Rīgas 64. Vidusskola

IZGLĪTOJOŠA AUGU KOPŠANAS APLIKĀCIJA AR LAISTĪŠANAS ATGĀDINĀJUMIEM

Programmatūras prasību specifikācija

Autors:

Kārlis Atvase 12. DIT

Darba vadītājs:

Edvards Bukovskis

RĪGA 2023

Anotācija

Pielaides darbā tika izstrādāta pamata funkciju programmatūra izglītojošai augu kopšanas aplikācijai, programmatūras spesifikācija un to projektējums. Šis darbs ir pamats funkcionējošas aplikācijas izveidei un var tikts izmantots tālākai projekta attīstībai.

Lietojumprogramma "Augo" ir vienkāršs un lietotājam draudzīgs rīks, kas palīdz augu entuziastiem sekot līdzi augu laistīšanas grafikiem. Lietojumprogramma ļauj lietotājiem izvēlēties konkrētus augus, kas ir viņu kolekcijā, un iestatīt pielāgotus laistīšanas atgādinājumus, pamatojoties uz viņu laistīšanas vajadzībām.

Lai sāktu, lietotāji var ievadīt sava auga veidu, un lietotne sniegs informāciju par to, cik bieži tas ir jālaista un cik daudz ūdens nepieciešams. Šo informāciju var pielāgot, ņemot vērā auga īpašo vidi un augšanas apstākļus. Kad lietotājs ir iestatījis augu profilus, aplikācija ieplāno atgādinājumus par laistīšanu. Lietotne noteiktos intervālos nosūtīs push paziņojumus, lai atgādinātu lietotājiem laistīt savus augus. Kad ir pienācis laiks laistīt augu, lietotne sniegs konkrētus norādījumus par to, kā laistīt augu, piemēram, cik daudz ūdens jālieto un kur laistīt augu (t.i., augsni vai lapas). Lietotāji var arī izsekot, kad viņi laistīja savus augus, un pievienot piezīmes par auga augšanu un tā veselību.

Abstract

In the permitting work, basic function software for an educational plant care application, software specification and their design were developed. This work is the basis for creating a functioning application and can be used for further development of the project.

The “Augo” app is a simple and user-friendly tool that helps plant enthusiasts keep track of plant watering schedules. The app allows users to select specific plants in their collection and set custom watering reminders based on their watering needs.

To get started, users can enter the type of their plant and the app will provide information on how often it needs to be watered and how much water it needs. This information can be adapted to the specific environment and growing conditions of the plant. Once the user has set plant profiles, the application schedules watering reminders. The app will send push notifications at set intervals to remind users to water their plants. When it's time to water the plant, the app will give you specific instructions on how to water the plant, such as how much water to use and where to water the plant (i.e. soil or leaves). Users can also track when they watered their plants and add notes about the plant's growth and health.

Saturs

1. Ievads 5
1.1. Nolūks 5
1.2. Darbības sfēra 5
1.3. Saistība ar citiem dokumentiem 5
1.4. Pārskats 5
2. Vispārējais apraksts 6
2.1. Produkta perspektīva 6
2.2. Produkta funkcijas 6
2.3. Lietotāji 6
2.4. Vispārējie ierobežojumi 7
2.5. Sistēmas funkcionalitāte 7
2.6 Datu plūsmas shēma 7
3. Konkrētās prasības 7
3.1. Funkcionālās prasības 8
3.1.1. Ievads 8
3.1.2. Ievade 9
3.1.3. Apstrāde 9
3.1.4. Izvade 9
3.2. Ārējās saskarnes prasības 9
3.2.1. Lietotāja saskarne 10
3.3. Atribūti 10
3.3.1. Drošība 10
3.3.2. Uzturamība 10
3.4. Citas prasības 10
3.4.1. Datu bāze API 10
4. Koda apraksts 11
1. Datubāzes piekļuve 11
2. Klases definēšana 11
3. Profilu izveide 11
4. Laika funkcija 11
5. Paziņojumu sistēma 11
5. Testēšana 12
6. Secinājumi un autora komentāri 12
Izmantotā literatūra un informācijas avoti: 13
Pielikumi 13

