**Piekļuves darbs Programmēšanas II eksāmenam**

**Ideju menedžeris**

**Darba autors:**

Olivers Nils Robs

**Saturs**

2. Problēmas izpēte un analīze

4. Programmatūras prasību specifikācija

6. Programmatūras izstrādes plāns

7. Atkļūdošanas un akcepttestēšanas pārskats

10. Lietotāja ceļvedis

12. Piemērotās licences pamatojums

13. Programmatūras kods

**Problēmas izpēte un analīze**

Ikdienas darba process, kuru varētu automatizēt, ir ideju novērtēšana un dokumentēšana projekta vadības nolūkos. Bieži vien darba grupās idejas tiek ierosinātas un diskutētas, bet ne vienmēr tās tiek pienācīgi dokumentētas un novērtētas, kas var novest pie nepilnīgas projekta vadības un nespējas izmantot vērtīgas idejas.

Ideju dokumentēšanas un novērtēšanas procesa automatizēšana uzlabotu projekta vadības efektivitāti un nodrošinātu, ka visi darba grupas locekļi ir iesaistīti un informēti par diskusiju gaitu un ideju stāvokli. Idejas varētu ievadīt tieši platformā, tās varētu tikt novērtētas un iekļautas tabulā, kuru varētu kārtot pēc datuma un vērtējuma. Turklāt, meklēšanas funkcionalitāte varētu palīdzēt viegli atrodot iepriekšējās idejas, un pop-up logi varētu dot iespēju apskatīt idejas aprakstus bez tabulas aizvēršanas.

**Problēma:** Ideju dokumentēšana un novērtēšana darba grupās var būt nepietiekama un neorganizēta, kas var novest pie nepilnīgas projekta vadības un vērtīgu ideju neizmantošanas.

**Darba uzdevums:** Izveidot programmatūru, kas automatizē ideju dokumentēšanas un novērtēšanas procesu.

Dažādas izpētes metodes var būt noderīgas, lai veiktu ikdienas procesu izpēti un analīzi, atkarībā no izpētītās problēmas un jautājumiem. Šeit ir dažas izpētes metodes, kas varētu būt piemērotas:

* Intervijas - Izmantojot intervijas, var veikt sarunas ar cilvēkiem, kas tieši iesaistīti ikdienas procesā. Tas var palīdzēt saprast viņu pieredzi, problēmas un ieteikumus, kā uzlabot procesu.
* Observācija - Observējot ikdienas procesu, var iegūt izpratni par darba gaitām, kas tiek veiktas, un identificēt vietu, kur varētu tikt ieviestas automatizācijas iespējas.
* Anketēšana - Izmantojot anketēšanu, var iegūt plašāku aptauju dalībnieku un iegūt kvantitatīvu informāciju par procesu, tā darbību un iespējamajām problēmām.
* Fokusa grupas diskusijas - Fokusa grupas diskusijas var palīdzēt saprast dažādus iespējamos risinājumus un ieteikumus, kas var uzlabot ikdienas procesu, un iegūt perspektīvu no dažādiem procesa dalībniekiem.

Salīdzinot šīs metodes, lai veiktu dažādu ikdienas procesu izpēti un analīzi, man šķiet, ka piemērotākā metode būtu observācija, jo tā ļauj iegūt detalizētu izpratni par procesu un identificēt vietu, kur varētu tikt ieviestas automatizācijas iespējas. Lai gan intervijas un anketēšana var sniegt vērtīgu informāciju, tomēr, ja process ir sarežģīts vai tajā ir daudz kārtojumu, var būt grūti iegūt pilnīgu izpratni par to, kā process tieši notiek. Fokusa grupas diskusijas varētu būt noderīgas, lai apspriestu dažādus iespējamos risinājumus, taču tās var būt mazāk efektīvas, ja process ir ļoti specifisks vai sarežģīts.

**Kas ir konkrētais process?**

Ideju pierakstīšanas process ietver ideju ievadīšanu, apraksta un tagu pievienošanu, ideju uzkrāšanu datubāzē un to vērtēšanu.

