Rīgas Tālmācības Vidusskola

12. klase

**INVENTĀRA PĀRVALDĪBAS SISTĒMAS IZSTRĀDE**

Ralfs Tralmaks

Riga – 2025

**Saturs**

1. Problēmas izpēte un analīze 3
2. Programmatūras prasību specifikācija 4
3. Programmatūras izstrādes plāns 6
4. Atkļūdošanas un akcepttestēšanas pārskats 7
5. Lietotāja ceļvedis 8
6. Piemērotās licences pamatojums 12
7. Programmatūras kods 13
   1. app.py 13
   2. add\_user.html 28
   3. change\_password.html 30
   4. dashboard.html 32
   5. login.html 37
   6. rate\_limited.html 39
   7. view\_users.html 39
8. **Problēmas izpēte un analīze**

**Izpētes metodes un pamatojums**

Lai izprastu vajadzību pēc elektroniskas inventāra pārvaldības sistēmas, tika veikta aptauja starp maza biznesa un noliktavu īpašniekiem. Aptauja tika veikta, lai noskaidrotu, kādas metodes tiek izmantotas inventāra pārvaldībai un vai pastāv interese par elektroniskiem rīkiem.

**Izpētes datu apkopojums**

Aptaujāti tika 8 cilvēki, kuri bija radinieki vai draugi, un visi bija maza biznesa vai noliktavas īpašnieki.

* 3 no 8 cilvēkiem jau izmanto elektronisku inventāra pārvaldības rīku.
* 2 no 8 neizmanto nekādus rīkā inventāra pārvaldībai.
* 3 no 8 izmanto neelektronisku pierakstu (papīra formā).

No tiem, kas neizmanto elektronisku rīku, 4 izteica vēlmi pāriet uz elektronisku sistēmu.

Šie dati norāda uz nepieciešamību pēc vienkāršas, bet efektīvas elektroniskas sistēmas, kas atvieglotu inventāra pārvaldību un uzlabotu datu precizitāti.

1. **Programmatūras prasību specifikācija**

**Mērķauditorija**

Galvenā mērķauditorija šim projektam ir maza biznesa un noliktavu īpašnieki, kuriem ir nepieciešamība ātri, ērti un efektīvi pārvaldīt inventāru lokāli elektroniskā aplikācijā.

**Programmatūras apraksts**

Sistēma ir paredzēta inventāra uzskaites automatizācijai, ar dotajām funkcijām:

* Lietotāju autentifikācija
* Iespēja administrātoriem veidot lietotāju kontus ar atšķirīgiem piekļūves līmeņiem
* Paroles šifrēšana
* Administratīvs lietotāju parvaldes rīks (mainīt vai dzēst lietotājus)
* Pievienot, rediģēt un dzēst inventāra/noliktavas preces
* Meklēt un filtrēt inventāra/noliktavas preces
* Informācijas saglabāšana datubāzē

**Pielietotās tehnoloģijas**

Izvēlētā programmēšanas valoda un vide ir Python ar Flask ietvaru un SQLite datubāzi.

* **Python** ir viegli lasāma un uzturama valoda, kas nodrošina strauju izstrādi.
* **Flask** ir viegls un jaudīgs ietvars, kas ļauj elastīgi izstrādāt web lietojumprogrammas.
* **SQLite** ir vienkārši integrējama un uzturama datubāze, kas labi atbilst nelieliem un vidējiem projektiem.
* **Bootstrap** nodrošina ērtu un vizuāli pievilcīgu lietotāja saskarni.
* **Jquery**
* **Flask-WTF** atvieglo formu apstrādi un validāciju.
* **Flask-Session** ļauj saglabāt lietotāju sesijas informāciju.
* **Flask-Limiter** palīdz ierobežot pieprasījumu skaitu, uzlabojot drošību un aizsardzību pret pārmērīgu slodzi.

**Programmatūras sastāvs**

Wireframe diagramma, jeb applikācijas izplānotā izskata skice

**A screenshot of a computer

AI-generated content may be incorrect.**

[**Bildes pielikums**](https://imgur.com/a/DMdakJF)

Datubāzes shēma  
A screenshot of a computer

AI-generated content may be incorrect.

**3. Programmatūras izstrādes plāns**

**Izstrādes modelis**

Tiks izmantots Spējā programmatūras izstrādes modelis (Agile software development), kas ļauj pakāpeniski papildināt funkcionalitāti un pielāgot sistēmu lietotāju vajadzībām, balstoties uz regulāru atgriezenisko saiti.

