Rīgas Tālmācības vidusskola

Programmēšana II projekts – Produktu sekotājs

Darba autors: Tomass Alberts Bezdelīga

Rīga 2025

**Satura rādītājs**

[I. Problēmas izpēte un analīze 4](#_Toc195644165)

[1. Problēmas apraksts: 4](#_Toc195644166)

[2. Izpētes metode: 4](#_Toc195644167)

[3. Izpētes procesa apraksts: 4](#_Toc195644168)

[4. Izpētes datu apkopojums: 5](#_Toc195644169)

[5. Projekta mērķis: 5](#_Toc195644170)

[II. Programmatūras prasību specifikācija 5](#_Toc195644171)

[1. Saimniecības ‘‘Virgenes’’ prasības programmai: 5](#_Toc195644172)

[2. Papildus (administratīvās) prasības programmai: 7](#_Toc195644173)

[3. Izvēlētā izstrādes vide un valoda: 10](#_Toc195644174)

[III. Programmatūras izstrādes plāns 11](#_Toc195644175)

[1. Izstrādes modelis: 11](#_Toc195644176)

[2. Izstrādes soļi: 11](#_Toc195644177)

[IV. Atkļūdošanas un akcepttestēšanas pārskats 12](#_Toc195644178)

[1) Atkļūdošanas pārskats: 12](#_Toc195644179)

[2) Akcepttestēšana pārskats: 12](#_Toc195644180)

[3) Secinājumi: 13](#_Toc195644181)

[V. Lietotāja ceļvedis 13](#_Toc195644182)

[1. Instalācija: 13](#_Toc195644183)

[1) Sistēmas prasības: 13](#_Toc195644184)

[2) Instalācijas instrukcijas: 14](#_Toc195644185)

[2. Lietotāja reģistrācija, pieteikšanās un konta pārvalde: 14](#_Toc195644186)

[1) Reģistrācijas process: 14](#_Toc195644187)

[2) Pieteikšanās procesa apraksts: 14](#_Toc195644188)

[3) Konta pārvaldība: 14](#_Toc195644189)

[3. Galveno funkciju apraksts un navigācija: 14](#_Toc195644190)

[1) Lietotāja galvenie skati un navigācija: 14](#_Toc195644191)

[2) Interfeisa elementi un mijiedarbība: 15](#_Toc195644192)

[4. Administratīvās funkcijas un papildus iespējas: 15](#_Toc195644193)

[1) Administratora panelis un funkcionalitātes: 15](#_Toc195644194)

[2) Tehniskā uzturēšana un rezerves kopiju veidošana: 15](#_Toc195644195)

[VI. Piemērotākā licence 15](#_Toc195644196)

[VII. Programmatūras kods 16](#_Toc195644197)

[1. main.py : 16](#_Toc195644198)

[2. admin.html: 39](#_Toc195644199)

[3. change\_password.html: 41](#_Toc195644200)

[4. edit\_delivery.html: 42](#_Toc195644201)

[5. historya.html: 43](#_Toc195644202)

[6. historym.html: 44](#_Toc195644203)

[7. index.html: 46](#_Toc195644204)

[8. layout.html: 47](#_Toc195644205)

[9. login.htm: 47](#_Toc195644206)

[10. member\_history.html: 48](#_Toc195644207)

[11. members.html: 49](#_Toc195644208)

[12. registrationa.html: 51](#_Toc195644209)

[13. registrationm.html: 53](#_Toc195644210)

[14. shares.html: 54](#_Toc195644211)

[15. statusa.html: 56](#_Toc195644212)

[16. statusm.html: 59](#_Toc195644213)

# Problēmas izpēte un analīze

## Problēmas apraksts:

Ekoloģiskā saimniecība “Virgenes” sastopas ar grūtībām, kas saistītas ar preču izsekojamību un piegādes procesa pārvaldību. Pašreiz darbiniekiem manuāli jāseko līdzi tam, kas un cik daudz no saražotā tiek saņemts vai piegādāts klientam. Tas prasa lielu uzticēšanos un rada risku datu neprecizitātēm vai nesakritībām, īpaši gadījumos, kad klients ierodas pats, lai saņemtu savu pirkumu.

Lai šos riskus samazinātu un uzlabotu procesu pārskatāmību, saimniecība vēlas ievest automatizētu risinājumu – sistēmu, kas reģistrē katru darījumu, saglabā informāciju par preču kustību un ļauj gan klientiem, gan darbiniekiem ērti piekļūt nepieciešamajiem datiem.

## Izpētes metode:

Es izvēlējos pāris izpētes metodes, lai noteiktu, ko saimniecībai vajadzētu automatizēt. Tā dēļ, es izmantoju strukturētu interviju un novērojumus saimniecības darbības vidē. Tas man ļāva gūt padziļinātu izpratni par ikdiena procesiem un klientu vajadzībām.

Papildus jāatzīmē, ka esmu strādājis šajā saimniecībā kā darbinieks, līdz ar to man bija iespēja praktiski novērot un izprast ikdienas procesus, problēmsituācijas un klientu saskarsmes aspektus. Šī pieredze man sniedza nozīmīgu ieskatu un ļāva objektīvi noteikt, kādas funkcijas ir kritiskas sistēmai.

## Izpētes procesa apraksts:

Tika veikta intervija ar pasūtītāju – saimniecības pārstāvi, kuras laikā tika identificētas galvenās problēmas un vēlmes saistībā ar programmatūras funkcionalitāti. Novērošanā tika lietota, lai noskaidrotu esošos ikdienas darbības scenārijus, kas ietver preču saņemšanu, piegādes procesu un klientu apkalpošanu. Tā kā es esmu jau iepriekš strādājis šajā saimniecībā kā darbinieks, tika izmantotas arī personīgās zināšanas par ikdienas rutīnu un tās sāpīgajiem punktiem.

## Izpētes datu apkopojums:

* Klientiem ir atšķirīgi preču apkopojumi atkarībā no, kādu abonementu veidu viņi ir nopirkuši;
* Reģistrācija notiek manuāli vai tiek veikta pēc atmiņas;
* Trūkst vienotas sistēmas, kurā redzami dati par to, kas, kad un cik daudz ir saņēmis;
* Darbiniekiem nav iespējas viegli papildināt vai pārvaldīt ražas daļu tipus;
* Klientiem nav iespējas pārskatīt savu vēsturi vai statusu.

## Projekta mērķis:

Izveidot sistēmu saimniecībai ‘‘Virgenes’’, kas ļauj:

* Piekļūt datiem, caur dalībnieku vai administratīva konta;
* Saimniecības darbiniekiem ievadīt maiņas preču skaitā;
* Klientiem reģistrēt, ko viņi ir paņēmuši no fermas;
* Klientiem reģistrēt, cik daudz viņi ir paņēmuši no fermas;
* Klientiem reģistrēt, kad viņi ir paņēmuši no fermas;
* Klientiem redzēt savu konta statusu;
* Klienta status satur viņu gada ražas daļas atlikumu, no tā, ko viņi jau ir savākuši;
* Klientiem iespēja redzēt tabulu ar, ko un kad viņi ir paņēmuši no fermas;
* Iespēja iziet no sava konta.

# Programmatūras prasību specifikācija

## **Saimniecības ‘‘Virgenes’’ prasības programmai:**

Šīs ir prasības, ko klients pieprasīja programmai, kad tika sarunāts, kam tur vajadzētu būt.



