**Praktiskais darbs priekšmetā**

**Programmatūras Inženierija**

**„Spēles *Balloon popping*”**

**Prasību dokuments, Projektējums, Lietotāja ceļvedis un Testēšana**

Izstrādāja:

ak-it-p-ns Kārkliņš Ēriks

Rīga

**Saturs**

[Ievads 4](#_Toc151378141)

[1. Prasību dokumentācijas sadaļa 5](#_Toc151378142)

[1.1. Prasību specifikācija 5](#_Toc151378143)

[1.1.1. Vienkāršots sistēmas modelis 5](#_Toc151378144)

[1.1.2. Sistēmas apkārtnes shēma 6](#_Toc151378145)

[1.1.3. Viedokļu vākšana 7](#_Toc151378146)

[1.1.4. Viedokļu tabulārās diagrammas 7](#_Toc151378147)

[1.1.5. Datu plūsmu diagrammas 9](#_Toc151378148)

[1.2. Datu modelēšana 9](#_Toc151378149)

[1.2.1. Sistēmas E-R modelis 9](#_Toc151378150)

[1.2.2. Datu vārdnīca 10](#_Toc151378151)

[1.3. Sistēmas evolūcija 11](#_Toc151378152)

[1.4. Prasības sistēmai 11](#_Toc151378153)

[1.4.1. Funkcionālās prasības 11](#_Toc151378154)

[1.4.2. Nefunkcionālās prasības 12](#_Toc151378155)

[1.5. Prasību specifikācija 12](#_Toc151378156)

[1.5.1. Funkcionālo prasību specifikācija strukturētajā valodā 12](#_Toc151378157)

[1.5.2. Funkcionālo prasību specifikācija PDL valodā 13](#_Toc151378158)

[2. Projektējuma daļa 15](#_Toc151378159)

[2.1. Dekompozīcijas apraksts 15](#_Toc151378160)

[2.1.1. Moduļu dekompozīcija 15](#_Toc151378161)

[2.1.2. Vienlaicīgo procesu dekompozīcija 17](#_Toc151378162)

[2.1.3. Datu dekompozīcija 18](#_Toc151378163)

[2.2. Atkarību apraksts 19](#_Toc151378164)

[2.2.1. Moduļu struktūra un atkarības 19](#_Toc151378165)

[2.2.2. Starpprocesu atkarības 20](#_Toc151378166)

[2.2.3. Datu atkarības 21](#_Toc151378167)

[2.3. Moduļu saskarnes apraksts 21](#_Toc151378168)

[2.3.1. Moduļu saskarne 21](#_Toc151378169)

[2.4. Detalizētais projektējums 23](#_Toc151378170)

[2.4.1. Moduļu detalizētais projektējums 23](#_Toc151378171)

[2.4.2. Lēmumu tabula 23](#_Toc151378172)

[2.4.3. Pseidokods 23](#_Toc151378173)

[2.4.4. Datu detalizētais projektējums 24](#_Toc151378174)

[3. Lietotāja ceļvedis 26](#_Toc151378175)

[3.1. Spēles apraksts 26](#_Toc151378176)

[3.1.1. Spēles palaišana 26](#_Toc151378177)

[3.1.2. Spēles spēlēšana 28](#_Toc151378178)

[3.1.3. Spēles beigas 29](#_Toc151378179)

[4. Testēšana 30](#_Toc151378180)

[4.1. Testēšanas plāns 30](#_Toc151378181)

[4.2. Ievades dati 31](#_Toc151378182)

[4.3. Testēšanas scenāriju izpilde 32](#_Toc151378183)

[4.4. Kļūdu pārvaldība 33](#_Toc151378184)

[4.5. Dokumentācija 34](#_Toc151378185)

[Secinājumi 35](#_Toc151378186)

[Literatūra 36](#_Toc151378187)

# Ievads

Dotais dokuments ir izstrādāts, lai sniegtu pilnu un skaidru izpratni par spēles *Balloon\_popping* (turpmāk – spēle) mērķiem, darbību, un tehnikām, kuras būs nepieciešamas, lai sasniegtu izstrādātu šo spēli. Dokuments ir domāts kā norādījums gan izstrādātājiem, gan projektu vadībai, lai nodrošinātu vienotu redzējumu par projekta galvenajiem elementiem.

Spēles mērķis:

Spēles galvenais mērķis ir uzspridzināt pēc iespējas vairāk balonus, lai iegūtu maksimālu punktu skaitu. Spēlētājiem būs jāparāda savas reakcijas spējas, noklikšķinot uz parādībās esošajiem baloniem, pirms tie pārvietojas pār ekrānu un pazūd.

Spēle ir aizraujoša un vienkārša interaktīva izklaide, kurā spēlētājiem būs iespēja izbaudīt balonu spridzināšanu gan kā izklaides veids, gan kā izaicinošs uzdevums spēlētājiem, kas vēlas pārbaudīt savas reakcijas spējas un sasniegt maksimālo rezultātu.

Šis projekts ir izstrādāts, izmantojot Windows Presentation Foundation (WPF) tehnoloģiju, piedāvājot vizuāli pievilcīgu un viegli lietojamu spēles pieredzi.

# Prasību dokumentācijas sadaļa

## Prasību specifikācija

### Vienkāršots sistēmas modelis

Modelī būs divi galvenie komponenti - lietotāja saskarne (UI) un spēles dzinējs shematiski 1.

Lietotāja saskarne (UI)

Spēlētājs

Spēles dzinējs

1.1. att. Spēles vienkāršotās sistēmas modelis

Lietotāja saskarne (UI):

Ievieš *MainWindow*, kas ir galvenais loga elements, kurā atrodas vizuālā sastāvdaļa.

II satur *Canvas (MyCanvas*), kur tiek attēlota spēles telpa.

UI reaģē uz taustiņiem (*canvasKeyIsDown*), lai ļautu lietotājam sākt un atiestatīt spēli.

Spēles dzinējs:

Ietver *gameTimer (DispatcherTimer)*, kas regulāri izsauc *gameEngine* funkciju, lai atjauninātu spēles stāvokli.

Pārvalda spēles laiku, noteikto ātrumu, un paziņo par zaudējumu, ja tiek izlaistas pārāk daudz balonu no spēles lauka (*gameEngine funkcija*).

Ģenerē un pārvalda balonus, to kustību un punktu uzskaiti (*startGame, ResetGame, PoPBalloons, gameEngine* funkcijas).

### Sistēmas apkārtnes shēma

Spēles sistēmas apkārtnes shēma ir attēlota 1.2. attēlā.

Lietotāja ierīce

(Datora ekrāns)

Operētājsistēma (OS)

.NET *Runtime*

*(CLR - Common Language Runtime*)

Spēles izpildfails (*WPF\_Balloon\_popping\_game.exe*

WPF Bibliotēkas

Audio un Bildes

Resursi (baloni, fons, skaņas utt.)

