ierīcēs

```dart
Future<void> signInWithFacebookAndroid() async {  // Funkcija ielogošanai ar Facebook kontu Android ierīcēs
try {  // Sāk kļūdu apstrādes bloku
  final LoginResult result = await FacebookAuth.instance.login(  // Iniciē Facebook pieslēgšanās procesu
    loginBehavior: LoginBehavior.nativeWithFallback,  // Izmanto vietējo lietotni ar atkāpšanos uz tīmekli
    permissions: ['email', 'public_profile'],  // Pieprasītās atļaujas - e-pasts un publiskais profils
  );

  if (result.status == LoginStatus.success) {  // Pārbauda, vai pieslēgšanās ir veiksmīga
    final userData = await FacebookAuth.instance.getUserData();  // Iegūst lietotāja datus no Facebook
    debugPrint('Facebook user data: $userData');  // Izvada lietotāja datus atkļūdošanai

    final accessToken = result.accessToken?.tokenString;  // Iegūst piekļuves tokenu

    if (accessToken == null) {  // Pārbauda, vai tokens eksistē
      throw FirebaseAuthException(  // Izveido un izmet kļūdu, ja tokena nav
        code: 'facebook-auth-token-null',  // Kļūdas kods
        message: 'Facebook access token is null',  // Kļūdas ziņojums
      );
    }

    final facebookAuthCredential =  // Izveido Facebook autentifikācijas kredenciāļus Firebase
        FacebookAuthProvider.credential(accessToken);  // Izmanto iegūto piekļuves tokenu

    final userCredential = await FirebaseAuth.instance  // Piesakās Firebase ar Facebook kredenciāļiem
        .signInWithCredential(facebookAuthCredential);

    final user = userCredential.user;  // Iegūst lietotāja objektu no kredenciāļiem
    if (user != null && user.email != null && user.displayName != null) {  // Pārbauda, vai ir visi nepieciešamie lietotāja dati
      UserDocumentPayload payload = UserDocumentPayload(  // Izveido datu objektu lietotāja dokumentam
        name: user.displayName!,  // Lietotāja vārds
        email: user.email!,  // Lietotāja e-pasts
        profileImage: user.photoURL,  // Lietotāja profila attēls (ja ir)
      );

      await _userController.createUserDocument(user, payload);  // Izveido lietotāja dokumentu datubāzē
    } else {  // Ja trūkst lietotāja datu
      debugPrint(  // Izvada trūkstošos datus atkļūdošanai
          'User data incomplete: ${user?.displayName}, ${user?.email}');
    }
  } else {  // Ja pieslēgšanās nav veiksmīga
    debugPrint('Facebook login failed: ${result.status.toString()}');  // Izvada neveiksmīgās pieslēgšanās statusu
    if (result.message != null) {  // Pārbauda, vai ir kļūdas ziņojums
      debugPrint('Facebook login error message: ${result.message}');  // Izvada kļūdas ziņojumu, ja tāds ir
    }
  }
} catch (e) {  // Uztver visas kļūdas, kas rodas pieslēgšanās procesā
  debugPrint('Error during Facebook sign in: $e');  // Izvada kļūdas ziņojumu
}
}
```