* Lietotāja iespēja ‘Login’, jeb piekļūt, savam kontam;
* Lai piekļūtu, lietotājam ir jāievada savs lietotājvārds un parole teksta laukos;
* Jābūt lietojamam uz telefona, datora vai planšetdatora;
* Jābūt pieejamam ar interneta savienojumu;
* Poga, kuru nospiežot lietotājs, ja ir ievadījis pareizu lietotājvārdu un paroli, piekļūst savam kontam, ja ir ievadīts neeksistējošs lietotājvārds vai parole, parāda kļūdas ziņu;
* Piekļūstot savam kontam lietotāju aizved uz savu STATUS, ja tas ir klients, ADMIN, ja tas ir darbinieks;



* Lietotājam dabūjot piekļuvi, ir iespēja reģistrēt datubāzē datus par, ko viņi fermā savāc;
* Izvēlne, kur lietotājs var izvēlēties skaitu preču no savām ražas daļām;
* Kalendārs, kur parāda, kurā datumā, lietotājs ir saņēmis šo pirkumu;
* Izvēlne, kur lietotājs var izvēlēties opciju jā, nē vai neizvēli, par vai viņi ņems ekstra olas vistas vietā;
* Poga, kuru spiežot ievada datus datubāzē;
* Poga, kuru spiežot lietotājs nonāk atpakaļ uz savu STATUS;



* Lietotājs var redzēt savu STATUS;
* Lietotājs var redzēt savu vārdu;
* Lietotājs var redzēt savu ražas daļas tipu;
* Lietotājs var redzēt, vai viņi ir ņēmušu papildus olas vistas vietā;
* Lietotājs var redzēt, cik daudz preces viņiem ir bijušas piegādātas vai paši saņēmuši;
* Lietotājs var redzēt, cik daudz preces viņiem ir palikušas no tā gada ražas;
* Lietotājs var redzēt, vai viņiem ir palikušas preces no iepriekšējā gada ražas;
* Poga, kuru spiežot lietotājs var iziet no sava konta;
* Poga, kuru spiežot lietotājs nonāk pie REGISTRATION;
* Poga, kuru spiežot lietotājs nonākt savā HISTORY;



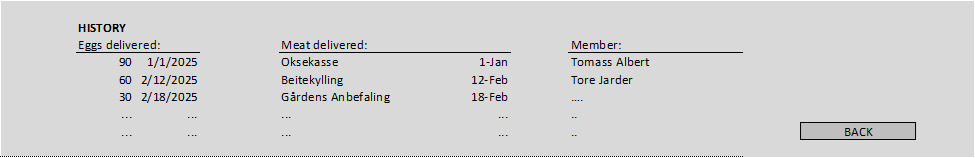
* Lietotājs var redzēt, cik daudz, kad, un ko viņi ir saņēmuši no fermas;
* Poga, kuru spiežot lietotājs nonāk atpakaļ uz savu STATUS;

## Papildus (administratīvās) prasības programmai:

Programma, lai būtu ilgstoši lietojama un noderīga klientam, tai vajag veidu, kā administratīvi papildināt tās datni. Piemēram, klients vēlas papildināt ražas daļas tipus, ko pircējs klienta var iegādāties, vai vēlas, kādu noņemt.



* Darbinieks, ieejot administratīvajā kontā var redzēt ADMIN;
* Poga, kuru spiežot lietotājs nonāk SHARES;
* Poga, kuru spiežot lietotājs nonāk REGISTRATION;
* Poga, kuru spiežot lietotājs nonāk HISTORY;
* Poga, kuru spiežot lietotājs nonāk MEMBERS;
* Poga, kuru spiežot lietotājs var iziet no konta;



* Darbinieks var redzēt, ko, kad, cik daudz un kas ir saņēmis preci;
* Poga, kuru spiežot darbinieks nonāk atpakaļ ADMIN;



* Tekstlauks, kurā administrators ievada ražas dalījuma nosaukumu;
* Tekstlauks, kurā administrators ievada, kāda veida preces šajā dalījumā tiek iekļautas;
* Skaitlisks ievadlauks, kurā norāda daudzumu;
* Poga, kuru spiežot darbinieks pievieno jauno ražas dalījumu ierakstu datubāzē;
* Tabula ar kolonnām: SHARE NAME, CONTENT, AMOUNT;
* Katram ierakstam blakus atrodas REMOVE poga, kas ļauj dzēst konkrēto dalījuma tipu no sistēmas;
* Poga, kuru spiežot darbinieku atgriež uz ADMIN;



* Izvēlnes lauks vai meklēšanas lauks, kurā darbinieks izvēlas klientu, kuram piegāde reģistrēta;
* Ievadāms preces skaitlis un datums;
* Brīvs tekstlauks produkta nosaukumam, un piegādes datumam;
* Poga, kuru spiežot tiek saglabāta ievadītie dati sistēmā, saistot tos ar izvelēto klientu;
* Poga, kuru spiežot darbinieku atgriež uz ADMIN;



* Darbinieks var redzēt MEMBER;
* Tekstlauks, kurā administrators ievada jaunu MEMBER vārdu;
* Izvēlne, kurā administrators izvēlas SHARE priekš MEMBER;
* Poga, kuru spiežot darbinieks var redzēt MEMBER STATUS;
* Poga, kuru spiežot administrators var pievienot jaunu MEMBER;
* Poga, kuru spiežot darbinieks tiek atgriezts ADMIN;



* Darbinieks var redzēt MEMBER STATUS;
* Darbinieks var redzēt MEMBER vārdu;
* Darbinieks var redzēt MEMBER ražas daļas tipu;
* Darbinieks var redzēt, vai MEMBER ir ņēmušu papildus olas vistas vietā;
* Darbinieks var redzēt, cik daudz preces MEMBER ir bijušas piegādātas vai paši saņēmuši;
* Darbinieks var redzēt, cik daudz preces MEMBER ir palikušas no tā gada ražas;
* Darbinieks var redzēt, vai MEMBER ir palikušas olas vai gaļas kastes no iepriekšēja gada ražas;
* Poga, kuru spiežot darbinieks var mainīt MEMBER ražas daļas tipu;
* Katrai precei atrodas blakus poga EDIT, kas ļauj darbiniekam mainīt MEMBER daudzumu katrā no šīm precēm;
* Poga, kuru spiežot darbinieks nonāk pie MEMBER HISTORY;
* Poga, kuru spiežot darbinieks atgriežas uz MEMBERS.

## Izvēlētā izstrādes vide un valoda:

Programma tiks izstrādāta ar Python programmēšanas valodu, izmantojot Flask ietvaru (framework). Šī izvēle ir balstīta uz vairākiem faktoriem:

* Python ir viegli uzturams un piemērots straujai izstrādei;
* Flask ļauj ātri izveidot tīmekļa lietotni ar autentifikāciju un datu apstrādi;
* Sistēma būs viegla un ērti lietojama vairākām dažādām ierīcēm – datoriem, planšetēm, telefoniem u.c.;
* Tiks izmantota arī SQL datubāze, kas ir piemērota mazākām sistēmām ar vienkāršu struktūru;
* Flask lietotnes viegli papildināt ar citām programmēšanas valodām, kā HTML, CSS un JAVASCRIPT, lai uzlabotu lietotājām dizainu un funkcionalitāti;
* Tā ir man viss ērtāk lietojamā valoda, tas ļauj drošāk un kvalitatīvāk izstrādāt galaproduktu.

# Programmatūras izstrādes plāns

## Izstrādes modelis:

Lai izstrādātu programmu saimniecībai “Virgenes”, es apsvēru divus no izplatītākajiem programmatūras izstrādes modeļiem: ūdenstece (Waterfall) un iteratīvais/agilais modelis.