Izveido un pārvalda

Iekļauj spēles izpildfailu un vajadzīgās bibliotēkas

Iekļauj un iegūst:

Izmanto:

Satur un izpilda:

1.2. att. Sistēmas apkārtnes shēma

Lietotāja ierīce ir ierīce, kurā tiek palaista spēle (piemēram, dators ar ekrānu).

Operētājsistēma ir platforma, uz kuras tiek palaista spēles izpildfaila, piemēram, Windows vai *Linux*.

Spēles izpildfails ir jūsu izstrādātā WPF spēles izpildfails, kas ietver visu nepieciešamo informāciju, lai palaistu spēli.

*NET Runtime* ir izpildvide, kas nepieciešama, lai palaistu .NET aplikācijas, ieskaitot WPF spēles.

WPF Bibliotēkas ir Windows Presentation Foundation bibliotēkas, kas nodrošina lietotāja saskarnes un vizuālās elementu izveidi.

Audio un Bildes Resursi ietver visus nepieciešamos vizuālos un skaņas resursus, ko izmantojat spēles izveidē.

### Viedokļu vākšana

Spēles viedokļi ir apkopoti hierarhiskā diagrammā, kura ir redzama 1.3. diagrammā.

Lietotāji/ Interesenti

(Spēles lietotāji, izstrādātāji)

Lietotāja saskarne (UI)

Tehniskais izstrādātājs

Spēles testētājs

Spēles dzinējs

Spēles dizains

1.3. att. Viedokļu hierarhiskā struktūra

Lietotāji/ Interesenti: Šī grupa ietver visus spēles lietotājus un izstrādātājus, kuriem ir interese par spēles darbību un funkcionalitāti.

Lietotāja Saskarne (UI): Šeit tiek iekļauts viedoklis par lietotāja saskarni, spēles izskatu un lietojamību. Šajā grupā var ietilpt dizaineri un spēles lietotāji.

Tehniskais Izstrādātājs: Šeit tiek iekļauts viedoklis no tehnikas puses, ietverot sistēmas modeli, koda struktūru un citus aspektus, kas ir svarīgi izstrādātājiem.

Spēles Dizains: Šajā grupā ir dizaineri, kuri vērtē spēles vizuālo izskatu, tekstūras, skaņas un kopējo dizainu.

Spēles Dzinējs: Šeit iekļauts viedoklis no spēles dzinēja izstrādātāja puses, ietverot spēles loģiku, balonu parādīšanas biežumu un citus dzinēja aspektus.

Testētājs: Šī grupa vērtē spēles kvalitāti, veicot testēšanu un identificējot iespējamus defektus.

### Viedokļu tabulārās diagrammas

Zemāk ir attēlotas spēles viedokļu tabulārās diagramma.

1. tabula

Viedokļu tabulārās diagramma

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Avots | Ieeja | Darbība | Izvade | Norīkojuma vieta |
| Lietotāja Saskarne (UI) | Lietotāji/ Interesenti | Spēles Dizains | Spēles Dzinējs | Ekrānu izvade |
| Tehniskais Izstrādātājs | Spēles izpildfails | Koda izstrāde | Spēles Dzinējs | Konsolē vai failā izvadīta informācija |
| Spēles Dizains | Dizaineri | Spēles izskats, tekstūras, skaņas | Spēles Dzinējs | Lietotāja Saskarne (UI) |
| Spēles Dzinējs | Tehniskais Izstrādātājs | Spēles Dizains | Koda izpilde, loģika | Ekranu izvade, skaņas efekti |
| Testētājs | Tehniskais Izstrādātājs | Lietotāja Saskarne (UI) | Spēles Dzinējs | Konsolē vai failā izvadīta testēšanas informācija |

### Datu plūsmu diagrammas

1.4. attēlā ir parādīta datu plūsma starp procesiem spēlei.

Lietotāja ierīce

(Ievades/izvades ierīce)

Ievades apstrāde (UI events)

Spēles loģika

Audio/Vizuālā apstrāde

Datu uzglabāšana un apstrāde

Spēles UI

*(WFS controls)*

1.4. att. Preces datu pievienošanas darbību diagramma

Lietotāja ierīce: Pārstāv datora ekrānu vai citu ierīci, kurā notiek ievade un izvade.

Ievades apstrāde: Veic ievades datu apstrādi, ieskaitot taustiņu nospiešanu, peles kustības, utt.

Spēles UI: Attēlo spēles vizuālo sastāvdaļu, kuras izveide notiek, izmantojot *WPF Controls*.

Spēles Loģika: Iekļauj spēles loģiku, kā baloni parādās un izzūd, kā arī citas spēles mehānikas.

Audio/Vizuālā apstrāde: Nodrošina skaņas un vizuālos efektus, kas atbilst spēles notikumiem.

Datus uzglabāšana un apstrāde: Pārvalda spēles datu uzglabāšanu un apstrādi, piemēram iegūtos punktus.

## Datu modelēšana

### Sistēmas E-R modelis

Lai attēlotu spēles datu modeli izveidoju E-R modeli, kas ir redzams 1.5. attēlā.

Spēles dati

(Game)

Balloons

GameID (PK)

StartTime

EndTime

Score

Ballon ID (PK)

GameID (FK)

BallonType

PossitionX

PossitionY

1.5. att. Sistēmas datu modelis

### Datu vārdnīca

Lai precīzāk aprakstītu E-R modeļa entītijas (norādot atribūtu tipus, nosaukumus un to garumus), izveidoju datu vārdnīcu, kas ir redzama 2. tabulā.

2. tabula

Datu vārdnīca

|  |  |  |  |
| --- | --- | --- | --- |
| Nosaukums | Apraksts | Tips | Garums |
| GameID | Primārā atslēga | Integer | - |
| StartTime | Spēles beigu laiks | DateTime | - |
| EndTime | Spēles beigu laiks | DateTime | - |
| Score | Spēles rezultāts | Integer | - |
| BalloonID | Primārā atslēga | Integer | - |
| GameID | Ārējā atslēga (Game.GameID) | Integer | - |
| BalloonType | Balona veids | Varchar | 255 |
| PositionX | X-koordinātes balonam | Integer | - |
| PositionY | Y- koordinātes balonam | Integer | - |

## Sistēmas evolūcija

Paredzot spēles attīstību pastāv sekojoši spēles evolūcijas virzieni kuri uzskaitīti zemāk:

1. Grafikas un animāciju uzlabošana:

* Palielināt detalizāciju un kvalitāti grafikā un animācijās.
* Izmantot jaunākās tehnoloģijas, piemēram, augstas izšķirtspējas tekstūras, reālistiskus efektus un augstas kvalitātes 3D modelēšanu.