Vārdnīca

API	Funkciju un procedūru kopums, kas ļauj izveidot lietojumprogrammas, kuras piekļūst operētājsistēmas, lietojumprogrammas vai cita pakalpojuma funkcijām vai datiem.
UI	Lietotāja un datorsistēmas mijiedarbības līdzekļi, jo īpaši ievadierīču un programmatūras izmantošana.
PPS	Programmatūras prasību specifikācija
IAKAALA	**Izglītojoša augu kopšanas aplikācija ar laistīšanas atgādinājumiem**
GUI	Vizuāls veids, kā mijiedarboties ar datoru, izmantojot tādus vienumus kā Windows, ikonas un izvēlnes, ko izmanto vismodernākās operētājsistēmas.
RDBMS	Programmatūru, ko izmanto, lai glabātu, pārvaldītu, vaicātu un izgūtu relāciju datu bāzē saglabātos datus, sauc par relāciju datu bāzes pārvaldības sistēmu
MySQL	Relāciju datu bāzes pārvaldības sistēma
FCM & APNS	Firebase mākoņa ziņojumapmaiņa, Apple paziņojumu pakalpojums.

1. Ievads

1.1. Nolūks

Izveidojot lietotni, kas atgādina Jums laistīt augus, cik daudz, kuri augi un kā var sniegt vairākas priekšrocības, padarot augu kopšanu vieglāku, ātrāku un efektīvāku. Tādēļ lietojumprogramma var uzlabot augu veselību, samazināt pārmērīgas vai nepietiekamas laistīšanas risku, palielināt produktivitāti, nodrošināt pielāgotu aprūpi un sniegt labumu videi.

1.2. Darbības sfēra

Lietotājprogramma Augo var noderēt ikvienam, kam patīk dārza darbi vai augu turēšana mājās. Turklāt šī lietotne var būt noderīga Lietotāji ar lielu augu kolekciju, jo var būt grūti izsekot katra auga unikālajām laistīšanas vajadzībām. Ikviens, kurš vēlas pārliecināties, ka viņu augi ir veselīgi var gūt labumu no šīs lietojumprogrammas, kas atgādina par augu laistīšanas tendencēm.

1.3. Saistība ar citiem dokumentiem

- Dokumenta PPS noformēšanā ievērotas standarta LVS 68:1996 prasības
- IAKAALA PPS ir cieši saistīs ar šīs pašas sistēmas PPA.

1.4. Pārskats

Darbs sastāv no četrām daļām, Pirmajā daļā jeb ievadā tiek pārskatīta lietotājprogrammas nolūks, darbības sfēra un saistība ar citiem dokumentiem.

Darba otrajā daļā tiek paskaidrots produkta nākotne, tās funkcijas, kas to lietots, kādi var būt ierobežojumi lietotnē un datu plūsma.

Trešajā daļā tiek definētas funkcionālās prasības, precīzi aprakstīta lietotājprogrammas sistēma, tās ievade, apstrāde un izvade. Trešās daļas beigās tiek aprakstīti atribūti par lietotnes drošību un uzturamību

Darba pēdējā daļā tiek detalizēti aprakstīts koda daļas un tā nolūks. Pielikumā var apskatīt to.

2. Vispārējais apraksts

2.1. Produkta perspektīva

Pašlaik programma ir neatkarīga, jo tikai balsta visas funkcijas uz datubāzēm, savukārt nākotnē lietojumprogramma, varētu integrēt sensorus, lai vēl vairāk optimizētu augu kopšanu. Sensorus var novietot augsnē vai uz paša auga, lai uzraudzītu mitruma līmeni, temperatūru un citus vides faktorus, kas ietekmē augu augšanu. Šos datus varētu pārsūtīt uz lietotni, ļaujot tai pielāgot laistīšanas grafikus un sniegt precīzākas kopšanas instrukcijas, pamatojoties uz pašreizējā laika informāciju. Turklāt sensorus var izmantot, lai noteiktu kaitēkļu vai slimību klātbūtni, brīdinot lietotāju rīkoties, lai novērstu turpmākus bojājumus.