Ūdensteces modelis, tiek darīts soli pa solim. Tam ir vairāki posmi, kur nākamo tikai sāk, kad iepriekšējais ir pilnīgi izstrādāts. Šis modelis derētu, ja prasības būtu pilnībā zināmas un nemainīgas jau sākumā. Iteratīvais/agilais modelis ļauj sistēmu veidot pa posmiem, pielāgojoties mainīgām prasībām un pamazām uzlabojot rezultātu. Tas ļauj testēt un uzlabot katru funkcionalitātes daļu jau izstrādes laikā.

Šim projektam tika izvēlēta iteratīvais modelis, jo:

* Programma tiek veidota individuāli vienam klientam;
* Daļa funkciju tika precizēta izstrādes gaitā;
* Ir svarīgi ātri pārbaudīt un pielabot sistēmas funkcionalitāti saskaņā ar lietotāja vajadzībām.

## Izstrādes soļi:

Programmas izstrāde tika plānota šādos, vienkāršotos, posmos:

1. Prasību apkopošana – tika noskaidrotas saimniecības vajadzība un lietotāju vēlmes;
2. Funkcionalitātes plānošana – tika izveidots programmai vajadzīgo funkciju saraksts un sadalījums pa moduļiem;
3. Datu modeļa un lietotāja saskarsmes skiču izveide – izveidotas vienkāršas diagrammas un skices, kas attēlo, kā programmai vajadzētu, pamatā, izskatīties;
4. Programmas kodēšana pa daļām – sākot ar pamatfunkcijām, piemēram, reģistrāciju, datu ievadi un pārskatu apskati;
5. Testēšana un labojumi – katra daļa tika pārbaudīta uzreiz pēc izveides, pielabota, ja bija nepieciešams;
6. Lietotāja ceļveža izveide un nodošana lietošanā – pēc programmas izstrādes tika sagatavota īsa pamācība.

# Atkļūdošanas un akcepttestēšanas pārskats

## Atkļūdošanas pārskats:

Lai identificētu un novērstu programmatūras kļūdas, tika veikta vairāku līmeņu testēšana. Tika izmantota vienību testēšana, kur katra programmas funkcija tika pārbaudīta, lai nodrošinātu, ka katrs komponents darbojas, integrācijas testēšana, kur tika pārbaudīts, vai atsevišķas vienības var savstarpēji sadarboties, un manuālā testēšana, kur sākot programmu pārbaudīja vai, kaut kas nestrādā.

Programmas taisīšanas laikā, tika atrastas vairākas kļūdas ar šīm testēšanas metodēm. Piemēram, lietojot vienību testēšanu, es atradu vairākas reizes vienkāršas sintakses kļūdas kodā. Lietojot integrācijas testēšanu, es atradu, ka administratīvajiem skatiem, kā MEMBERS un STATUS (admin), bija liela kļūda, kas neļāva atvērt STATUS (admin). Lietojot manuālo testēšanu, es atradu, ka REGISTER (admin) ievadītie dati netiek saglabāti/pārsūtīti uz datubāzes failu, lai tos varētu lietot citur programmā.

## Akcepttestēšana pārskats:

Akcepttestēšana tika veikta ar nejauši izzvetētiem cilvēkiem, lai pārliecinātos, ka sistēma pilnībā atbilst sākotnēji formulētajām prasībām. Testēšanas posmā tika izmantoti šādi akcepttestēšanas scenāriji:

* Lietotāja pieteikšanās un autentifikācija: Pārbaudīja vai programmā var piekļūt pareizi kontam, kļūdu ziņojumi nepareizu datu ievades gadījumā un navigācija uz lietotāja galvenajiem paneļiem;
* Datu ievade un apstrāde: Tika testēta datu reģistrācija procesa funkcionalitāte gan klienta kontam, gan administratīvajam, pareiza informācijas ievade, administratīvajam kontam mainīt datu specifiku, datubāzes atjaunināšana un atbilstošu ziņojumu ģenerēšana lietotāja saskarnē.

Secinājums pēc akcepttestēšanas bija, ka sistēma lielākoties atbilst izvirzītajām prasībām, taču tika identificēti daži punkti, kuros nepieciešami turpmāki uzlabojumi, lai maksimāli atvieglotu galalietotāju darbu.

## Secinājumi:

Pēc atkļūdošanas un akcepttestēšanas posma veikšanas var secināt, ka sistēmas galvenās funkcionalitātes ir lielākoties stabilas un atbilst sākotnēji izvirzītajām prasībām, bet ir problēmas ar vairākām administratīvajām funkcijām, kuras vajadzētu uzlabot.

# Lietotāja ceļvedis

Sistēma “Virgenes Andelsgård Share Tracker” ir izstrādāta, lai automatizētu ikdienas darbības, saistītas ar preču izsekojamību un piegādes procesu pārvaldību. Sistēma nodrošina divas galvenās piekļuves iespējas:

* Klientu konts: kurā lietotāji var reģistrēt saņemto informāciju, pārbaudīt savu konta statusu un vēstures datus.
* Administratīvā konta panelis: paredzēts darbiniekiem, kuri pārrauga datu ievadi, atjaunināšanu un veic specifiskus administratīvos uzdevumus, piemēram, jaunu dalījumu vai locekļu pievienošanu.

## Instalācija:

### Sistēmas prasības:

1. Sistēma darbojas uz servera ar Flask lietotni;
2. Ierīcei ir jābūt pietiekamai aprīkojumā, lai nodrošinātu atbilstošu veiktspēju.

### Instalācijas instrukcijas:

Lietotājam jāuzstāda Flask vide, jāiegūst atbilstošie Python moduļi (piemēram, argon2, sqlite3) un jānodrošina pareizs vides mainīgo FLASK\_SECRET iestatījums. Sistēma tiek uzsākta, palaižot skriptu, kas aktivizē serveri.

## Lietotāja reģistrācija, pieteikšanās un konta pārvalde:

### Reģistrācijas process:

Atšķirībā no tradicionālās reģistrācijas, kurā lietotāji var pašreiz veidot savus kontus, sistēmā klientu reģistrāciju veic tikai administrators. Administrators ir atbildīgs par aunu klienta kontu izveidi. Šajā procesā administrators pievieno jauno klientu ar nepieciešamo informāciju (piemēram, klienta vārdu, ražas daļas izvēli) atbilstoši noteiktiem datu standarta kritērijiem.

### Pieteikšanās procesa apraksts:

Pēc tam, kad administrators ir izveidojis klienta kontu, klientam var ievadīt savu lietotājvārdu un paroli, kods, tad pārbauda ievadītos datus pret datubāzē saglabātajiem ierakstiem. Atbilstošs sesijas ievads tiek saglabāts, un lietotājs tiek pāradresēts uz savu skatu atkarībā no lomas.

### Konta pārvaldība:

Klientam ir pieejams savs lietotāja konts, kurā viņš var redzēt un uzraudzīt savu aktivitāti. Klient var mainīt savu paroli, bet jauno kontu izveide un izmaiņas klientu reģistrācijā notiek tikai caur administratīvo paneli. Turklāt, administrācija var atjaunināt klienta ražas daļas izvēli un apskatīt detalizētu klienta statusa informāciju.

## Galveno funkciju apraksts un navigācija:

### Lietotāja galvenie skati un navigācija:

Klientiem:

* Pēc veiksmīgas pieteikšanās tiek attēlots STATUS skats, kur redzami informācija par klienta pasūtītajām dalībām, atlikumiem un vēsturiskie darījumi;
* Klienti var reģistrēt jaunas transakcijas un apskatīt darījumu vēsturi.

Administratoriem:

* Administrators piekļūst galvenajam administratīvajam panelim ADMIN, kur var pārvaldīt ražas dalījumu tipus, reģistrēt piegādes, rediģēt piegādes un apskatīt klientu vēsturi.
* Administrators izmanto galapunktu (endpoint) MEMBERS, lai reģistrētu jaunus klientus, un var pārvaldīt klientu lietotāja datus.