1. Papildu funkcijas un līmenis:

* Pievienot jaunus līmeņus, uzdevumus un izaicinājumus, lai uzturētu spēlētāju interesi.
* Ieviest jaunas funkcijas, piemēram, multi-player režīmus, kooperatīvu spēlēšanu, lielāku pasaules izmēru vai interaktīvāku vidi.

1. Realitātes paplašināšana:

* Izmantot paplašinātās realitātes (AR) vai virtuālās realitātes (VR) tehnoloģijas, lai piedāvātu jaunu un iedvesmojošu spēles pieredzi.
* Ieviest interaktīvu un iesaistošu vidi, kur spēlētāji var sajust spēles notikumus apkārtējā pasaulē.

1. Stāsta un rakstura attīstība:

* Turpināt spēles stāstu, ieviešot jaunus personāžus, intrigu un pavērsienus.
* Ļaut spēlētājiem veikt izvēles, kas ietekmē stāsta attīstību un rezultātus.

1. Platformas un ierīču paplašināšana:

* Pielāgot spēli dažādām platformām un ierīcēm, piemēram, mobilo telefonu, planšetdatoru, datoru un konsoles.
* Ieviest crossplatform spēlēšanas iespējas, lai spēlētāji varētu dalīties ar saviem sasniegumiem visās ierīcēs.

## Prasības sistēmai

### Funkcionālās prasības

1. Spēles sākums un atiestatīšana:
   1. Lai spēli sāktu no jauna, lietotājam ir jāspiež taustiņš "Space".
   2. Spēles atiestatīšana notiek, noņemot esošos balonus un sākot jaunu spēli ar sākotnējiem iestatījumiem.
2. Balonu klikšķināšana:
   1. Lietotājs var noklikšķināt uz baloniem ar peli, lai tos pārtrauktu.
   2. Par katru noklikšķinājumu spēlētājam tiek piešķirts punkts.
3. Balonu parādīšanās un kustība:
   1. Jauna balona parādīšanās notiek ar noteiktu laika intervālu.
   2. Baloni kustas uz augšu ekrāna virzienā ar noteiktu ātrumu.
4. Skaņas efekti:
   1. Kad lietotājs noklikšķina uz balona, tiek atskaņots skaņas efekts ("pop\_sound.mp3").
5. Spēles beigas:
   1. Spēle beidzas, ja spēlētājs izlaiž 10 vai vairāk balonus no spēles lauka.
   2. Paziņojums par spēles beigām tiek parādīts ar iespēju sākt jaunu spēli, nospiežot "Space".
6. Punktu uzskaitīšana un ātruma pieaugums:
   1. Spēlētājam tiek piešķirti punkti par katru noklikšķinājumu uz balona.
   2. Ja spēlētājs sasniedz 20 punktus, balonu kustības ātrums palielinās.
7. Grafiskais interfeiss:
   1. Spēles fons tiek iestatīts ar attēlu ("background-Image.jpg").
   2. Spēles rezultāts tiek parādīts uz ekrāna ar teksta kontroli.

### Nefunkcionālās prasības

1. Saderība un izmēģinājuma iespējas:
   1. Spēle ir izveidota, izmantojot Windows Presentation Foundation (WPF) tehnoloģiju.
   2. Spēli var palaist un izmēģināt uz Windows operētājsistēmas.
2. Prasības produktam
   1. Lietotāja saskarnei jānotiek angļu valodā

## Prasību specifikācija

### Funkcionālo prasību specifikācija strukturētajā valodā

2.tabula

Spēles funkcijas specifikācija

|  |  |  |
| --- | --- | --- |
| F1 | Spēles sākums un atiestatīšana | Lietotājam ir jāspēj sākt jaunu spēli, nospiežot taustiņu "Space". Ja spēlētājs nokavē 10 vai vairāk balonus, ir jābūt iespējai atiestatīt spēli. |
| F2 | Balonu klikšķināšana | Spēlētājam ir jāspēj noklikšķināt uz baloniem ar peli, lai tos pārtrauktu un iegūtu punktus. |
| F3 | Balonu parādīšanās un kustība | Jauna balona parādīšanās notiek ar noteiktu laika intervālu, un baloni kustas uz augšu ekrāna virzienā ar noteiktu ātrumu. |
| F4 | Skaņas efekti | Kad lietotājs noklikšķina uz balona, tiek atskaņots skaņas efekts ("pop\_sound.mp3"). |
| F5 | Spēles beigas | Spēle beidzas, ja spēlētājs izlaiž no spēles lauka 10 vai vairāk balonus, un parādās paziņojums ar iespēju sākt jaunu spēli. |
| F6 | Punktu uzskaitīšana un ātruma pieaugums | Spēlētājam tiek piešķirti punkti par katru noklikšķinājumu uz balona, un ja sasniedz 20 punktus, balonu kustības ātrums palielinās. |
| F7 | Grafiskais interfeiss | Spēles fons tiek iestatīts ar attēlu ("background-Image.jpg"), un spēles rezultāts tiek parādīts ar teksta kontroli. |

### Funkcionālo prasību specifikācija PDL valodā

-- F1: Spēles sākums un atiestatīšana

Procedūra StartGame():

Ja nospiež taustiņu "Space" un spēle nav aktīva:

Izsauc procedūru ResetGame().

-- F2: Balonu klikšķināšana

Procedūra PopBalloon(balloon: Rectangle):

Ja spēle ir aktīva un klikšķināts uz balona:

Atskaņot skaņas efektu "pop\_sound.mp3".

Izņemt balonu no Canvas.

Palielināt rezultātu par vienu.

-- F3: Balonu parādīšanās un kustība

Procedūra SpawnBalloon():

Ja pagājušas noteiktas intervāla sekundes:

Izvēlēties gadījuma skaitli no 1 līdz 5 kā balona krāsas numuru.

Izveidot jaunu balonu ar izvēlēto krāsu un pievienot to Canvas.

Noteikt jaunu intervālu.

Procedūra MoveBalloons():

Katram Canvas pakārtotajam:

Ja pakārtotais ir balons un spēle ir aktīva:

Pārvietot balonu uz augšu par noteiktu ātrumu.

-- F4: Skaņas efekti (Skaņas efekti tiek atskaņoti balonu klikšķināšanas laikā, tāpēc nav nepieciešama papildu PDL specifikācija.)

-- F5: Spēles beigas

Procedūra CheckGameOver():

Ja nokavēti 10 vai vairāk baloni:

Noslēgt spēli.

Izsauc procedūru ShowGameOverMessage().

-- F6: Punktu uzskaitīšana un ātruma pieaugums (Šīs funkcionalitātes iekļaut procedūrās PopBalloon un MoveBalloons.)