2.2. Produkta funkcijas

- Augu profili - lietotāji var izveidot profilus katram augam, tostarp informāciju, piemēram, augu veidu, laistīšanas prasības un vēlamo atrašanās vietu.
- Pielāgoti laistīšanas grafiki - lietotāji var iestatīt pielāgotus laistīšanas grafikus, pamatojoties uz katra auga vajadzībām, tostarp nepieciešamo ūdens biežumu un daudzumu.
- Laistīšanas atgādinājumi - lietotne var nosūtīt atgādinājumus, kad ir pienācis laiks laistīt katru augu, pamatojoties uz pielāgoto laistīšanas grafiku.
- Laistīšanas norādījumi - lietotne var sniegt konkrētus norādījumus par katra auga laistīšanu, tostarp nepieciešamo ūdens daudzumu un vietu, kur augu laist (t.i., augsni vai lapas).
- Padomi augu kopšanai - lietotne var sniegt papildu informāciju un padomus par katra auga kopšanu, tostarp atzarošanu, mēslošanu un pārstādīšanu.
- Augu žurnāls - lietotāji var izsekot, kad viņi laistīja katru augu, pievienot piezīmes par auga augšanu un veselību un noteikt augu kopšanas mērķus.
- Augu bibliotēka - lietotne var nodrošināt augu bibliotēku, tostarp informāciju par katra auga laistīšanas prasībām, vēlamo vidi un kopšanas instrukcijas.

2.3. Lietotāji

- Lietotāji, kuriem patīk dārzkopība vai kuri aizraujas ar augiem, šī lietotne var būt noderīga, lai sekotu līdzi savu augu laistīšanas vajadzībām.
- Lietotāji ar aizņemtiem grafikiem, kuriem, iespējams, nav laika sekot līdzi augu kopšanai, var uzskatīt, ka šī lietotne ir noderīga, lai pārvaldītu viņu augu kopšanu.
- Lietotāji, kuri ir iesācēji augu aprūpē, šī lietotne var būt noderīga, lai uzzinātu, kā pareizi rūpēties par saviem augiem.
- Cilvēki, kuri mēdz aizmirst, kad laistīt savus augus, vai kuriem var būt grūtības sekot līdzi laistīšanas grafikiem, var uzskatīt, ka šī lietotne ir noderīga, lai sekotu līdzi augu kopšanai.
- Lietotāji, kuriem ir liela augu kolekcija, šī lietotne var būt noderīga, lai sekotu līdzi katra auga unikālajām laistīšanas prasībām.

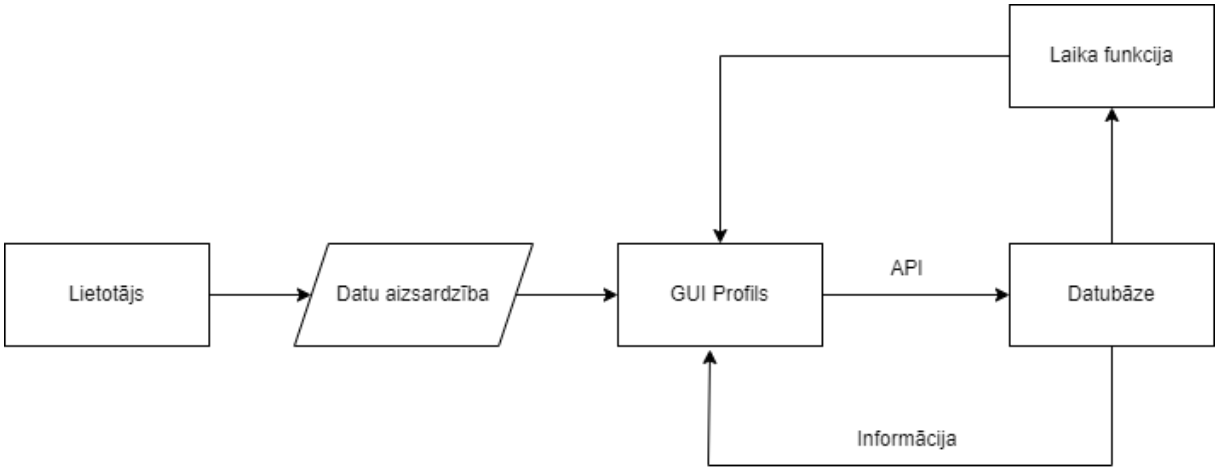
2.4. Vispārējie ierobežojumi

- Neprecīza informācija par augu - lietotnes datu bāzē var nebūt pilnīgas vai precīzas informācijas par katru augu veidu, kā rezultātā var tikt sniegti nepareizi laistīšanas norādījumi.
- Lietotāja kļūda - lietotāji var aizmirst ievadīt pareizo informāciju par saviem augiem vai nevar pareizi iestatīt atgādinājumus, tādējādi radot nepareizus laistīšanas grafikus.
- Atkarība no tehnoloģijām - lietotne var būt atkarīga no stabila interneta savienojuma vai piekļuves viedtālrunim, kas var ierobežot tās funkcionalitāti apgabalos ar sliktu savienojumu vai zemu viedtālruņa lietojumu.
- Pārmērīga paļaušanās uz atgādinājumiem - pārāk liela paļaušanās uz atgādinājumiem var likt lietotājiem atstāt novārtā citus augu kopšanas aspektus, piemēram, mēslošanu vai atzarošanu.
- Ierobežota augu šķirne - lietotnes datu bāzē var nebūt iekļauti visi augu veidi, tādējādi ierobežojot tās noderīgumu lietotājiem ar daudzveidīgu augu kolekciju.