### Interfeisa elementi un mijiedarbība:

Lietotāja saskarne ir veidota ar atsevišķiem HTM šabloniem, kas nodrošina intuitīvu navigāciju un skaidri marķētus pogas vai saites – piemēram, “LOGIN”, “LOGOUT”, “CHANGE PASSWORD” un citi. Tādi elementi sniedz tūlītēju atgriezenisko saiti, izmantojot flask ziņojumus, kas norāda par veiksmīgu operāciju, kļūdām vai nepieciešamām darbībām.

## Administratīvās funkcijas un papildus iespējas:

### Administratora panelis un funkcionalitātes:

* Administrators var pievienot, dzēst un rediģēt dalījumu tipus caur galapunktu SHARES;
* Pēdējie darījumi un klientu aktivitātes tiek apskatītas caur HISTORY (admin) un STATUS (admin);
* Jaunu klientu reģistrācija tiek veikta tikai caur MEMBERS, atspoguļojot sistēmas politiku, kurā klientiem pašiem nav piekļuves reģistrācijas lapai.

### Tehniskā uzturēšana un rezerves kopiju veidošana:

Sistēmas kodā ir iekļauta datubāzes tabulu izveide, kas nodrošina, ka nepieciešamās struktūras tiek automātiski izveidotas. Administrators regulāri var pārbaudīt, vai visi datus saistītie ieraksti ir pareizi sinhronizēti, un veikti nepieciešamos labojumus vai atjauninājumu.

# Piemērotākā licence

Licence, ko es izvēlējos priekš šī projekta, ir Public Domain (sabiedriskā īpašuma statuss). Es šo licenci izvēlējos, jo gan man, gan klientam nav velmē noturēt autoriskos ierobežojumus vai prasīt juridiskas atļaujas katram koda izmantojumam.

Tā rezultātā:

* Darbs ir pieejams bez jebkādiem ierobežojumiem – ikviens var brīvi izmantot, modificēt un uzlabojumus, nepārkāpjot noteiktas licences prasības;
* Projektā netiek ierobežota inovāciju attīstība, jo ikviens interesents var iekļaut izmaiņas un uzlabojumus, nepārkāpjot noteiktas licences prasības;
* Public Domain statuss veicina atklātu sadarbību un izglītību, ļaujot citiem izstrādātājiem un lietotājiem pilnībā iepazīt sistēmas darbību un integrēt to citos projektos bez administratīvām barjērām.

# Programmatūras kods

## main.py :

import os

from flask import Flask, render\_template, request, redirect, url\_for, session, flash

import sqlite3

import datetime

from argon2 import PasswordHasher, exceptions

app = Flask(\_\_name\_\_)

app.secret\_key = os.environ.get("FLASK\_SECRET", "default\_secret\_key")

ph = PasswordHasher()

hashed\_demo\_password = ph.hash("your\_password")

def get\_db\_connection():

conn = sqlite3.connect("flask.db")

conn.row\_factory = sqlite3.Row

conn.execute("PRAGMA foreign\_keys = ON")

conn.execute("PRAGMA journal\_mode = WAL;")

conn.execute("PRAGMA synchronous = NORMAL;")

return conn

def insert\_sql(cmd, vals=None):

with get\_db\_connection() as conn:

c = conn.cursor()

if vals:

c.execute(cmd, vals)

else:

c.execute(cmd)

conn.commit()

def get\_sql(cmd, vals=None):

with get\_db\_connection() as conn:

c = conn.cursor()

if vals:

c.execute(cmd, vals)

else:

c.execute(cmd)

res = c.fetchall()

return res

@app.route('/')

def index():

insert\_sql(

"CREATE TABLE IF NOT EXISTS users (\

id INTEGER PRIMARY KEY AUTOINCREMENT,\

username TEXT UNIQUE NOT NULL,\

password TEXT NOT NULL,\

role TEXT CHECK(role IN ('client', 'admin')) NOT NULL,\

name TEXT,\

status TEXT,\

share\_type\_id INTEGER,\

FOREIGN KEY(share\_type\_id) REFERENCES share\_types(id)\

);"

)

insert\_sql(

"CREATE TABLE IF NOT EXISTS transactions (\

id INTEGER PRIMARY KEY AUTOINCREMENT,\

user\_id INTEGER NOT NULL,\

transaction\_date DATE NOT NULL,\

extra\_option BOOLEAN,\

FOREIGN KEY(user\_id) REFERENCES users(id)\

);"

)

insert\_sql(

"CREATE TABLE IF NOT EXISTS share\_types (\

id INTEGER PRIMARY KEY AUTOINCREMENT,\

share\_name TEXT NOT NULL,\

content TEXT NOT NULL,\

amount INTEGER NOT NULL\

);"

)

insert\_sql(

"CREATE TABLE IF NOT EXISTS transaction\_shares (\

id INTEGER PRIMARY KEY AUTOINCREMENT,\

transaction\_id INTEGER NOT NULL,\

share\_type\_id INTEGER NOT NULL,\

quantity INTEGER NOT NULL,\

FOREIGN KEY(transaction\_id) REFERENCES transactions(id),\

FOREIGN KEY(share\_type\_id) REFERENCES share\_types(id)\

);"

)

insert\_sql(

"CREATE TABLE IF NOT EXISTS history (\

id INTEGER PRIMARY KEY AUTOINCREMENT,\

user\_id INTEGER NOT NULL,\

transaction\_id INTEGER,\

change\_date DATE NOT NULL,\

FOREIGN KEY(user\_id) REFERENCES users(id)\

);"

)

insert\_sql(

"CREATE TABLE IF NOT EXISTS deliveries (\

id INTEGER PRIMARY KEY AUTOINCREMENT,\

user\_id INTEGER NOT NULL,\

product\_name TEXT NOT NULL,\

delivery\_date DATE NOT NULL,\

quantity INTEGER NOT NULL,\

FOREIGN KEY(user\_id) REFERENCES users(id)\

);"

)

return render\_template("index.html")

@app.route("/login", methods=["GET", "POST"])

def login():

if request.method == "POST":

username = request.form.get("username", "")

password = request.form.get("password", "")

if not username or not password:

flash("Please, enter username and password!", "danger")

return redirect(url\_for("login"))

user = get\_sql(

"SELECT id, username, password, role, name, status\

FROM users WHERE username = ?", (username,)

)

if user:

user = user[0]

hashed\_password = user[2]

try:

ph.verify(hashed\_password, password)

session["user\_id"] = user[0]

session["username"] = user[1]

session["role"] = user[3]

flash("Login successful!", "success")

if user[3] == "client":

return redirect(url\_for("statusm"))

elif user[3] == "admin":

return redirect(url\_for("admin"))

except exceptions.VerifyMismatchError:

flash("Wrong username or password!", "error")

return redirect(url\_for("index"))

return render\_template("login.html")

@app.route("/statusm")

def statusm():

if "user\_id" not in session:

flash("Please, login!", "warning")

return redirect(url\_for("login"))

user\_id = session["user\_id"]

user\_data = get\_sql(

"SELECT id, username, name, status\

FROM users WHERE id = ?", (user\_id,)

)

if not user\_data:

flash("User not found!", "danger")

return redirect(url\_for("logout"))

user = user\_data[0]

share\_data = get\_sql(

"SELECT st.share\_name, st.amount as yearly\_amount,\

COALESCE(SUM(ts.quantity), 0) as total\_quantity\

FROM transactions t\

JOIN transaction\_shares ts ON t.id = ts.transaction\_id\

JOIN share\_types st ON ts.share\_type\_id = st.id \

WHERE t.user\_id = ? GROUP BY st.id;", (user\_id,)