F7: Grafiskais interfeiss (PDL valodā nav nepieciešams aprakstīt.)

# Projektējuma daļa

## Dekompozīcijas apraksts

### Moduļu dekompozīcija

Tālāk dots apraksts katram spēles modulim – tā nosaukums, nolūks un funkcijas un uzdevumi.

1. Spēles sākums un inicializācija:

* Apraksts: Šī daļa ietver spēles sākuma un inicializācijas procesus, tostarp spēles loga un laika (*timer*) inicializāciju, fona uzstādīšanu un spēles sākuma stāvokļa iestatīšanu.
* Funkcijas vai uzdevumi:
  + - *MainWindow* konstruktors
    - *Timer* inicializācija
    - Fona uzstādīšana
    - Spēles sākuma stāvokļa iestatīšana

1. Galveno spēles funkciju vadība:

* Apraksts: Šī daļa ietver galveno spēles loģiku un funkcionalitāti, piemēram, balonu parādīšanos, to kustību un spēlētāja darbību apstrādi.
* Funkcijas vai uzdevumi:
  + - *startGame funkcija*
    - *ResetGame funkcija*
    - *gameEngine funkcija*
    - *PoPBalloons funkcija*

1. Balonu ģenerēšana un kustība:

* Apraksts: Šī daļa apakšvadībā atbild par balonu dinamisko parādīšanos un to kustību pa ekrānu.
* Funkcijas vai uzdevumi:
  + - *generateBalloon* funkcija
    - Balonu izvēles un attēlošanas loģika

1. Skaņas efekti:

* Apraksts: Šī daļa nodrošina skaņas efektu atskaņošanu, piemēram, skaņu, kad lietotājs noklikšķina uz balona.
* Funkcijas vai uzdevumi:
  + - *playSound* funkcija

1. Rezultātu un stāvokļa atjaunināšana:

* Apraksts: Šī daļa ir atbildīga par spēles rezultātu un stāvokļa atjaunināšanu, kā arī par ātruma palielināšanos sasniegtā rezultāta gadījumā.
* Funkcijas vai uzdevumi:
  + - *Rezultāta un stāvokļa atjaunināšanas loģika*

Spēles sistēmas projektēšanā tika izmantota metode „Objektu sadalīšanas metode” uz komponentu orientētas programmēšanas bāzes, 2.1.attēlā ir redzama sistēmas hierarhija.

Lietotāja saskarne (UI)

*gameTimer*

*MyCanvas*

*Balloon Logic*

*Sound Logic*

*Sound & stats*

2.1. att. Sistēmas struktūras diagramma

### Vienlaicīgo procesu dekompozīcija

Vienlaicīgo procesu dekompozīcija ir process, kurā tiek sadalīts izpildes process vairākos vienlaicīgos procesos jeb pavedienos (thread). Šāda pieeja ļauj dažādiem uzdevumiem vai darbībām izpildīties vienlaicīgi, tādējādi uzlabojot spēles veiktspēju un reaktivitāti.

Spēles vienlaicīgo procesu dekompozīciju definē sekojoši:

1. Laiks:

* Apraksts: *DispatcherTimer* ir objekts, kas palīdz veikt noteiktas darbības regulāri un vienlaicīgi. Tas tiek izmantots kā taimeris, kas regulāri izsauc spēles dzinēja *(gameEngine)* metodi ar noteiktu intervālu.
* Vienlaicīgais process: *DispatcherTimer* nodrošina veidu, kā regulāri izpildīt kodu noteiktā taimera ritmā, tādējādi radot vienlaicīgu darbību ar spēles *gameEngine*.

1. Balonu kustība:

* Apraksts: Baloni tiek pārvietoti pa ekrānu, un šī darbība notiek vienlaicīgi. Katrs balons darbojas neatkarīgi, kustoties uz augšu un pa labi vai pa kreisi.
* Vienlaicīgais process: Katrs balons darbojas kā atsevišķs pavediens, kustoties vienlaicīgi ar citiem baloniem, kurus varētu būt ekrānā vienlaicīgi.

Vienlaicīgais procesu modelis šajā spēlē ļaus saskaņot dažādas spēles darbības, piemēram, laika atjaunināšanu, balonu kustību un lietotāja ievadi, tādējādi nodrošinot dinamisku un reaktīvu spēles pieredzi.

### Datu dekompozīcija

Datu dekompozīcija ir process, kurā datus sadala un organizē, lai tos būtu viegli pārvaldīt un izmantot programmā. Dotajā spēlē dati jāiestrādā vairākos mainīgajos, kas atbildēs par dažādām spēles aspektiem. Paredzamie datu dekompozīcijas elementi šajā kodā:

1. Galvenie dati:

* *speed, intervals, rand:* Mainīgie, kas nosaka balonu kustības ātrumu, parādīšanās intervālu un gadījuma skaitļu ģeneratoru.
* *balloonSkins, i, missedBalloons, score:* Mainīgie, kas uztur informāciju par balonu izskatu, kustību, nokavētajiem baloniem un rezultātu.

1. Grafiskie dati:

* *background: ImageBrush* objekts, kas nosaka fona attēlu.

1. Grafikas elementu dzēšana:

* *itemRemover: List<Rectangle>* objekts, kas tiek izmantots, lai uzglabātu *Rectangle (balona)* objektus, kuri tiek dzēsti no ekrāna.

1. Skaņas dati:

* *player: MediaPlayer* objekts, kas tiek izmantots skaņas efekta atskaņošanai.

1. Spēles laika dati:

* *gameTimer: DispatcherTimer* objekts, kas nosaka spēles laika periodu.

1. Grafiskās sastāvdaļas:

* *MyCanvas: Canvas* objekts, uz kura attēlojas spēles komponentes.

Datu dekompozīcijai spēlē jānodrošina skaidru un organizētu veidu, kā pārvaldīt dažādus aspektus un funkcijas. Katrs datiem paredzētais mainīgais vai objekts ir atbildīgs par noteiktu spēles aspektu, kas atvieglo izpratni un uzturēšanu. Tas arī palīdz pievienot vai modificēt spēles funkcionalitāti, nemainot citus datu aspektus, kas ir svarīgi modulārai un viegli uztveramai programmai.

## Atkarību apraksts

### Moduļu struktūra un atkarības

Lai labāk izprastu sadarbības stratēģiju starp projektējuma entītijām, apskatīsim moduļu saistības un moduļu saķēdēšanas tipus.

Preču katalogam ir komunikatīvā saistība, jo tas sevī ietver vairākus procesus ar kopēju datu struktūru (preces dati), moduļa strukturālā diagramma redzama 2.1.1 attēlā. Un 2.1.2 attēlā.