2.5. Sistēmas funkcionalitāte

Šādas lietojumprogrammas ietvertu lietotnes saskarsnes izstrādi, kā arī integrāciju ar datu bāzi, lai saglabātu un izņemtu informāciju par augiem. Lietojumprogramma prasītu arī lietotājam atļaut atbilstošos laikos nosūtīt paziņojumus un atgādinājumus, kas prasītu integrāciju ar paziņojumu sistēmu.

2.6 Datu plūsmas shēma



Attēls Nr. 2 "Datu plūsmas shēma"

3. Konkrētās prasības

3.1. Funkcionālās prasības

Lietotāja saskarne - Lai izveidotu lietotāja saskarni šai augu kopšanas lietojumprogrammai, kodā būtu nepieciešami daži galvenie komponenti. Pirmkārt, kodā būtu jāiekļauj funkciju vai metožu kopums, lai parādītu dažādus lietotāja saskarnes ekrānus un formas, tostarp iepreikš minētos profilus un vizualizācijas rīkus. Šīs funkcijas vai metodes, visticamāk, izmantos grafiskā lietotāja saskarnes (GUI) bibliotēku.

API - API šajā augu kopšanas lietojumprogrammā darbosies kā starpniekslānis, kas savieno datu bāzi ar lietotāju. Kad lietotājs mijiedarbojas ar lietotāja saskarni, piemēram, ievada augu kopšanas datus vai apskata augu kopšanas ieteikumus, atbilstošie notikumi vai pieprasījumi tiks nosūtīti uz API. Pēc tam API apstrādās lietotāja ievadi, kas var ietvert augu datu izgūšanu vai atjaunināšanu datu bāzē, un ģenerēs atbildi. Šī atbilde var ietvert atjauninātus augu kopšanas datus, ieteikumus vai kļūdu ziņojumus atkarībā no konkrētā lietotāja pieprasījuma un sistēmas stāvokļa. Pēc tam atbilde tiks nosūtīta atpakaļ uz lietotāja saskarni, kur to var parādīt lietotājam.

Datubāzes - Šai augu kopšanas lietojumprogrammai būtu nepieciešama datu bāze, kurā var glabāt un pārvaldīt augu datus un lietotāju iestatījumus. Datubāzei jāspēj apstrādāt dažādus datu veidus, piemēram, augu nosaukumus, laistīšanas grafikus un mēslošanas līdzekļu izvēli, un jāspēj mērogot, pievienojot vairāk datu. Relāciju datu bāzes pārvaldības sistēma (RDBMS), piemēram, MySQL būtu laba izvēle šai lietojumprogrammai. Šīs datu bāzes ir strukturētas ap tabulām, kas var attēlot dažādus datu elementus augu kopšanas lietojumprogrammā.

Integrācija ar ierīces paziņojumu sistēmu - Lietojumprogramma var nosūtīt paziņojumus tieši uz lietotāja ierīci, izmantojot tādu pakalpojumu kā Firebase Cloud Messaging (FCM) vai Apple paziņojumu pakalpojums (APNS). Šie paziņojumi tiks parādīti kā uznirstošie logi lietotāja ierīcē un var ietvert informāciju, piemēram, kuriem augiem un kad ir nepieciešama laistīšana.