)

updated\_shares = []

purchased\_shares\_names = []

for row in share\_data:

share\_name, yearly\_amount, total\_quantity = row

leftover = yearly\_amount - total\_quantity if yearly\_amount > total\_quantity else 0

updated\_shares.append({

"share\_name": share\_name,

"yearly\_amount": yearly\_amount,

"total\_quantity": total\_quantity,

"leftover": leftover

})

if total\_quantity > 0:

purchased\_shares\_names.append(share\_name)

purchased\_shares\_names = list(set(purchased\_shares\_names))

return render\_template("statusm.html", user=user, share\_info=updated\_shares, purchased\_shares\_names=purchased\_shares\_names)

@app.route("/admin")

def admin():

return render\_template("admin.html")

@app.route("/shares", methods=["GET", "POST"])

def shares():

if "user\_id" not in session:

flash("Please, login!", "warning")

return redirect(url\_for("login"))

if session.get("role") != "admin":

flash("Access denied!", "danger")

return redirect(url\_for("index"))

if request.method == "POST":

if "add\_share" in request.form:

share\_name = request.form.get("share\_name", "").strip()

share\_content = request.form.get("share\_content", "").strip()

content\_amount\_str = request.form.get("content\_amount", "0").strip()

try:

content\_amount = int(content\_amount\_str)

except ValueError:

content\_amount = 0

if share\_name and share\_content and content\_amount > 0:

insert\_sql(

"INSERT INTO share\_types (share\_name, content, amount)\

VALUES (?, ?, ?)", (share\_name, share\_content, content\_amount)

)

flash("New share added!", "success")

else:

flash("Please, fill in all fields correctly!", "warning")

elif "remove\_id" in request.form:

remove\_id = request.form.get("remove\_id")

if remove\_id:

insert\_sql(

"DELETE FROM share\_types WHERE id = ?", (remove\_id,)

)

flash("Share type removed!", "success")

else:

flash("Invalid share selected for removal!", "warning")

return redirect(url\_for("shares"))

else:

share\_list = get\_sql(

"SELECT id, share\_name, content, amount FROM share\_types"

)

return render\_template("shares.html", share\_list=share\_list)

@app.route("/registrationa", methods=["GET", "POST"])

def registrationa():

if "user\_id" not in session:

flash("Please, login!", "warning")

return redirect(url\_for("login"))

if session.get("role") != "admin":

flash("Access denied!", "danger")

return redirect(url\_for("index"))

all\_clients = get\_sql(

"SELECT id, name FROM users WHERE role='client'"

)

if request.method == "POST":

user\_id\_str = request.form.get("user\_id", "")

product\_name = request.form.get("product\_name", "").strip()

quantity\_str = request.form.get("quantity", "")

delivery\_date = request.form.get("delivery\_date", "")

previous\_input = {

"user\_id": user\_id\_str,

"product\_name": product\_name,

"quantity": quantity\_str,

"delivery\_date": delivery\_date

}

if not (user\_id\_str and product\_name and quantity\_str and delivery\_date):

flash("Please, fill all fields!", "warning")

return render\_template("registrationa.html", all\_clients=all\_clients, previous\_input=previous\_input)

try:

user\_id = int(user\_id\_str)

quantity = int(quantity\_str)

except ValueError:

flash("Please, enter valid numeric values!", "danger")

return render\_template("registrationa.html", all\_clients=all\_clients, previous\_input=previous\_input)

insert\_sql(

"INSERT INTO deliveries (user\_id, product\_name, quantity, delivery\_date)\

VALUES (?, ?, ?, ?)", (user\_id, product\_name, quantity, delivery\_date)

)

flash("Delivery registered successfully!", "success")

return redirect(url\_for("registrationa"))

else:

return render\_template("registrationa.html", all\_clients=all\_clients, previous\_input=None)

@app.route("/historya")

def historya():

if "user\_id" not in session:

flash("Please, login!", "warning")

return redirect(url\_for("login"))

if session.get("role") != "admin":

flash("Access denied!", "danger")

return redirect(url\_for("index"))

history\_data = get\_sql(

"SELECT t.transaction\_date, st.share\_name, ts.quantity, u.name\

FROM transactions t\

JOIN transaction\_shares ts ON t.id = ts.transaction\_id\

JOIN share\_types st ON ts.share\_type\_id = st.id\

JOIN users u ON t.user\_id = u.id\

ORDER BY t.transaction\_date DESC"

)

return render\_template("historya.html", history\_data=history\_data)

@app.route("/members", methods=["GET", "POST"])

def members():

if "user\_id" not in session:

flash("Please, login!", "warning")

return redirect(url\_for("login"))

if session.get("role") != "admin":

flash("Access denied!", "danger")

return redirect(url\_for("index"))

if request.method == "POST":

new\_member\_name = request.form.get("new\_member\_name", "").strip()

share\_type\_id\_str = request.form.get("share\_type\_id", "")

if not new\_member\_name:

flash("Please, enter member name!", "warning")

return redirect(url\_for("members"))

try:

share\_type\_id = int(share\_type\_id\_str)

except ValueError:

share\_type\_id = None

default\_username = new\_member\_name.lower().replace(" ", "\_")

default\_password = "password123"

hashed\_pw = ph.hash(default\_password)

insert\_sql(

"INSERT INTO users (username, password, role, name, status, share\_type\_id)\

VALUES (?, ?, ?, ?, ?, ?)",

(default\_username, hashed\_pw, "client", new\_member\_name, "", share\_type\_id)

)

flash(f"New member '{new\_member\_name}' added successfully!", "success")

return redirect(url\_for("members"))

else:

share\_types = get\_sql("SELECT id, share\_name FROM share\_types")

members\_list = get\_sql(

"SELECT u.id, u.name, st.share\_name\

FROM users u \

LEFT JOIN share\_types st ON u.share\_type\_id = st.id\

WHERE u.role='client'"

)

return render\_template("members.html", share\_types=share\_types, members\_list=members\_list)

@app.route("/logout")

def logout():

session.clear()

flash("You have been logged out!", "success")

return redirect(url\_for("index"))

@app.route("/historym")

def historym():

if "user\_id" not in session:

flash("Please, login!", "warning")

return redirect(url\_for("login"))

user\_id = session["user\_id"]

history\_data = get\_sql(

"SELECT transaction\_date, st.share\_name, ts.quantity\

FROM transactions t\

JOIN transaction\_shares ts ON t.id = ts.transaction\_id\

JOIN share\_types st ON ts.share\_type\_id = st.id\

WHERE t.user\_id = ?\

ORDER BY t.transaction\_date DESC", (user\_id,)

)

return render\_template("historym.html", history\_data=history\_data)

@app.route("/registrationm", methods=["GET", "POST"])

def registrationm():

if "user\_id" not in session:

flash("Please, login!", "warning")

return redirect(url\_for("login"))

user\_id = session["user\_id"]

if request.method == "POST":

transaction\_date = request.form.get("transaction\_date", "")

extra\_option\_str = request.form.get("extra\_option", "")

extra\_option = True if extra\_option\_str == "YES" else False

conn = sqlite3.connect("flask.db")

conn.execute("PRAGMA foreign\_keys = ON")

c = conn.cursor()

c.execute(

"INSERT INTO transactions (user\_id, transaction\_date, extra\_option)\

VALUES (?, ?, ?)", (user\_id, transaction\_date, extra\_option)

)

transaction\_id = c.lastrowid

conn.commit()

conn.close()

insert\_sql(

"INSERT INTO history (user\_id, transaction\_id, change\_date)\

VALUES (?, ?, date('now'))", (user\_id, transaction\_id)

)

share\_types = get\_sql(

"SELECT id, share\_name, content, amount FROM share\_types"