*gameEngine* modulis

(Spēles loģika)

Lietotāja saskarne (UI) modulis

Skaņas modulis

**2.1.1. att. Spēles moduļa vienkāršotā strukturētā diagramma**

Lietotāja saskarne (UI)

*UI Handling*

*Visualisation*

*gameEngine* modulis

(Spēles loģika)

Skaņas modulis

Spēles *state* un loģika

*GameTimer*

*Balloon* loģika

Skaņas loģika

Skaņas atskaņošana

**2.1.2. att. Spēles moduļa paplašinātā strukturētā diagramma**

### Starpprocesu atkarības

Spēles sistēma sastāv no vairākiem procesiem, to atkarības un atbildības aprakstītas zemāk:

1. Lietotāja saskarnes modulis (UI):

* Atbildība: Nodrošina lietotāja saskarni un vizuālo reprezentāciju.
* Atkarības: Sākas un pārvalda galveno logrīku, kas saistīts ar lietotāja interakciju un vizuālo attēlojumu.

1. Spēles loģikas modulis (*gameEngine*):

* Atbildība: Nodrošina galveno spēles loģiku, tai skaitā balonu parādīšanu, kustību un rezultātu apstrādi.
* Atkarības: Sazinās ar lietotāja saskarnes moduli, lai attēlotu rezultātus un vadītu spēles laiku.

1. Skaņas modulis (*MediaPlayer*):

* Atbildība: Pārvalda skaņas efektu atskaņošanu, īpaši, kad balons tiek noklikšķināts.
* Atkarības: Sazinās ar spēles loģikas moduli, lai reaģētu uz notikumiem un atskaņotu skaņas efektu.

### Datu atkarības

1. Lietotāja saskarnes un spēles loģikas datu atkarība:

* Lietotāja saskarnes modulis UI (MainWindow) ir saistīts ar spēles loģikas moduli (gameEngine). Lietotāja darbības, piemēram, pogas noklikšķināšana, ietekmē spēles loģiku un var izraisīt spēles stāvokļa izmaiņas.

1. Skaņas un spēles loģikas datu atkarība:

* Skaņas modulis (*MediaPlayer*) ir saistīts ar spēles loģikas moduli (*gameEngine*). Skaņas efekti tiek atskaņoti, kad spēles loģika to norāda, piemēram, balons tiek noklikšķināts un sprāgst.

1. Grafiskās lietotāja saskarnes un spēles loģikas datu atkarība:

* Grafiskā lietotāja saskarnes modulis (MainWindow) saņem informāciju par spēles stāvokli no spēles loģikas moduļa (gameEngine). Tas attiecas uz rezultātu attēlošanu un spēles stāvokļa atjaunināšanu ekrānā.

## Moduļu saskarnes apraksts

### Moduļu saskarne

Dotajā spēlē ir trīs galvenie moduļi: Lietotāja saskarnes modulis UI (MainWindow), Spēles loģikas modulis (gameEngine), un Skaņas modulis (MediaPlayer). Zemāk šo moduļu saskarnes apraksts:

1. Lietotāja saskarnes modulis UI (MainWindow):

* Atbildība:
  + Nodrošina grafisko lietotāja saskarni spēlei.
  + Reaģē uz lietotāja darbībām, piemēram, pogas noklikšķināšana.
* *Interface:*
  + *canvasKeyIsDown(object sender, KeyEventArgs e):* Metode, kas apstrādā taustiņu nospiešanu un nodrošina spēles restartu.
  + *PoPBalloons(object sender, MouseButtonEventArgs e):* Metode, kas apstrādā peles klikšķi un reaģē uz balonu sprādzieniem.

1. Spēles loģikas modulis (gameEngine):

* Atbildība:
  + Pārvalda galveno spēles loģiku un stāvokli.
  + Nosaka balonu ģenerēšanas un kustības loģiku.
  + Izlemj par spēles notikumu izraisīšanu un rezultātu atjaunināšanu.
* Interface:
  + *startGame():* Metode, kas sāk jaunu spēli un inicializē spēles sākuma stāvokli.
  + *ResetGame():* Metode, kas izsaucas, lai atjaunotu spēli pēc tās beigām vai restartētu.
  + *gameEngine(object sender, EventArgs e):* Metode, kas tiek izsaukta regulāri, lai atjauninātu spēles stāvokli.

1. Skaņas modulis (*MediaPlayer*):

* Atbildība:
  + Pārvalda skaņas efektu atskaņošanu.
  + Reaģē uz spēles notikumiem, piemēram, balona sprādzieniem.
* *Interface:*
  + Nav īpašas saskarnes metodes, bet tiek aktivizēts ar citiem moduļiem, kad ir nepieciešams atskaņot skaņas efektus.

## Detalizētais projektējums

### Moduļu detalizētais projektējums

Moduļu projektēšanā izmantošu gan lēmumu tabulas gan arī pseidokodu.

### Lēmumu tabula

Nākamajās tabulā tiks attēlota spēles lēmumi

3.tabula

Lēmumu tabula

|  |  |  |  |
| --- | --- | --- | --- |
| *Space* taustiņš nospiests | Peles kreisais taustiņš nospiests | Spēle aktīva *(Game active)* | Darbība |
| nē | - | nē | Nedari neko |
| jā | - | nē | Atjaunot spēli |
| nē | jā | jā | Uzspridzināt balonu |
| jā | jā | jā | Uzspridzināt balonu un spēlēt skaņu |

### Pseidokods

Tālāk aprakstīšu moduļus ar pseidokoda palīdzību.

Inicializē spēles mainīgos un objektus

Iestatījumi:

ātrums = 2

intervāls = 90

skaits = 0

i = 0

neizšautiebaloni = 0

spēlesAktīva = false

Uzstādi fona attēlu un sāc spēli:

Izsauc funkciju: IzveidoJaunuSpēli()

Funkcija: IzveidoJaunuSpēli()

Notīra ekrānu no esošajiem objektiem

Uzstāda sākuma vērtības mainīgajiem

Sāk spēles taimeri

Uzstāda spēles aktīvumu uz true

Funkcija: AtjauninaSpēli(katra 20 milisekundēs)

Attēlo rezultātu ekrānā

Atjauno balonu pozīcijas un ātrumu

Pārbauda, vai ir laiks ģenerēt jaunu balonu

Ja jā, izveido jaunu balonu un uzstāda ģenerēšanas intervālu

Pārvieto esošos balonus uz augšu un sānos

Pārbauda, vai balons ir ārpus ekrāna

Ja jā, palielina neuzsprāgušo balonu skaitu

Funkcija: UzliktSkaņu(balonu sprādzienā)

Atskaņo "pop\_sound.mp3"

Lietotāja darbību apstrāde:

Ja taustiņš "Space" nospiests un spēle nav aktīva

Izsauc funkciju: IzveidoJaunuSpēli()

Ja peli noklikšķina un spēle ir aktīva

Pārbauda, vai klikšķis trāpījis uz balona

Ja jā, izsauc funkciju: UzliktSkaņu(balonu sprādzienā)

un noņem balonu no ekrāna, palielina rezultātu

Pamatprogrammas cilpa:

Katrā ciklā izsauc funkciju: AtjauninaSpēli()

Pārtrauc cilpu, ja ir pārsniegts maksimālais neuzsprāgušo balonu skaits

vai sasniegts noteikts rezultāts

### Datu detalizētais projektējums

Datu detalizētajā projektējumā tiek parādīts mainīgo apraksts.