**Kurus procesa elementus ir iespējams automatizēt?**

Ideju ievadīšanas, apraksta un tagu pievienošanas un to uzkrāšanas datubāzē var automatizēt. Turklāt, var izmantot meklēšanas un kārtošanas funkciju, lai meklētu idejas pēc idejām un tagiem, kārtotu tās pēc datuma un vērtējuma.

**Kāpēc procesa automatizācija ir nepieciešama?**

Ikdienā mēs bieži pierakstām un uzkrājam daudz ideju, kas var būt grūti pārvaldāmas, kad to skaits pieaug. Procesa automatizācija var uzlabot darba efektivitāti un ļauj uzkrāt lielāku ideju skaitu, kā arī palīdzēt tos vērtēt un organizēt.

**Kam un kādi būs ieguvumi no procesa automatizācijas?**

Ieguvumi no procesa automatizācijas būs visiem, kas iesaistīti ideju pierakstīšanas procesā. Darba devējs varēs iegūt labāku pārvaldību un uzskaiti par idejām, bet darbinieki varēs vieglāk pierakstīt idejas un vērtēt tos. Turklāt, tas varētu veicināt inovatīvāku un efektīvāku darba vietu kultūru.

**Izpētes procesa dokumentēšana:**

1. Nosaka izpētes mērķus un jautājumus.
2. Izpēta pieejamās izpētes metodes un izvēlas piemērotāko.
3. Veic izpēti, kuru rezultātā tiek apkopota informācija par procesu un tā automatizācijas iespējām.
4. Analizē iegūtos datus un identificē iespējas procesa automatizācijai.
5. Dokumentē iegūtos rezultātus un izmanto tos, lai veiktu turpmākus uzlabojumus.

**Programmatūras prasību specifikācija**

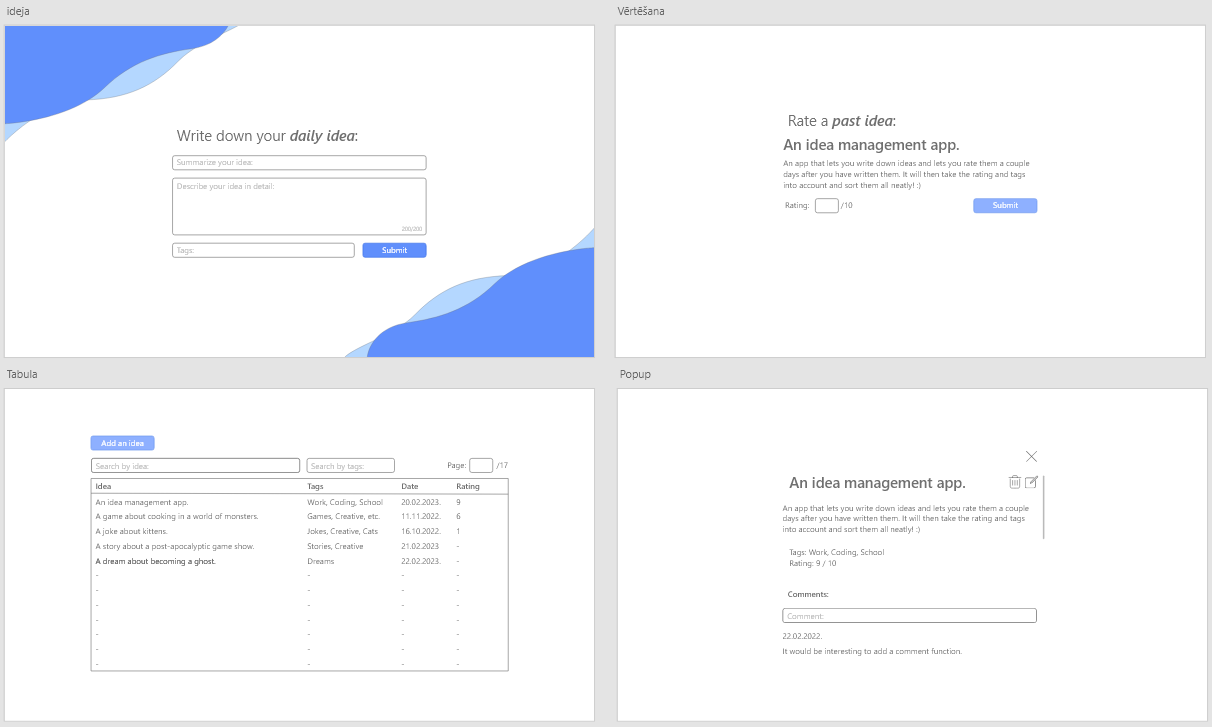
**Problēma:** Komanda vēlas izveidot sistēmu, kas ļauj lietotājiem pierakstīt, organizēt un vērtēt idejas, lai veicinātu komandas inovācijas un attīstības stratēģiju.