)

for share in share\_types:

share\_id = share[0]

quantity\_str = request.form.get(f"share\_{share\_id}", "0")

try:

quantity = int(quantity\_str)

except ValueError:

quantity = 0

if quantity > 0:

insert\_sql(

"INSERT INTO transaction\_shares \

(transaction\_id, share\_type\_id, quantity) \

VALUES (?, ?, ?)", (transaction\_id, share\_id, quantity)

)

flash("Data added successfully!", "success")

return redirect(url\_for("statusm"))

else:

share\_types = get\_sql(

"SELECT id, share\_name, content, amount FROM share\_types"

)

return render\_template("registrationm.html", share\_types=share\_types)

@app.route("/edit\_delivery/<int:delivery\_id>", methods=["GET", "POST"])

def edit\_delivery(delivery\_id):

if "user\_id" not in session:

flash("Please, login!", "warning")

return redirect(url\_for("login"))

if session.get("role") != "admin":

flash("Access denied!", "danger")

return redirect(url\_for("index"))

if request.method == "POST":

product\_name = request.form.get("product\_name", "").strip()

delivery\_date = request.form.get("delivery\_date", "").strip()

quantity\_str = request.form.get("quantity", "").strip()

if not product\_name or not delivery\_date or not quantity\_str:

flash("Please, fill all the required fields!", "warning")

return redirect(url\_for("edit\_delivery", delivery\_id=delivery\_id))

try:

quantity = int(quantity\_str)

except ValueError:

flash("Invalid quantity provided!", "danger")

return redirect(url\_for("edit\_delivery", delivery\_id=delivery\_id))

insert\_sql(

"UPDATE deliveries SET product\_name = ?,\

quantity = ?, delivery\_date = ?\

WHERE id = ?", (product\_name, quantity, delivery\_date, delivery\_id)

)

flash("Delivery information updated successfully!", "success")

return redirect(url\_for("registrationa"))

else:

delivery = get\_sql(

"SELECT id, user\_id, product\_name, quantity, delivery\_date\

FROM deliveries WHERE id = ?", (delivery\_id,)

)

if not delivery:

flash("Delivery not found!", "warning")

return redirect(url\_for("registrationa"))

delivery = delivery[0]

return render\_template("edit\_delivery.html", delivery=delivery)

@app.route("/change\_password", methods=["GET", "POST"])

def change\_password():

if "user\_id" not in session:

flash("Please, login!", "warning")

return redirect(url\_for("login"))

if request.method == "POST":

new\_password = request.form.get("new\_password", "")

confirm\_password = request.form.get("confirm\_password", "")

if not new\_password or not confirm\_password:

flash("Please, fill in all fields!", "warning")

return redirect(url\_for("change\_password"))

if new\_password != confirm\_password:

flash("The confirmation doesn't match the password!", "warning")

return redirect(url\_for("change\_password"))

new\_hashed = ph.hash(new\_password)

user\_id = session["user\_id"]

insert\_sql(

"UPDATE users SET password = ?\

WHERE id = ?", (new\_hashed, user\_id)

)

flash("Password changed successfully!", "success")

return redirect(url\_for("statusm"))

return render\_template("change\_password.html")

@app.route("/user\_share\_type/<int:user\_id>", methods=["POST"])

def user\_share\_type(user\_id):

new\_share\_type = request.form.get("new\_share\_type")

if new\_share\_type:

insert\_sql(

"UPDATE users SET share\_type\_id = ?\

WHERE id = ?", (new\_share\_type, user\_id)

)

flash("Share type updated!", "success")

insert\_sql(

"INSERT INTO history (user\_id, change\_date)\

VALUES (?, date('now'))", (user\_id,)

)

else:

flash("Invalid share type", "warning")

return redirect(url\_for("statusa", user\_id=user\_id))

@app.route("/statusa/<int:user\_id>")

def statusa(user\_id):

if "user\_id" not in session or session["role"] != "admin":

flash("Access denied!", "danger")

return redirect(url\_for("login"))

user\_info = get\_sql(

"SELECT u.id, u.name, u.share\_type\_id, u.status,\

(SELECT t.extra\_option FROM transactions t\

WHERE t.user\_id = u.id\

ORDER BY t.transaction\_date DESC LIMIT 1) as extra\_option,\

st.share\_name, st.amount as yearly\_amount\

FROM users u\

LEFT JOIN share\_types st ON u.share\_type\_id = st.id\

WHERE u.id = ?", (user\_id,)

)

if not user\_info:

flash("Member not found!", "warning")

return redirect(url\_for("members"))

user\_info = user\_info[0]

if user\_info["share\_name"] is None:

user\_info["share\_name"] = "N/A"

if user\_info["status"] is None:

user\_info["status"] = ""

current\_year = str(datetime.date.today().year)

previous\_year = str(datetime.date.today().year - 1)

current\_picked = get\_sql(

"SELECT COALESCE(SUM(ts.quantity), 0)\

FROM transactions t\

JOIN transaction\_shares ts ON t.id = ts.transaction\_id\

WHERE t.user\_id = ? AND strftime('%Y', t.transaction\_date) = ?", (user\_id, current\_year)

)

current\_delivered = get\_sql(

"SELECT COALESCE(SUM(quantity), 0)\

FROM deliveries WHERE user\_id = ?\

AND strftime('%Y', delivery\_date) = ?", (user\_id, current\_year)

)

delivered\_total = current\_picked[0][0] + current\_delivered[0][0]

yearly\_amount = user\_info["yearly\_amount"] if user\_info["yearly\_amount"] else 0

remaining\_units = yearly\_amount - delivered\_total if yearly\_amount > delivered\_total else 0

prev\_picked = get\_sql(

"SELECT COALESCE(SUM(ts.quantity), 0)\

FROM transactions t\

JOIN transaction\_shares ts ON t.id = ts.transaction\_id\

WHERE t.user\_id = ? AND strftime('%Y', t.transaction\_date) = ?", (user\_id, previous\_year)

)

prev\_delivered = get\_sql(

"SELECT COALESCE(SUM(quantity), 0)\

FROM deliveries WHERE user\_id = ?\

AND strftime('%Y', delivery\_date) = ?", (user\_id, previous\_year)

)

previous\_total = prev\_picked[0][0] + prev\_delivered[0][0]

previous\_leftover = yearly\_amount - previous\_total if yearly\_amount > previous\_total else 0

deliveries = get\_sql(

"SELECT id, product\_name, quantity, delivery\_date\

FROM deliveries WHERE user\_id = ?", (user\_id,)

)

return render\_template("statusa.html", user=user\_info, delivered\_total=delivered\_total, remaining\_units=remaining\_units, previous\_leftover=previous\_leftover, deliveries=deliveries)

@app.route("/member\_history/<int:user\_id>")

def member\_history(user\_id):

if "user\_id" not in session or session.get("role") != "admin":

flash("Access denied!", "danger")

return redirect(url\_for("login"))

history\_data = get\_sql(

"SELECT transaction\_date, st.share\_name, ts.quantity\

FROM transactions t\

JOIN transaction\_shares ts ON t.id = ts.transaction\_id\

JOIN share\_types st ON ts.share\_type\_id = st.id\

WHERE t.user\_id = ?\

ORDER BY t.transaction\_date DESC", (user\_id,)

)

return render\_template("member\_history.html", history\_data=history\_data)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0', port=int(os.environ.get("PORT", 5000)), debug=False)