4. tabula

Autorizācijas moduļa mainīgo apraksts

|  |  |  |
| --- | --- | --- |
| Nosaukums | Datu tips | Nolūks |
| Score | int | Spēlētāja rezultāts. |
| IncreaseScore | Metode | Palielina spēlētāja rezultātu par norādītu punktu skaitu. |
| ResetScore | Metode | Atiestata spēlētāja rezultātu uz nulli. |
| BalloonID | int | Unikāls identifikators balonam |
| BalloonType | int | Veida identifikators balonam (1-5). |
| PositionX | double | X koordināta ekrānā. |
| PositionY | double | Y koordināta ekrānā. |
| IsPopped | bool | Norāda, vai balons ir saplīsis. |
| Pop | Metode | Atzīmē balonu kā saplīsis. |
| Move | Metode | Pārvieto balonu uz jaunām koordinātēm ekrānā. |

# Lietotāja ceļvedis

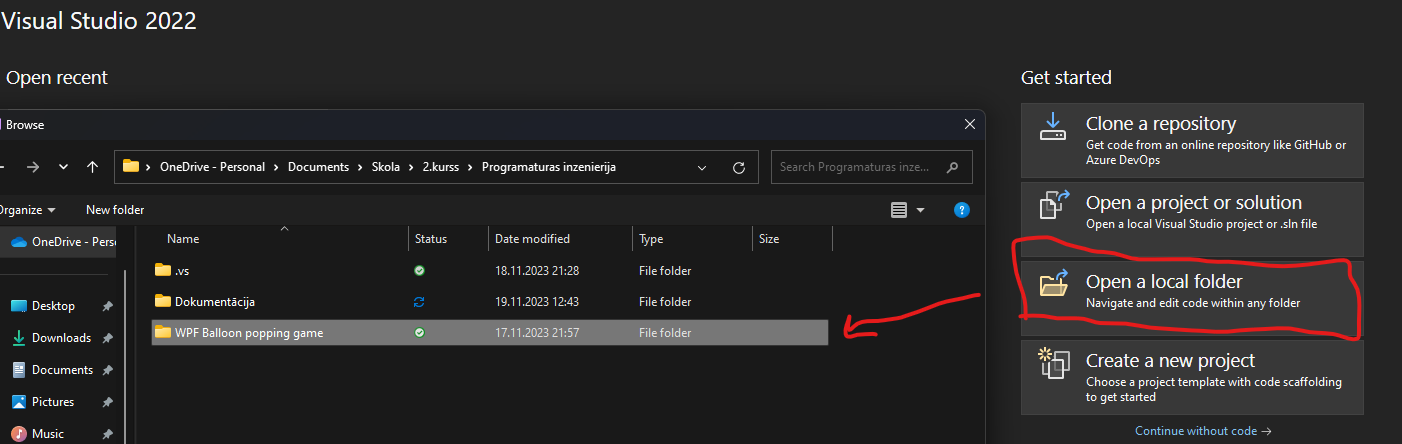
## Spēles apraksts

Lai uzsāktu spēli spēles lietotajam nepieciešams uzstādīt programmatūru *Visual studio.*

Spēles galvenais mērķis ir uzspridzināt pēc iespējas vairāk balonus, lai iegūtu maksimālu punktu skaitu. Spēlētājiem būs jāparāda savas reakcijas spējas, noklikšķinot uz parādībās esošajiem baloniem, pirms tie pārvietojas pār ekrānu un pazūd

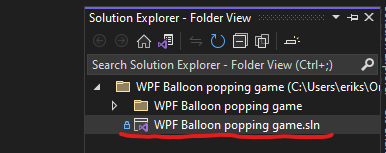
### Spēles palaišana

1. Lai palaistu spēli lietotājam programma *Visual studio* jānorāda spēles atrašanās vieta (direktorija) (3.1. att)



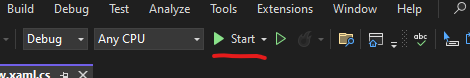
3.1. att. spēles atrašanās vietas norādīšana

1. Pēc kā atverot Solution explorer jāizvēlas fails “WPS Balloon popping game.sln” (3.2. att).



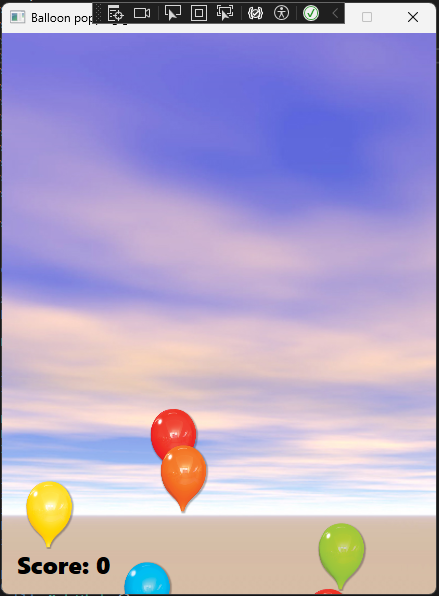
3.2. att. spēles palaišanas fails

1. Pēc palaišanas iespējošajās rīkjoslas poga “Start” (3.3. att).



3.3. att. spēles palaišanas poga “Start”

1. Pēc pogas *Start* nospiešanas programmatūra izvada spēles lauku un spēli var uzsākt (3.4. att).



3.4. att. uzsākta spēle

### Spēles spēlēšana

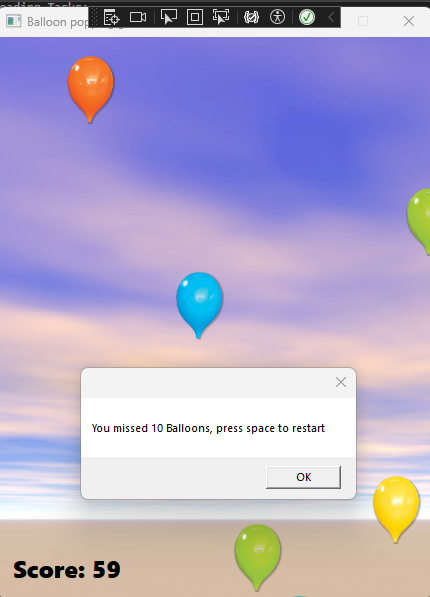
Spēles uzdevums lietotājam ir ar peles palīdzību virzīt peles kursoru pa spēles lauku un uzvirzot to uz ballona nospiest kreiso peles taustiņu tādejādi uzspridzinot balonu. Par katru uzspridzināto balonu tiek saņemts 1 punkts un atskaņots sprādziena skaņu efekts. Spēles uzdevums uzspridzināt pēc iespējas vairāk balonus kamēr tie nav pametuši spēles lauku, lai iegūtu maksimālu punktu skaitu pilnveidojot savas reakcijas spējas (3.5. att). ik pēc 20 veiksmīgi saspridzinātiem baloniem balonu kustības ātrums palielinās.