**Galvenais mērķis:** Izstrādāt automatizācijas risinājumu, kas uzlabo ideju pierakstīšanas, organizēšanas un vērtēšanas procesu.

Funkcionālās prasības:

* Izstrādāt datubāzes, kas uzglabās ideju informāciju, tostarp aprakstu, datumu un vērtējumu, un kas uzglabās tagu informāciju, un kas uzglabās ideju un tagu saistījumu.
* Izstrādāt automatizācijas risinājumu, kas ļauj automātiski ievadīt ideju informāciju datubāzē.
* Izstrādāt meklēšanas un kārtošanas funkcijas, kas spēj meklēt pēc idejām un tagiem un kas spēj kārtot idejas pēc datuma un vērtējuma.
* Izstrādāt vērtējumu sistēmu, kas ļauj lietotājiem vērtēt idejas.
* Izstrādāt sistēmu, kas ļauj lietotājiem dzēst esošās idejas.

Nefunkcionālās prasības:

* Izstrādāt lietotājam draudzīgu interfeisu, kas ļauj viņiem pierakstīt, organizēt un vērtēt idejas.
* Izstrādāt interaktīvus filtrus, kas ļauj meklēt pēc idejām un tagiem un kas ļauj kārtot idejas pēc datuma un vērtējuma.
* Izstrādāt pop-up logus, kas ļauj lietotājam apskatīt sīkāku informāciju par idejām.
* Izstrādāt vērtējumu sistēmu, kas ļauj lietotājiem vērtēt idejas.
* Nodrošināt atbalsta dokumentāciju.

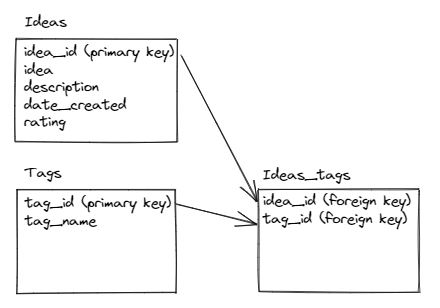
1.att. Sistēmas dizaina un lapu skice.

**Lietotāju lomas un paredzētās darbības:**

Lietotāji: Ikdienas lietotāji, kas pievieno idejas, to organizē un vērtē. Darbības:

* Pierakstīt idejas, pievienojot aprakstu un tagus.
* Skatīt un organizēt esošās idejas, piemēram, izmantojot filtrus un meklēšanu.
* Vērtēt idejas.
* Dzēst esošās idejas.

**Datu klases:**

1. Ideju datubāze ar informāciju par idejām, aprakstiem, datumiem un vērtējumiem.
2. Tagu datubāze ar informāciju par tagiem.
3. Ideju un tagu datubāze, kas saista idejas ar tagiem.

2.att. Sistēmas datubāzes skice.