3.1.1. Ievads

Šī augu kopšanas lietojumprogramma ir izstrādāta, lai palīdzētu lietotājiem rūpēties par saviem augiem, sniedzot personalizētu informāciju un atgādinājumus, kas saistīti ar augu kopšanas uzdevumiem. Lietojumprogramma ļauj lietotājam ievadīt informāciju par saviem augiem, piemēram, to nosaukumiem, sugām. Pamatojoties uz šo informāciju, lietojumprogramma ģenerē atgādinājumus lietotājam veikt augu kopšanas uzdevumus, piemēram, laistīšanu, mēslošanu un atzarošanu. Lietojumprogramma sniedz arī ieteikumus augu kopšanas prakses uzlabošanai un žurnālu augu kopšanas darbības vēlākai atsaucei. Datu analīze sniedz ieskatus lietotājam, kā uzlabot augu kopšanas praksi, tostarp ieteikumus dažādām augu sugām, optimālu laistīšanas daudzumu un citu augu kopšanas paraugpraksi. Kopumā šī augu kopšanas lietojumprogramma palīdz lietotājiem uzturēt savus augus veselīgus un plaukstošus, sniedzot viņiem personalizētu un visaptverošu augu kopšanas informāciju un atgādinājumus.

3.1.2. Ievade

Lietotājs ievada informāciju par augiem: lietotājs lietojumprogrammā ievadīs informāciju par saviem augiem, piemēram, nosaukumus vai sugu. Šo informāciju var ievadīt manuāli vai izvelkot no augu datubāzes vai API.

3.1.3. Apstrāde

Šajā augu kopšanas lietojumprogrammā funkcija, kas apstrādā informāciju datu bāzē, parasti ietver lietotāja augu datu izvilkšanu. Funkcija izveidotu savienojumu ar datu bāzi un autentificētu lietotāju, pēc tam izgūtu attiecīgos datus, pamatojoties uz lietotāja ievadīto informāciju. Funkcija var izmantot SQL vaicājumus vai citas metodes, lai filtrētu un kārtotu datus pēc nepieciešamības. Kad dati ir izvilkti, funkcija tos var atjaunināt, pamatojoties uz lietotāja augu kopšanas darbībām, piemēram, laistīšanas pasākumiem. Funkcija varētu arī izmantot algoritmus, lai analizētu augu datus un sniegtu personalizētus ieteikumus lietotājam, piemēram, pielāgotu laistīšanas grafikus vai atzarošanas paņēmienus.

3.1.4. Izvade

Lietotne sniedz noderīgu informāciju par augu, pamācību, kā to iestādīt un kopt, kā arī automātiski pievieno atgādinājumus kalendārā, kad un kādā daudzumā ir jāaplaista katrs augs atkarīgi no izvēlētās sugas. Telefonā parādīsies paziņojums norādōt kurš augs, cik ml ūdens, vai ūdenim jābūt siltam vai vēsam, tiek dota informācija par auga stādīšanu, aplaistīšanu, ieteicamo istabas temperatūru, laistīšanas intervālu, raksturu un cikliem.

Paziņojumi - Konkrētā informācijas izsūtīšanas metode būtu atkarīga no lietotāja vēlmēm un nosūtāmās informācijas veida. Lietojumprogramma, visticamāk, ietvers iestatījumu izvēlni, kurā lietotāji varēs izvēlēties, kurām metodēm viņi dod priekšroku, un konfigurēt konkrētu informāciju, piemēram, paziņojumu biežumu un laiku.

3.2. Ārējās saskarnes prasības

Augu kopšanas lietojumprogrammu var viegli izmantot tālruņa ierīcēs, jo lielākā daļa cilvēku vienmēr nēsā līdzi savus tālruņus. Lietotāji varēs lejupielādēt lietojumprogrammu no Apple App Store vai autora tīmekļavietmes un lejuplādēt to savā tālrunī. Pēc uzstādīšanas lietotājam būs jāizveido konts un jāizveido savi augu profili, lai uzzinātu laistīšanas grafikus un nepieciešamo ūdens daudzumu.

3.2.1. Lietotāja saskarne

Lietotājs mijiedarbotos ar lietojumprogrammas saskarni, izmantojot skārienekrānu savā ierīcē. Lietojumprogramma nodrošinātu grafisku lietotāja saskarni (GUI), kas ļauj lietotājam apskatīt un mijiedarboties ar saviem augu profiliem. (Skatīt attēlu Nr. 1 pielikumā)