3.5. att. spēles gaita ar punktu uzskaiti

### Spēles beigas

Spēles beidzas ja lietotājs izlaiž no spēles loga 10 balonus. Pēc kā ir iespēja uzsākt spēli atkārtoti nospiežot uz ievades klaviatūras taustiņu “Space” (1.6. att.) vai ja lietotājs izvēlas neturpināt spēli aizvērt programmu *Visual studio*



3.6. att. spēles beigu paziņojums

# Testēšana

Es testēju Antona “Mobila lietojumprogramma sporta nodarbības procesa kontrolei”, testēšanai esmu izvēlējis melnās kastes metodi.

## Testēšanas plāns

Lai sekmīgi veiktu testēšanu tiek izstrādāts sekojošs testēšanas plāns:

1. Lietotāja reģistrācija

* Pārbaudīt, vai lietotājs var veiksmīgi reģistrēties ar Google kontu.
* Pārbaudīt, vai lietotājam tiek atgādināts par nepieciešamajiem datiem, ja tie nav ievadīti.
* Pārbaudīt, vai lietotne ģenerē unikālu lietotāja identifikatoru.

1. Lietotāja pieslēgšanās

* Pārbaudīt, vai lietotājs var veiksmīgi pieslēgties ar Google kontu.
* Pārbaudīt, vai lietotne informē lietotāju par interneta savienojuma trūkumu, ja tas notiek.

1. Lietotāja profila pārvaldība

* Pārbaudīt, vai lietotājs var rediģēt savu profilu, mainot vārdu, svaru un augumu.
* Pārbaudīt, vai lietotājs var apskatīt savu profilu un dzēst savu profilu, ja nepieciešams.
* Pārbaudīt, vai informācija par rezultātiem tiek pareizi atspoguļota lietotāja profilā.

1. Lietotāja treniņu pārvaldība

* Pārbaudīt, vai lietotājs var izveidot treniņus ar vingrinājumu sarakstu.
* Pārbaudīt, vai treniņu izpilde notiek bez problēmām un tiek uzglabāta informācija par to, piemēram, izpildītajiem vingrinājumiem un laiku.
* Pārbaudīt, vai lietotājs var rediģēt izveidotos treniņus un dzēst nepieciešamības gadījumā.

1. Standarta treniņu pārvaldība

* 5.1. Pārbaudīt, vai lietotājs var apskatīt informāciju par standarta treniņiem.
* 5.2. Pārbaudīt, vai standarta treniņu izpilde notiek bez problēmām un tiek pareizi atspoguļota informācija par to.

1. Rezultātu pārvaldība

* Pārbaudīt, vai lietotājs var apskatīt rezultātus par izpildītajiem treniņiem.
* Pārbaudīt, vai informācija par izpildītajiem un uzsāktajiem treniņiem tiek pareizi atspoguļota.

1. Lietotāja interfeisa pārbaude

* Pārbaudīt, vai lietotne atbilst specifikācijām attiecībā uz krāsām, virsrakstiem, tekstu un vispārējo lietotāja saskares izskatu.
* Pārbaudīt, vai lietotne ir viegli pārskatāma un lietojama uz jebkuras mobilas ierīces.

## Ievades dati

Zemāk izveidotā 5.tabula nodrošina strukturētu skatu par dažādām testēšanas situācijām un atbilstošajiem ievades datiem. Katrā rindiņā ir norādīts konkrēts testa scenārijs un tā apraksts, kā arī, kur nepieciešams, ievades dati, kas jāievada, lai pārbaudītu attiecīgo funkcionalitāti. Tabula palīdzēs organizēt un plānot testēšanas procesu atbilstoši izveidotajam testēšanas plānam.

5. tabula

Ievades dati testēšanas plānas

|  |  |
| --- | --- |
| **Testa scenārijs** | **Ievades dati** |
| 1. **Lietotāja reģistrācija** |  |
| Lietotājs reģistrējas ar Google kontu | Google konta informācija (simbolu virkne) |
| Lietotājs neievada nepieciešamos datus reģistrācijas laikā | Nepieciešamie dati: vārds, svars, augums |
| 1. **Lietotāja pieslēgšanās** |  |
| Lietotājs pieslēdzas ar Google kontu | Google konta informācija (simbolu virkne) |
| Lietotājam nav interneta savienojuma | Paziņojums par trūkstošu interneta savienojumu |
| 1. **Lietotāja profila pārvaldība** |  |
| Lietotājs maina sava profila informāciju | Jauni dati: vārds, svars, augums |
| Lietotājs apskata savu profilu | Funkcionalitāte |
| 1. **Lietotāja treniņu pārvaldība** |  |
| Lietotājs izveido treniņu ar vingrinājumu sarakstu | Nosaukums, vingrinājumu saraksts |
| Lietotājs rediģē izveidoto treniņu | Mainīti dati: nosaukums, vingrinājumu saraksts |
| Lietotājs izdzēš izveidoto treniņu | Apstiprinājuma poga "Dzēst Treniņu" |
| 1. **Standarta treniņu pārvaldība** |  |
| Lietotājs apskata informāciju par standarta treniņiem | Funkcionalitāte |
| Lietotājs veiksmīgi izpilda standarta treniņu | Funkcionalitāte |
| 1. **Rezultātu pārvaldība** |  |
| Lietotājs apskata rezultātus par izpildītajiem treniņiem | Funkcionalitāte |
| Lietotājs apskata rezultātus par uzsāktajiem treniņiem | Funkcionalitāte |
| 1. **Lietotāja interfeisa pārbaude** |  |
| Vispārējā lietotāja pieredze | Reģistrācijas un pieslēgšanās procesa pārbaude |
| UI elementu pārbaude (pogas, formas, teksta lauki) | Pārbaude, vai UI elementi darbojas pareizi |
| Grafiskās saskares saskaņotība | Krāsu shēmas, fonti, ikonas utt. pārbaude |
| Responsīvās dizaina pārbaude | Pārbaude uz dažādām ekrāna izmēriem |

## Testēšanas scenāriju izpilde

Zemāk izveidotā 6.tabula atspoguļo vai programmas funkcijas darbojas pareizi un atbilst specifikācijām balstoties uz ievades datiem un paredzamo rezultātu.