**Programmatūras darbības apraksts:**

* Ideju pierakstīšana: Lietotāji var pievienot jaunas idejas, ievadot pašu ideju, tagus un aprakstu. Automātiski tiek piešķirts un ievietots arī izveidošanas datums.
* Ideju organizēšana: Lietotāji var apskatīt savas esošās idejas un organizēt tās, izmantojot filtrus un meklēšanas funkcijas. Sistēma ļauj filtrēt idejas un kārtot tās pēc datuma un vērtējuma.
* Ideju vērtēšana: Lietotāji var vērtēt idejas, sistēma uzglabā vērtējuma datus.
* Ideju dzēšana: Lietotāji var dzēst esošās idejas no sistēmas.
* Ideju un tagu saistīšana: Sistēma sasaista idejas ar tagiem, izmantojot ideju un tagu datubāzes. Tas palīdzēs lietotājiem atrast un organizēt idejas pēc tagiem.
* Atbalsta dokumentācija: Sistēma nodrošina atbalsta dokumentāciju, kas palīdzēs lietotājiem izprast un izmantot sistēmu.

**Cita informācija:**

Sistēma izstrādāta, izmantojot SQLite datubāzi un JavaScript programmēšanas valodu, kas tiek izmantota SQLite vienkāršības dēļ. Lietotājam draudzīgs interfeiss izstrādāts, izmantojot React Router DOM un JavaScript, kas izmantots, jo ļauj sasaistīt komponentus un izveidot pārskatāmu failu struktūru. Sistēma tiek uzglabāta un darbojas uz servera, kurā ir uzstādīta nepieciešamā programmatūra un faili.

**Programmatūras izstrādes plāns**

Programmatūras izstrādes modeļus var iedalīt divos plašos virzienos: tradicionālajā un iteratīvajā. Tradicionālie modeļi paredz, ka visi projektā iesaistītie elementi tiek precīzi definēti un izstrādāti pirms izstrādes darbu sākuma. Iteratīvie modeļi paredz, ka izstrādes darbi tiek sadalīti ciklos, kas ļauj izstrādāt programmatūru daļu pa daļai, pakāpeniski uzlabojot to.

Ņemot vērā sistēmas prasības un datu bāzes lielumu, iteratīvā modeļa piemērošana šajā gadījumā varētu būt piemērotāka. Viens no populārākajiem iteratīvajiem modeļiem ir Agile, kas paredz, ka izstrādājamās programmatūras izstrādāšana notiek ciklos, kas saukti par sprintiem. Katrā sprintā programmētāji pievieno un uzlabo programmatūras funkcijas, kuras tad tiek demonstrētas klientam. Klients var sniegt atsauksmes, kas palīdz uzlabot programmatūras izstrādi. Tas palīdz izvairīties no kļūdām un uzlabo produktu.

Pēc Agile iteratīvā modeļa veidots programmatūras izstrādes plāns:

1. Posms: Projektu plānošana un sagatavošana (4 dienas)

* Identificēt programmatūras izstrādes mērķus un prasības.
* Izveidot sistēmas vides dizaina un datubāzes tabulu un to relāciju skices.
* Sagatavot vides izstrādei, ieskaitot datubāzes un programmēšanas vides izveidi.

2. Posms: Pirmās funkcijas izstrāde (10 dienas)

* Izstrādāt lietotājam draudzīgu interfeisu, kas ļauj viņiem pierakstīt, organizēt un vērtēt idejas.
* Izstrādāt vērtējumu sistēmu, kas ļauj lietotājiem vērtēt idejas.
* Izveidot meklēšanas un kārtošanas funkcijas, kas spēj meklēt pēc idejām un tagiem un kas spēj kārtot idejas pēc datuma un vērtējuma.
* Izveidot datubāzes, kas uzglabās ideju informāciju, tostarp aprakstu, datumu un vērtējumu, un kas uzglabās tagu informāciju, un kas uzglabās ideju un tagu saistījumu.
* Izveidot pop-up logus, kas ļauj lietotājam apskatīt sīkāku informāciju par idejām, kā to aprakstus, tagus, datumus, kad tika izveidota ideja un vērtējumus.