**Izstrādes gaita**

1. Sākotnējā versija

* Programmatūras pamatstruktūras izveide, lietotāju autentifikācija, vadības panelis (dashboard), inventāra saraksts un datubāzes konfigurēšana.
* Drošības funkciju ieviešana, galvenokārt paroles šifrēšana.

1. Inventāra saraksta uzlabojumi

* Automātiska vienumu apvienošana, ja identisks vienums jau pastāv sarakstā.
* Iespēja meklēt sarakstā esošās sastāvdaļas.
* Automātiska esošo vienumu ieteikšana, sākot rakstīt to nosaukumu pievienošanas laukā (autocomplete funkcionalitāte).
* Saraksta vienumu kārtošanas iespējas: alfabētiskā secība, augošā un dilstošā secība.
* Iespēja mainīt sarakstā esošo sastāvdaļu daudzumu.
* Katras individuālās sastāvdaļas jaunākās izmaiņas vēstures saglabāšana.

1. Lietotāju pārvaldība
   * Administratori varēs pievienot jaunus lietotājus un iestatīt tiem paroles.
   * Administratori varēs apskatīt visus lietotājus un mainīt lietotāju paroles.
   * Lietotājiem būs iespēja mainīt savu paroli.

**4. Atkļūdošanas un akcepttestēšanas pārskats**

Tika veikta atkļūdošana un testēšana. Testēšanas procesā tika ietverti šādi soļi:

**Vienībtestēšana (Unit Testing)**

Mērķis ir pārbaudīt katru atsevišķu funkciju un moduli atsevišķi.

* Pārbaudīt datu validāciju (lietotāja ievadītie dati)
* Testēt CRUD operācijas (Create, Read, Update, Delete) datubāzē
* Pārbaudīt autentifikācijas un autorizācijas funkcionalitāti

Testēšanas laikā atklāju iespēju ignorēt lietotāja autentifikāciju un tikt vadības panelī, varot tam pievienot un rediģēt saraksta preces bez lietotāja autentifikācijas. Izlaboju to pievienojot visām esošajām lapām/vidēm pārbaudi, kas apskatās vai lietotājs ir autentificējies pirms to pielaiž pie vides.

**Integrācijas testēšana (Integration Testing)**

Šī testēšanas fāze nodrošina, ka dažādas sistēmas komponentes pareizi sadarbojas savā starpā.

* Testēta frontend un backend mijiedarbība
* Pārbaudītas datubāzes operāciju pareizība
* Simulētas lietotāju sesijas

**Akcepttestēšana (Acceptance Testing)**

Mērķis ir pārbaudīt, vai sistēma atbilst sākotnējām prasībām un ir lietojama reālajā vidē.

* Veikta manuālā testēšana, simulējot reālu lietotāju darbības
* Testēti drošības mehānismi, kā sesiju pārvaldība un piekļuves kontrole

**Veiktspējas un drošības testēšana**

* Simulēts liels pieprasījumu skaits, lai pārbaudītu Flask-Limiter iestatījumus
* Pārbaudīta paroļu drošība un sesiju ilgumu

**5. Lietotāja ceļvedis**

**Instalācija**

1. Instalēt [Python](https://www.python.org/downloads/) (vismaz versiju 3.11 vai jaunāku)
2. Palaist installprerequisite.bat (instalēs visas nepieciešamās atkarības) vai atvērt CMD un palaist doto komandu 
3. Palaist run.bat, lai startētu serveri
4. Atvērt <http://127.0.0.1:5000/>
5. Sākumlapai vajadzētu izskatīties šādi

A screenshot of a computer

AI-generated content may be incorrect.

**Lietošana**

Pēc noklusējuma būs izveidots admin lietotājs, pieslēdzies tam lai tiktu pie sistēms.

**Lietotājvārds: admin**

**Parole: admin**

**A screenshot of a computer

AI-generated content may be incorrect.**

* **Ieteikums uzreiz nomainīt admin konta paroli spiežot uz pelēko pogu “Mainīt paroli”!**
* Ir iespēja pievienot jaunus lietotājus spiežot uz zilās pogas “Pievienot lietotāju”. Aizpildi prasīto un izvēlies vai lietotājam būs administrātora rīki vai nē.

A screenshot of a computer

AI-generated content may be incorrect.