## admin.html:

{% extends "layout.html" %}

{% block body %}

<div class="admin-container">

<h2>ADMIN</h2>

<a href="/shares">

<button type="button" name="shares\_button">

SHARES

</button>

</a>

<a href="/registrationa">

<button type="button" name="admin\_registration\_button">

REGISTRATION

</button>

</a>

<a href="/historya">

<button type="button" name="admin\_history\_button">

HISTORY

</button>

</a>

<a href="/members">

<button type="button" name="members\_button">

MEMBERS

</button>

</a>

<a href="/logout">

<button type="button" name="log\_off\_button">

LOG OFF

</button>

</a>

</div>

{% endblock %}

## **change\_password.html:**

{% extends "layout.html" %}

{% block body %}

<div class="change\_password-container">

<h2>Change Password</h2>

<form method="POST" action="{{url\_for('change\_password')}}">

<div>

<label for="new\_password">New Password:</label>

<input type="password" id="new\_password" name="new\_password" required>

</div>

<div>

<label for="confirm\_password">Confirm Password:</label>

<input type="password" id="confirm\_password" name="confirm\_password" required>

</div>

<div>

<button type="submit">CHANGE PASSWORD</button>

</div>

</form>

<a href="{{url\_for('statusm')}}">

<button type="button">BACK</button>

</a>

</div>

{% endblock %}

## edit\_delivery.html:

{% extends "layout.html" %}

{% block body %}

<div class="edit\_delivery-container">

<h2>Edit Delivery</h2>

<form method="POST" action="{{url\_for('edit\_delivery', delivery\_id=delivery\_id)}}">

<div>

<label for="product\_name">Product Name:</label>

<input type="text" id="product\_name" name="product\_name" value="{{delivery.product\_name}}" required>

</div>

<div>

<label for="quantity">Quantity:</label>

<input type="number" id="quantity" name="quantity" value="{{delivery.quantity}}" min="1" required>

</div>

<div>

<label for="delivery\_date">Delivery Date:</label>

<input type="date" id="delivery\_date" name="delivery\_date" value="{{delivery.delivery\_date}}" required>

</div>

<div>

<button type="submit">SAVE CHANGES</button>

</div>

</form>

<a href="{{url\_for('statusa', user\_id=delivery.user\_id)}}">

<button type="button">BACK</button>

</a>

</div>

{% endblock %}

## historya.html:

{% extends "layout.html" %}

{% block body %}

<h2>HISTORY</h2>

<div class="admin\_history-container">

<table border="1">

<thead>

<tr>

<th>Member</th>

<th>Product</th>

<th>Quantity</th>

<th>Date</th>

</tr>

</thead>

<tbody>

{% for row in history\_data %}

<tr>

<td>{{ row["name"] }}</td>

<td>{{ row["share\_name"] }}</td>

<td>{{ row["quantity"] }}</td>

<td>{{ row["transaction\_date"] }}</td>

</tr>

{% endfor %}

</tbody>

</table>

<div style="margin-top: 20px;">

<a href="{{url\_for('admin')}}">

<button type="button">BACK</button>

</a>

</div>

</div>

{% endblock %}

## historym.html:

{% extends "layout.html" %}

{% block body %}

<h2>HISTORY</h2>

<div class="member\_history-container">

<div style="float:left; margin-right: 60px;">

<h3>Eggs picked up:</h3>

<table border="1">

<tr>

<th>Quantity</th>

<th>Date</th>

</tr>

{% for transaction\_date, share\_name, quantity in history\_data %}

{% if "egg" in share\_name|lower %}

<tr>

<td>{{quantity}}</td>

<td>{{transaction\_date}}</td>

</tr>

{% endif %}

{% endfor %}

</table>

</div>

<div style="float:left;">

<h3>Meat picked up:</h3>

<table border="1">

<tr>

<th>What</th>

<th>Quantity</th>

<th>Date</th>

</tr>

{% for transaction\_date, share\_name, quantity in history\_data %}

{% if "meat" in share\_name|lower

or "beitekylling" in share\_name|lower

or "okselasse" in share\_name|lower %}

<tr>

<td>{{share\_name}}</td>

<td>{{quantity}}</td>

<td>{{transaction\_date}}</td>

</tr>

{% endif %}

{% endfor %}

</table>

</div>

<div style="clear:both;"></div>

<div style="margin-top: 20px;">

<a href="{{url\_for('statusm')}}">

<button type="button">BACK</button>

</a>

</div>

</div>

{% endblock %}

## **index.html:**

{% extends "layout.html" %}

{% block body %}

<div class="index-container">

<h1>Virgenes Andelsgård Share Tracker</h1>

<a href="/login">

<button type="button" name="login\_screen\_button">

LOGIN

</button>

</a>

</div>

{% endblock %}

## layout.html:

<!doctype html>

<html lang="en-GB, nor">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>Virgenes Andelsgård Share Tracker</title>

<link rel="icon" type="image/x-icon" href="https://www.virgenes.no/favicon.ico">

</head>

<body>

{% block body %}{% endblock %}

</body>

</html>

## login.htm:

{% extends "layout.html" %}

{% block body %}

<div class="login-container">

<h2>LOGIN</h2>

<form method="POST" action="{{url\_for('login')}}">

<div>

<label for="username">Username:</label>

<input type="text" id="username" name="username" required>

</div>

<div>

<label for="password">Password:</label>

<input type="password" id="password" name="password" required>

</div>

<div>

<button type="submit">LOGIN</button>

</div>

</form>

</div>

{% endblock %}

## member\_history.html:

{% extends "layout.html" %}

{% block body %}

<div class="ind\_member\_history-container">

<h2>Member History for: {{user.name}}</h2>

{% if history\_data %}

<table border="1" cellspacing="0" cellpadding="5">

<thead>

<tr>

<th>Date</th>

<th>Share</th>

<th>Quantity</th>

</tr>

</thead>

<tbody>

{% for row in history\_data %}

<tr>

<td>{{row.transaction\_date}}</td>

<td>{{row.share\_name}}</td>

<td>{{row.quantity}}</td>

</tr>

{% endfor %}

</tbody>

</table>

{% else %}

<p>No history records found.</p>

{% endif %}

<div style="margin-top: 20px;">

<a href="{{url\_for('statusa', user\_id=user.id)}}">

<button type="button">BACK</button>

</a>

</div>

</div>

{% endblock %}

## members.html:

{% extends "layout.html" %}

{% block body %}

<h2>MEMBERS</h2>

<form method="POST" action="{{url\_for('members')}}">

<div>

<label for="new\_member\_name">Name:</label>

<input type="text" id="new\_member\_name" name="new\_member\_name" required>

</div>

<div>

<label for="share\_type\_id">Share:</label>

<select name="share\_type\_id" id="share\_type\_id">

<option value="">Select Share</option>

{% for share in share\_types %}

<option value="{{share[0]}}">{{share[1]}}</option>

{% endfor %}

</select>

</div>

<div style="margin-top: 10px;">

<button type="submit">ADD</button>

</div>

</form>

<h3>Existing Members:</h3>

<table border="1">

<thead>

<tr>

<th>Name</th>

<th>Share</th>

<th>Status</th>

</tr>

</thead>

<tbody>

{% for member in members\_list %}

<tr>

<td>{{member[1]}}</td>

<td>

{{member[2] if member[2] else "N/A"}}

</td>

<td>

<a href="{{url\_for('statusa', user\_id=member[0])}}">

<button type="button">STATUS</button>

</a>

</td>

</tr>

{% endfor %}

</tbody>

</table>

<div style="margin-top: 20px;">

<a href="{{url\_for('admin')}}">

<button>BACK</button>

</a>

</div>

{% endblock %}

## registrationa.html:

{% extends "layout.html" %}

{% block body %}

<h2>REGISTRATION (ADMIN)</h2>

<div class="admin\_registration-container">

<p>Delivered:</p>

<form method="POST" action="{{url\_for('registrationa')}}">

<div>

<label for="user\_id">Member:</label>

<select name="user\_id" id="user\_id" required>

<option value="">Select Member</option>

{% for client in all\_clients %}

<option value="{{client[0]}}" {% if previous\_input and previous\_input.user\_id == client[0]|string %}selected{% endif %}>{{client[1]}}</option>