3. Posms: Funkciju papildināšana un uzlabošana (4 dienas)

* Izveidot meklēšanas un kārtošanas funkcijas, kas spēj meklēt pēc idejām un tagiem un kas spēj kārtot idejas pēc datuma un vērtējuma.
* Izstrādāt sistēmu, kas ļauj lietotājiem dzēst esošās idejas.

**Atkļūdošanas un akcepttestēšanas pārskats**

**Testēšana**

* 1. Idejas ievade (obligāta):

Neko neievadot, parādās kļūda.

|  |
| --- |
|  |

Ideja parādās vērtēšanas lapā, tabulā un popup.

|  |
| --- |
| Attēls, kurā ir teksts  Apraksts ģenerēts automātiski |

Ir iespējams meklēt pēc idejas.

|  |
| --- |
|  |

* 1. Apraksta ievade (nav obligāta):

Apraksts parādās vērtēšanas lapā un popup.

|  |
| --- |
| Attēls, kurā ir teksts  Apraksts ģenerēts automātiski |

* 1. Tagu ievade (obligāta):

Neko neievadot, parādās kļūda.

|  |
| --- |
| Attēls, kurā ir teksts  Apraksts ģenerēts automātiski |

Ievades laukā ir iespējams ievadīt tagus, kuri atkārtojas, bet tie neparādās atkārtoti vērtēšanas logā, tabulā vai popup (atdalot ar komatu).

|  |
| --- |
|  |

Ir iespējams meklēt pēc taga.

|  |
| --- |
|  |

* 1. Vērtējuma ievade:

Pēc noklusējuma ir nulle.

|  |
| --- |
|  |

Ievadot tukšu vērtību, parādīs nākamo ideju, kuru vērtēt.

|  |
| --- |
|  |

Ievadot vērtību, kas ir mazāka par viens bet lielāka par 10, parādās kļūda.

|  |
| --- |
|  |

Ievadītā vērtība parādās tabulā un popup.

|  |
| --- |
|  |

Ir iespējams kārtot pēc vērtējuma

|  |
| --- |
|  |

* 1. Datums:

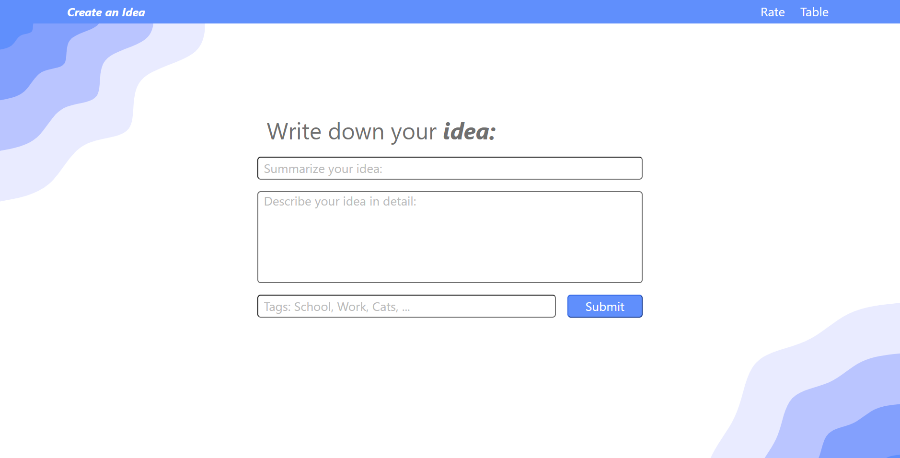
Parādās tabulā un popup.

|  |
| --- |
|  |

Ir iespējams kārtot pēc datuma.