6. tabula

Testēšanas scenāriju tabula

|  |  |  |  |
| --- | --- | --- | --- |
| **Testa scenārijs** | **Ievades dati** | **Paredzamais rezultāts** | **Iegūtais rezultāts** |
| 1. **Lietotāja reģistrācija** | E-pasts: [test@example.com](mailto:test@example.com), Vārds: Antons, Augums: 180 cm, Svars: 75 kg | Lietotājs ir veiksmīgi reģistrējies. Profilā redzams e-pasts, vārds, augums un svars. | Atbilst paredzamajam |
| 1. **Lietotāja pieslēgšanās** | Lietotājvārds: user123, Parole: Test123 | Lietotājs ir veiksmīgi pieslēdzies. Sistēma parāda lietotāja informāciju un ļauj piekļūt funkcijām. | Atbilst paredzamajam |
| 1. **Lietotāja profila rediģēšana** | Jaunais vārds:Juris  Augums: 179 cm, Svars: 78 kg | Profila svars ir veiksmīgi atjaunināts. Jaunā informācija atspoguļojas lietotāja profilā. | Atbilst paredzamajam |
| 1. **Treniņu programmas izveide** | Nosaukums: "Sākuma programma", Vingrinājumi: [Push-up, Sit-up, Pull-up]  Lietotājs nomaina Sit-up pret Plank  Dzēš treniņa programmu | Treniņu programma ir izveidota. Programmas nosaukums un iekļautie vingrinājumi ir redzami.  Vingrinājumi: [Push-up, Plank, Pull-up]  Treniņa programma ir izdzēsta | Atbilst paredzamajam |
| 1. **Treniņa izpilde** | Izvēlētā programma: "Sākuma programma" | Lietotājs var veiksmīgi izpildīt treniņu, sekojot programmai. Izpildes dati tiek saglabāti. | Atbilst paredzamajam |
| 1. **Rezultātu pārvaldība** | Pabeigtais treniņš: "Sākuma programma", Rezultāti: [Push-up - 3x15, Sit-up - 4x20, Pull-up - 3x12] | Saglabātie rezultāti atspoguļo veikto treniņu un to rezultātus. | Atbilst paredzamajam |
| 1. **Lietotāja interfeisa pārbaude** | Vispārējā lietotāja pieredze, UI elementu pārbaude, Grafiskās saskares saskaņotība, Responsīvās dizaina pārbaude | Lietotājiem ir ērti lietot un pārvietoties pa lietotni, UI elementi darbojas korekti un ir estētiski saskaņoti. | Atbilst paredzamajam |

## Kļūdu pārvaldība

Programmas testēšanas izpildes laikā kļūdas netika identificētas.

## Dokumentācija

Testēšanai tika iesniegs darbs kurs izstrādāts iesniegšanai Rīgas valsts tehnikumā kā kvalifikācijas darbs tajā ir iekļautas uzdevuma nostādnes, prasību specifikācija, uzdevuma risināšanas līdzekļu izvēles pamatojums, programmatūras produkta modelēšanas un projektēšanas apraksts, datu struktūru apraksts, lietotāja ceļvedi ar atbilstīgiem aprakstiem un piemēriem. Strukturēts dokumentācijas pārskats attēlots zemāk.

1. UZDEVUMU NOSTĀDNE

2. PRASĪBU SPECIFIKĀCIJA

2.1. Ieejas un izeja informācijas apraksts

2.1.1. Ieejas informācijas apraksts

2.1.2. Izejas informācijas apraksts

2.2. Funkcionālas prasības

2.3. Nefunkcionālas prasības

2.4. Skices

3. UZDEUMU RISINĀŠANAS IZVĒLES PAMATOJUMS

4. PROGRAMMATŪRAS PRODUKTA MODELĒŠANA UN PROJEKTĒŠANA

4.1. Sistēmas struktūras modelis

4.1.1. Sistēmas arhitektūra

4.1.2. Sistēmas ER modelis

4.2. Funkcionālas sistēmas modelis

4.2.1. Datu plūsmu modelis

5. DATU STRUKTŪRU APRAKSTS

6. LIETOTĀJA CEĻVEDIS

6.1. Sistēmas prasības aparatūrai un programmatūrai

6.2. Sistēmas instalācija un palaišana

6.3. Programmas apraksts

6.4. Testa piemērs

# Secinājumi

**Secinājumi par izstrādāto darba izstrādi:**

1. **Prasību izstrāde:** Prasību izstrāde ir kritiska programmas veiksmīgai izstrādei. Jo precīzāk un izsmeļojošāk ir izstrādātas prasības, jo mazāk ir neskaidrību un neprecizitāšu, kas varētu rasties izstrādes procesa laikā.
2. **Struktūru un arhitektūru:** Pareiza un elastīga arhitektūra ir būtiska, lai nodrošinātu, ka programma var pielāgoties mainīgiem apstākļiem un nākotnes papildinājumiem, kombinējot pieejamos resursus. Kas ņemot vērā kursa ietvaros pielietoto principu “no sākuma kods tad apraksts” veiksmīgi ļāva pielāgot gan aprakstu kodam gan otrādi.
3. **Lietotāja ceļvedis:** Lietotāja ceļvedis ir galvenais instruments, kas palīdz izstrādāt lietotājam draudzīgu un intuitīvu lietotāja saskari. Labi izstrādāts lietotāja ceļvedis atvieglo lietotāja pieredzi, veicinot efektīvu mijiedarbību ar lietotni.

**Secinājumi pat testēšanu:**

1. **Kompleksa funkcionalitāte:** Testētā programma paredz plašu funkcionalitāti, sākot no lietotāja reģistrācijas līdz treniņu programmu izveidei un rezultātu saglabāšanai. Lai nodrošinātu, ka ikviens šīs funkcionalitātes aspekts darbojas korekti, ir nepieciešama rūpīga un visaptveroša testēšana.
2. **Savietojamība un veiktspēja:** Testētās programmas aprakstā ir noteikts, ka lietotne jāizstrādā Kotlin valodā un jābūt saderīgai ar dažādām mobilajām ierīcēm. Testēšanai jāiekļauj savietojamības tests, lai pārliecinātos, ka lietotne darbojas uz dažādām platformām un ekrānu izmēriem, kā arī veiktspējas testi, lai nodrošinātu ātru ielādes laiku.

# Literatūra

1. Alberta Koledža, Programmatūras inženierija, lekciju materiāli.