{% endfor %}

</select>

</div>

<div>

<label for="product\_name">Product name:</label>

<input type="text" name="product\_name" id="product\_name" required value="{{previous\_input.product\_name if previous\_input else ''}}">

</div>

<div>

<label for="quantity">Quantity:</label>

<input type="number" name="quantity" id="quantity" min="1" required value="{{previous\_input.quantity if previous\_input else ''}}">

</div>

<div>

<label for="delivery\_date">Date:</label>

<input type="date" name="delivery\_date" id="delivery\_date" required value="{{previous\_input.delivery\_date if previous\_input else ''}}">

</div>

<div style="margin-top: 10px;">

<button type="submit">REGISTER</button>

</div>

</form>

<div style="margin-top: 20px;">

<a href="{{url\_for('admin')}}">

<button type="button">BACK</button>

</a>

</div>

</div>

{% endblock %}

## **registrationm.html:**

{% extends "layout.html" %}

{% block body %}

<div class="member\_registration-container">

<h2>REGISTRATION</h2>

<p>I picked up at the farm:</p>

<form method="POST" action="{{url\_for('registrationm')}}">

<div class="form-group">

<label for="transaction\_date">Picked up date:</label>

<input type="date" name="transaction\_date" id="transaction\_date" required>

</div>

<div class="form-group">

<label for="extra\_option">Would you like to exchange the chicken in your share for extra eggs:</label>

<select name="extra\_option" id="extra\_option">

<option value="YES">Yes</option>

<option value="NO">No</option>

</select>

</div>

<h3>Choose share:</h3>

{% for share in share\_types %}

<div class="form-group">

<label for="share\_{{share[0]}}">{{share[1]}}(Available: {{share[3]}})</label>

<input type="number" name="share\_{{share[0]}}" id="share\_{{share[0]}}" min="0" value="0" required>

</div >

{% endfor %}

<div class="form-group">

<button type="submit">REGISTER</button>

<a href="{{url\_for('statusm')}}">

<button type="button">BACK</button>

</a>

</div>

</form>

</div>

{% endblock %}

## shares.html:

{% extends "layout.html" %}

{% block body %}

<h2>SHARES</h2>

<form method="POST" action="{{url\_for('shares')}}">

<div>

<label for="share\_name">Share name:</label>

<input type="text" id="share\_name" name="share\_name" required>

</div>

<div>

<label for="share\_content">Share content:</label>

<input type="text" id="share\_content" name="share\_content" required>

</div>

<div>

<label for="content\_amount">Content amount:</label>

<input type="number" id="content\_amount" name="content\_amount" min="1" required>

</div>

<div style="margin-top: 10px;">

<button type="submit" name="add\_share" value="true">ADD</button>

</div>

</form>

<div style="margin-top: 20px;">

<a href="{{url\_for('admin')}}">

<button type="button">BACK</button>

</a>

</div>

<h3 style="margin-top: 30px;">Existing shares:</h3>

<table border="1">

<thead>

<tr>

<th>SHARE NAME</th>

<th>CONTENT</th>

<th>AMOUNT</th>

<th>REMOVE</th>

</tr>

</thead>

<tbody>

{% for share in share\_list %}

<tr>

<td>{{share[1]}}</td>

<td>{{share[2]}}</td>

<td>{{share[3]}}</td>

<td>

<form method="POST" action="{{url\_for('shares')}}" style="display: inline;">

<input type="hidden" name="remove\_id" value="{{share[0]}}">

<button type="submit" onclick="return confirm('Are you sure you want to remove this share type?');">REMOVE</button>

</form>

</td>

</tr>

{% endfor %}

</tbody>

</table>

{% endblock %}

## statusa.html:

{% extends "layout.html" %}

{% block body %}

<h2>Member Status for: {{user.name}}</h2>

<div class="admin\_status-container">

<p><strong>Status:</strong> {{user.status if user.status else "N/A"}}</p>

<p><strong>Member Name:</strong> {{user.name}}</p>

<p>

<strong>Share Type:</strong>

{% if user.share\_name %}

{{user.share\_name}}

{% else %}

(No share assigned)

{% endif %}

</p>

<p>

<strong>Extra Option (eggs in the place of a layer):</strong>

{% if user.extra\_option %}

Yes

{% else %}

No

{% endif %}

</p>

<p><strong>Total Delivered/Received:</strong> {{delivered\_total}}</p>

<p><strong>Remaining This Year's Allocation:</strong> {{remaining\_units}}</p>

<p><strong>Leftover from Previous Year:</strong> {{previous\_leftover}}</p>

<h3>Change Member Share Type</h3>

<form method="POST" action="{{url\_for('user\_share\_type', user\_id=user.id)}}">

<label for="new\_share\_type">Select new share type:</label>

<select name="new\_share\_type" id="new\_share\_type">

{% for share in all\_share\_types %}

<option value="{{share[0]}}"

{% if share[0] == user.share\_type\_id %} selected {% endif %}>

{{share[1]}}

</option>

{% endfor %}

</select>

<button type="submit">Update Share Type</button>

</form>

<h3>Delivered Products</h3>

<table border="1" cellspacing="0" cellpadding="5">

<thead>

<tr>

<th>Product</th>

<th>Quantity Delivered</th>

<th>Delivery Date</th>

<th>Action</th>

</tr>

</thead>

<tbody>

{% for delivery in deliveries %}

<tr>

<td>{{delivery.product\_name}}</td>

<td>{{delivery.quantity}}</td>

<td>{{delivery.delivery\_date}}</td>

<td>

<a href="{{url\_for('edit\_delivery', delivery\_id=delivery.id)}}">

<button type="button">EDIT</button>

</a>

</td>

</tr>

{% endfor %}

</tbody>

</table>

<div class="actions" style="margin-top: 20px;">

<a href="{{url\_for('member\_history', user\_id=user.id)}}">

<button type="button">MEMBER HISTORY</button>

</a>

<a href="{{url\_for('members')}}">

<button type="button">BACK</button>

</a>

</div>

</div>

{% endblock %}

## statusm.html:

{% extends "layout.html" %}

{% block body %}

<div class="member\_status-container">

<h2>STATUS</h2>

<div class="status-info">

<p>Name: {{user.name}}</p>

<p>Shares: {% if purchased\_shares\_names %}{{purchased\_shares\_names|join(", ")}}{% else %}No purchased shares{% endif %}</p>

<h3>Your shares overview:</h3>

<table border="1">

<thead>

<tr>

<th>Share content</th>

<th>Yearly amount</th>

<th>Already picked up/delivered</th>

<th>Leftover</th>

</tr>

</thead>

<tbody>

{% for share in share\_info %}

<tr>

<td>{{share.share\_name}}</td>

<td>{{share.yearly\_amount}}</td>

<td>{{share.total\_quantity}}</td>

<td>{{share.leftover}}</td>

</tr>

{% endfor %}

</tbody>

</table>

</div>

<a href="/registrationm">

<button type="button" name="member\_registration\_button">

REGISTRATION

</button>

</a>

<a href="/historym">

<button type="button" name="member\_history\_button">

HISTORY

</button>

</a>

<a href="/change\_password">

<button type="button" name="change\_password\_button">

CHANGE PASSWORD

</button>

</a>

<a href="/logout">

<button type="button" name="log\_off\_button">

LOG OFF

</button>

</a>

</div>

{% endblock %}