|  |
| --- |
|  |

**Lietotāja ceļvedis**

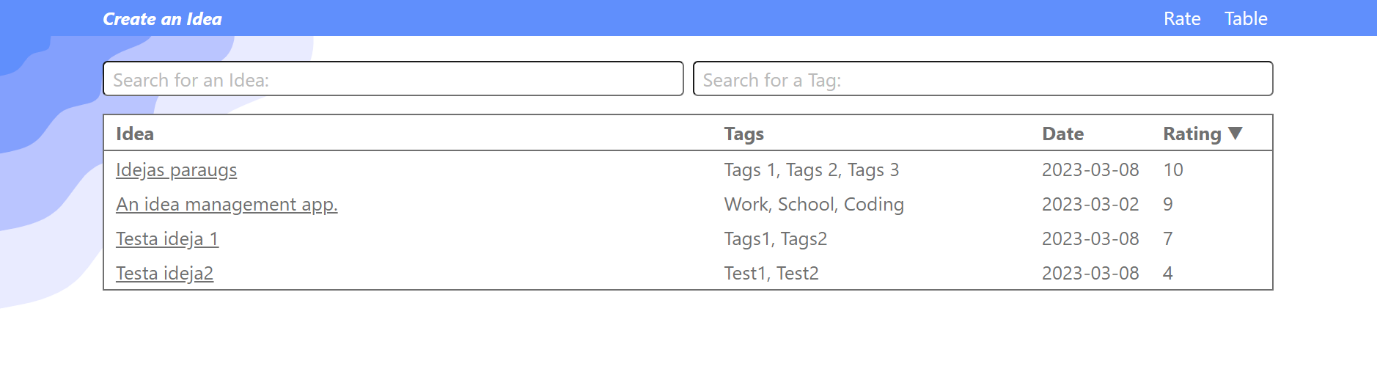
1. Pēc programmas un datubāzes palaišanas, iet uz <http://localhost:5173/>, kur būs redzama šī lapa:

3.att. Index lapa.

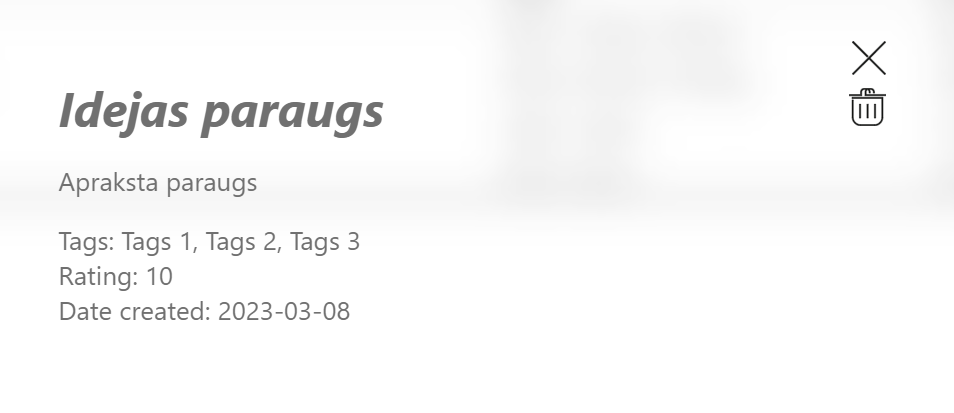
1. Pašā augšā ir trīs pogas: *Create an Idea*, kas ved uz sākuma lapu, kur ir iespējams ievadīt ideju, aprakstu un tagus, *Rate*, kas ved uz lapu, kur ir iespējams novērtēt ideju, un *Table*, kur ir iespējams apskatīt idejas tabulā un izdzēst tās.
2. Sākuma lapā ir trīs lauki, kuros ir iespējams ievadīt vērtības. Pirmajā laukā ir jāievada idejas nosaukums. Šis lauks nedrīkst palikt tukšs. Otrajā laukā ir iespējams detalizēti aprakstīt savu ideju. Pēdējais lauks ir tagiem, kurus ir iespējams atdalīt ar komatiem. Šis lauks ir obligāts. Spiežot uz *Submit* pogas, ideja tiek aizsūtīta uz datubāzi.
3. *Attēls, kurā ir teksts

   Apraksts ģenerēts automātiskiRate* lapā ir iespējams apskatīt vēl nenovērtētās idejas un tās novērtēt skalā no 1 līdz 10. Vērtējuma noklusējuma vērtība ir 0. Ja *Rating* lauks ir atstāts tukšs un tiek uzspiesta *Submit* poga, tad tiek rādīta nākamā nenovērtētā ideja, ja tāda eksistē. Ja vairs nav ideju, kuru vērtēt, rādīsies šī ziņa - *There are no Ideas to Rate*.