GUI parādīs lietotāja augu sarakstu, kā arī to pašreizējo laistīšanas grafiku un jebkuru citu būtisku informāciju, piemēram, augsnes mitruma līmeni, temperatūru un mitrumu. Pēc tam lietotājs var izvēlēties iekārtas profilu, lai skatītu detalizētāku informāciju vai veiktu izmaiņas laistīšanas grafikā. Lai veiktu izmaiņas, lietotājs var pieskarties attiecīgajai saskarnes sadaļai, piemēram, laistīšanas grafikam, un ievadīt jauno informāciju. Pēc tam lietojumprogramma atjaunināsdatubāzi ar jauno informāciju, un lietotājs saņems paziņojumu, kad būs pienācis laiks atkal laistīt iekārtu. Lietojumprogramma varētu ietvert arī papildu funkcijas, piemēram, iespēju pievienot fotogrāfijas vai piezīmes auga profilam, apskatīt vēsturiskos laistīšanas datus un saņemt padomus un padomus augu kopšanai. Visas šīs funkcijas būtu pieejamas, izmantojot lietojumprogrammas GUI, ļaujot lietotājam mijiedarboties ar saviem augu profiliem lietotājam draudzīgā un intuitīvā veidā.

3.3. Atribūti

3.3.1. Drošība

Lietotāju datu drošība un privātums ir svarīgs apsvērums jebkurai lietojumprogrammai, un šī augu kopšanas lietojumprogramma tam būtu jāizstrādā, lai nodrošinātu, ka lietotāja dati ir droši, lietojumprogramma varētu izmantot drošus šifrēšanas protokolus, lai aizsargātu sensitīvu informāciju, piemēram, lietotāja pieteikšanās datus un augu profilus. Datu bāzi var uzturēt drošā serverī ar ierobežotu piekļuvi, lai novērstu nevēlamu piekļuvi vai datu pārkāpumus.

3.3.2. Uzturamība

Viena no iespējām varētu būt lietojumprogrammas mitināšana mākoņa platformā, piemēram, Amazon Web Services (AWS). Mākoņa mitināšana nodrošinātu lietojumprogrammu ar mērogojamiem un uzticamiem resursiem, ļaujot tai apstrādāt dažādus trafika līmeņus un lietotāju pieprasījumu.

3.4. Citas prasības

3.4.1. Datu bāze API

Lai savienotu API ar datu bāzi, API būs jāizmanto atbilstoši datu bāzes draiveri vai bibliotēkas, lai sazinātos ar datu bāzes pārvaldības sistēmu (DBVS). API būs jāautentificē lietotājs un jānodrošina, ka pieprasītās datu bāzes darbības ir autorizētas un drošas. API būs jāapstrādā arī visas kļūdas vai izņēmumi, kas var rasties datu bāzes mijiedarbības laikā, piemēram, nederīgas ievades vai datu bāzes savienojuma kļūmes.

4. Koda apraksts

1. Datubāzes piekļuve *(Skatīt pielikumu Nr. 1)*

Šis Python kods importē "pieprasījumu" moduli, ko izmanto, lai vietnei veiktu HTTP pieprasījumus. Pēc tam kods definē URL un titulus ar API atslēgām, kas nepieciešamas, lai piekļūtu auga profila datubāzei. Tas izmanto pieprasījumu moduli, lai veiktu GET pieprasījumu API galapunktam un saglabātu atbildi mainīgajā "atbilde". Visbeidzot, kods izdrukā atbildi teksta formātā.

2. Klases definēšana *(Skatīt pielikumu Nr. 2)*

Šis Python kods definē klasi ar nosaukumu "Plant" ar konstruktora metodi "init". Konstruktora metode ietver piecus parametrus, proti, 'nosaukums', 'sugas', 'watering_interval', 'last_watered' un 'health'. Klasei ir pieci atribūti ar tādiem pašiem nosaukumiem kā konstruktoram nodotajiem parametriem. Šie atribūti atspoguļo auga nosaukumu, auga sugu, intervālu, kurā augs ir jālaista, pēdējo reizi, kad augs tika laistīts, un auga veselību. Kad tiek izveidots klases 'Plant' objekts, šie atribūti tiek inicializēti ar attiecīgajām vērtībām, kas tiek nodotas konstruktoram kā argumenti. Šo klasi var izmantot, lai šai lietojumprogrammai izveidotu augu profilu objektus.

3. Profilu izveide *(Skatīt pielikumu Nr. 3)*

Šis kods definē augu profilu vārdnīcu ar augu nosaukumiem kā taustiņiem un laistīšanas prasībām un vēlamajām vietām kā vērtībām. Kods liek lietotājam izvēlēties augu profilu, parādot pieejamo profilu sarakstu, un pēc tam lietotājam tiek prasīts ievadīt izvēlētā profila nosaukumu. Kods izgūst izvēlēto augu profilu no vārdnīcas un izdrukā atbilstošās laistīšanas prasības un vēlamo vietu. Ja lietotājs ievada nederīgu auga nosaukumu, kods izdrukā kļūdas ziņojumu. Ņemiet vērā, ka šis kods ir vienkāršots piemērs un tas būtu jāmaina, lai to integrētu ar plašāku augu kopšanas lietojumprogrammu.

4. Laika funkcija *(Skatīt pielikumu Nr. 4)*

Šis kods liek lietotājam ievadīt sekunžu skaitu starp atgādinājumiem un pēc tam iestata taimeri, kas katru sekundi pārbauda atgādinājumus. Kad pagājis laiks kopš pēdējā atgādinājuma ir lielāks par lietotāja norādīto intervālu, kods izdrukā atgādinājuma ziņojumu un atiestata taimeri. Šis process turpinās bezgalīgi, līdz lietotājs pārtrauc programmu. Kods ir vienkāršots piemērs un tas būtu jāmaina, lai to integrētu ar plašāku augu kopšanas lietojumprogrammu.

5. Paziņojumu sistēma *(Skatīt pielikumu Nr. 5)*

Dotais kods izmanto Pusher Push Notifications bibliotēku, lai izveidotu PushNotifications objektu, kam nepieciešams Pusher lietotnes instances ID un slepenā atslēga. Paziņojumu payload ir definēta "lietderīgās slodzes" vārdnīcā, kas ietver brīdinājuma ziņojumu un datus par rūpnīcas ID. Mainīgais "atbilde" tiek izmantots, lai nosūtītu paziņojumu konkrētam lietotāja interesentam, izmantojot "publicēšanas" metodi, kas izmanto paziņojumu, paziņojuma lietderīgo slodzi un papildu Firebase mākoņa ziņojumapmaiņas (FCM) datus, ja nepieciešams. Beigās kods izdrukā paziņojuma nosūtīšanas darbības statusa kodu un atbildes pamattekstu

5. Testēšana

Tabula Nr. 1

ID	Ievaddati	Sagaidāmais rezultāts	Reālais rezultāts
DB Requests	API.	Piekļuve pie datubāzes informācijas par augiem.	Izpildās
Class Plant	**Nosaukums, suga, laistīšanas intervāls, pēdējais laistais, veselība.**	**Tiek izveidota augu klase.**	**Izpildās**
Plant profiles	**Lietotājs ievada informāciju pēc input funkcijas.**	**Iespēja izveidot profilus.**	**Izpildās**
Time interval	**Intervāls.**	**Funkcionāla laika funkcija.**	**Izpildās**
PushNotifications	**API laistīšanas intervāls.**	**Tiek palaists paziņojums.**	**Izpildās**

6. Secinājumi un autora komentāri

Pētnieciskajā dokumentā ir izpētīta augu uzraudzības lietojumprogrammas prototips, kas var palīdzēt lietotājiem sekot līdzi viņu augu veselības un aprūpes vajadzībām. Darbā ir aprakstītas lietojumprogrammas iespējas, tostarp datubāze augu profiliem, paziņojumi augu veselības uzraudzībai. Darbā ir apskatīti arī lietojumprogrammas programmēšanas aspekti, kā API, augu profilu izveide, laika funkcijas un arī paziņošanas sistēmu ieviešana.

Veidojot šo darbu, autors ir guvis zināšanas dažādās jomās, tostarp datu bāzes pārvaldībā, API izstrādē un potenciālu mobilo aplikāciju projektēšanā. Šis pētnieciskais dokuments ir sniedzis vērtīgu ieskatu augu uzraudzības lietojumprogrammas izstrādes procesā un uzsvēris iespējamās funkcijas, ko tas var piedāvāt augu entuziastiem un hobijiem.

Izmantotā literatūra un informācijas avoti:

1. Oracle (2023) What Is a Database? [Skatīts tiešsaistē: 08.02.23] Pieejams: https://www.oracle.com/database/what-is-database/

2. Loopback (2023) Connect your API to a data source [Skatīts tiešsaistē: 08.02.23] Pieejams: https://loopback.io/doc/en/lb2/Connect-your-API-to-a-data-source.html

3. Matt Kostelnick (2022) The Ultimate Guide to Indoor Plants [Skatīts tiešsaistē: 26.01.23] Pieejams: https://www.ambius.com/blog/the-ultimate-guide-to-indoor-plants/

4. Ema Roso (2023) House Plants API Documentation [Skatīts tiešsaistē: 26.01.23] Pieejams: https://rapidapi.com/rosoemawd/api/house-plants