4. att. Rate lapa

1. *Table* lapā ir iespējams apskatīt visas ievadītās idejas, kopā ar to tagiem, datumiem un vērtējumiem. Izmantojot pirmo ievadlauku, kas atrodas virs tabulas, ir iespējams meklēt ideju pēc tās nosaukuma. Izmantojot otro ievadlauku, ir iespējams meklēt ideju pēc tagiem. Spiežot uz *Date* vai *Rating* tabulas virsrakstiem, ir iespējams kārtot tabulu pēc to vērtībām. Ja uzspiedīs tabulas virsrakstu vairākas reizes, tā kārtošanas secība mainīsies starp augošu un dilstošu.

5.att. Table lapa

1. *Table* lapā, spiežot uz jebkuru vērtību *Idea* kolonnā, lapu pārsegs logs, kurā ir iespējams redzēt detaļas par ideju, ieskaitot pašu ideju, aprakstu, tagus, datumu un vērtējumu. Spiežot uz krustiņu ir iespējams nokļūt atpakaļ pie tabulas. Spiežot uz miskastes ikonu, ir iespējams izdzēst vienumu no tabulas un datubāzes.

6.att. Popup

**Piemērotās licences pamatojums**

Ir dažādi programmatūras licenču veidi, un katrai no tām ir savs noteikumu un nosacījumu kopums. Šeit ir dažas no visizplatītākajām programmatūras licencēm:

* GPL (GNU vispārējā publiskā licence): šī licence ir paredzēta, lai nodrošinātu, ka jebkurai programmatūrai, kas izmanto saskaņā ar GPL licencētu kodu, jābūt pieejamai saskaņā ar to pašu licenci. GPL ir copyleft licence, kas uzsver bezmaksas programmatūras nozīmi.
* Apache: Apache licence ir pieļaujama licence, kas ļauj izmantot, modificēt un izplatīt programmatūru noteiktos apstākļos. Šī licence bieži tiek izmantota atvērtā pirmkoda programmatūrai, kas paredzēta darbam ar Apache tīmekļa serveri.
* BSD: BSD licence ir pieļaujama licence, kas ļauj izmantot, modificēt un izplatīt programmatūru noteiktos apstākļos. BSD licence bieži tiek izmantota atvērtā pirmkoda programmatūrai, kas paredzēta darbam ar BSD operētājsistēmām.
* MIT: MIT licence ir atļauja, kas ļauj izmantot, modificēt un izplatīt programmatūru noteiktos apstākļos. Šī licence bieži tiek izmantota atvērtā pirmkoda programmatūrai, kuru paredzēts iekļaut patentētā programmatūrā.

No šīm licencēm MIT licence ir laba izvēle daudziem atvērtā pirmkoda projektiem, jo ​​tā ir pieļaujama licence, kas nodrošina lielu elastību. MIT licence ļauj lietotājiem izmantot, modificēt un izplatīt programmatūru, ievērojot noteiktus nosacījumus, tostarp prasību programmatūras dokumentācijā iekļaut licences un autortiesību paziņojuma kopiju.

Vēl viena MIT licences priekšrocība ir tā, ka tā ir saderīga ar daudzām citām licencēm, tostarp GPL un Apache licenci, kas nozīmē, ka izstrādātāji var izmantot saskaņā ar šīm licencēm licencētu kodu programmatūrā, kas tiek izplatīta saskaņā ar MIT licenci.

Kopumā MIT licence nodrošina labu līdzsvaru starp programmatūras veidotāju tiesību aizsardzību un programmatūras plašas izmantošanas un modifikācijas iespēju.

**Programmatūras kods**

Kods ir pieejams saitē:  
https://github.com/ONRobs/Idea-Manager