5. Sven "Coding is fun" (2021) Send Daily Push Notifications to Your Phone Using Python [Skatīts 30.01.23] Pieejams: https://www.youtube.com/watch?v=tbzPcKRZlHg

Pielikumi

Attēls Nr. 1 – "Augo lietotājprogrammas autora veidotais paraugs"

1. Datubāzes piekļuve

```
import requests

#izvelkam no interneta atrasto datubāzes infromāciju ar doto API atslēgu
url = "https://house-plants.p.rapidapi.com/common/coralberry"

headers = {
    "X-RapidAPI-Key": "f10fb260e7msh71aa22b9522252bp12693djsn8ce10cefbceb",
    "X-RapidAPI-Host": "house-plants.p.rapidapi.com"
}
response = requests.request("GET", url, headers=headers)

print(response.text)
```

Pielikums Nr. 1 – "Datubāzes piekļuves koda daļa"

2. Klases definēšana

```
class Plant:
    def __init__(self, name, species, watering_interval, last_watered,
health):
        self.name = name
        self.species = species
        self.watering_interval = watering_interval
        self.last_watered = last_watered
        self.health = health
```

Pielikums Nr. 2 – "Klases definēšanas koda daļa"

3. Profilu izveide

```
# Definējiet augu profilu vārdnīcu ar augu nosaukumiem kā atslēgām un
profiliem kā vērtībām
plant_profiles = {
    'Monstera deliciosa': {
        'watering_requirements': 'once per week',
        'preferred_location': 'bright, indirect light'
    },
    'Spider Plant': {
        'watering_requirements': 'once per week',
        'preferred_location': 'bright, indirect light'
    },
    'Peace Lily': {
        'watering_requirements': 'once per week',
        'preferred_location': 'bright, indirect light'
```

```
    }
}
# Lūdz lietotājam izvēlēties augu profilu

print("Choose a plant profile:")
for plant_name in plant_profiles.keys():
    print(f"- {plant_name}")
selected_plant = input("Enter the name of the plant profile: ")

# Izgūt izvēlēto augu profilu no vārdnīcas
if selected_plant in plant_profiles:
    plant_data = plant_profiles[selected_plant]
    print(f"Watering Requirements: {plant_data['watering_requirements']}")
    print(f"Preferred Location: {plant_data['preferred_location']}")
else:
    print(f"No plant profile found for '{selected_plant}'")
```

Pielikums Nr. 3 – "Profilu izveides koda daļa"

4. Laika funkcija

```
import time

# Jautā datubāzei sekunžu skaitu starp atgādinājumiem
interval = int(input(
for i in plant.items():
  if i == "plant_time":)
))

# Iestata sākotnējo laiku pirmajam atgādinājumam
start_time = time.time()

while True:
    # Aprēķina laiku, kas pagājis kopš pēdējā atgādinājuma
    elapsed_time = time.time() - start_time

    # Ja pagājušais laiks ir lielāks par atgādinājuma intervālu, izdrukā
atgādinājuma ziņojumu un atiestata taimeri
    if elapsed_time > interval:
        print("Time to water the plants!")
        start_time = time.time()

    # Pagaidiet 1 sekundi, pirms vēlreiz pārbaudiet taimeri
    time.sleep(1)
```

Pielikums Nr. 4 – "Laika funkcijas koda daļa"

5. Paziņojumu sistēma

```python
from pusher_push_notifications import PushNotifications

# izveido PushNotifications gadījumu
pusher = PushNotifications(
    instance_id='INSTANCE_ID_HERE',
    secret_key='SECRET_KEY_THAT_WE_DONT_HAVE',
)

# definē paziņojumu payloadu
payload = {
    'aps': {
        'alert': {
            'title': 'Plant watering reminder',
            'body': 'It is time to water your plants!',
        },
        'badge': 1,
        'sound': 'default',
    },
    'data': {
        'plant_id': 'PLANT_ID_HERE',
    },
}

# nosūta paziņojumu uz noteiktu ierīci
response = pusher.publish(
    interests=['INTEREST_HERE'],
    notification=payload,
    fcm={
        'data': {
            'plant_id': 'PLANT_ID_HERE',
        },
    },
)

# izdrukā atbildes statusu un pamattekstu
print(response.status_code)
print(response.json())
```

Pielikums Nr. 5 – "Paziņojuma sistēma koda daļa"