* Lai pievienotu kaut ko sarakstam, ieraksti to nosaukumu laukā un ieraksti to daudzumu. Pēc ievadīšanas spied pievienot.

A screenshot of a computer

AI-generated content may be incorrect.

* Ievadītajai mantai vajadzētu parādīties inventāra saraksta laukā ar to daudzumu. Inventāra saraksta mantas daudzumu var arī mainīt ievadot skaitļu lauciņā vēlamo dauzumu un nospiežot uz zilās pogas “Atjaunināt”. Mantas arī var dzēst no saraksta spiežot uz sarkanās pogas “Dzēst”.

A screenshot of a computer

AI-generated content may be incorrect.

* Ir iespēja arī meklēt mantas ierakstot to nosaukumu un spiežot uz zilās pogas “Meklēt”. Sarakstu arī var kārtot vairākos veidos nospiežot uz kārtošans izvēlnes.

A screenshot of a computer

AI-generated content may be incorrect.

* Administrātori var apskatīties visus esošos lietotājus un nomainīt to paroles. Spiežot uz dzeltenās pogas “Skatīt lietotājus”.

A screenshot of a computer

AI-generated content may be incorrect.

**6. Piemērotās licences pamatojums**

Šīs inventāra pārvaldības sistēmas izplatīšanai tiek izvēlēta MIT licence.

Tā tika izvēlēta, jo tā nodrošina:

* **Brīvu lietošana un izplatīšana** – MIT licence ļauj ikvienam brīvi izmantot, modificēt un izplatīt sistēmu gan personīgiem, gan komerciāliem nolūkiem.
* **Minimāli ierobežojumi** – Vienīgā prasība ir saglabāt sākotnējā autora autortiesības paziņojumu un licences tekstu.
* **Elastība un paplašināmība** – Lietotāji var pielāgot un paplašināt sistēmu atbilstoši savām vajadzībām, neuztraucoties par ierobežojumiem, ko varētu uzlikt stingrākas licences.

Licence veicina programmatūras brīvu izmantošanu un attīstību, vienlaikus nodrošinot juridisko aizsardzību izstrādātājam.

**7. Programmatūras kods**

**app.py:**

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

import sqlite3

from werkzeug.security import generate\_password\_hash, check\_password\_hash

from flask\_limiter import Limiter

from flask\_limiter.util import get\_remote\_address

from flask\_wtf.csrf import CSRFProtect

from datetime import timedelta, datetime

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

app.secret\_key = '58195673'

# CSRF protection

csrf = CSRFProtect(app)

# Brute force protection

limiter = Limiter(key\_func=get\_remote\_address)

limiter.init\_app(app)

# Session expiration timer

app.permanent\_session\_lifetime = timedelta(minutes=30)

# Local time converter

def convert\_to\_local\_time(utc\_time):

    if utc\_time:

        local\_time = utc\_time + timedelta(hours=2)

        return local\_time.strftime('%Y-%m-%d, %H:%M:%S')

    return "N/A"

# Database connection

def get\_db\_connection():

    conn = sqlite3.connect('database.db')

    conn.row\_factory = sqlite3.Row

    return conn

# Home Route

@app.route('/')

def home():

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

# Flask limiter error page

@app.errorhandler(429)

def ratelimit\_error(e):

    return render\_template('rate\_limited.html'), 429

# Login Route

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

@limiter.limit("5 per minute") # Limits to 5 attempts per minute

def login():

    if request.method == 'POST':

        username = request.form['username'].strip().lower()

        password = request.form['password']

        conn = get\_db\_connection()

        cursor = conn.cursor()

        cursor.execute("SELECT \* FROM users WHERE username = ?", (username,))

        user = cursor.fetchone()

        if user is None:

            flash("Lietotājs nav atrasts!", "danger")

        elif not check\_password\_hash(user['password'], password):

            flash("Nepareiza parole!", "danger")

        else:

            session['user\_id'] = user['id']

            flash("Veiksmīga pievienošanās!", "success")

            conn.close()

            return redirect(url\_for('dashboard'))

        conn.close()

    return render\_template('login.html')

# Logout Route

@app.route('/logout')

def logout():

    session.pop('user\_id', None)

    flash("Jūs esat izgājis!", "info")

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

# Admin check

def is\_admin(user\_id):

    conn = get\_db\_connection()

    cursor = conn.cursor()

    cursor.execute("SELECT \* FROM users WHERE id = ?", (user\_id,))

    user = cursor.fetchone()

    conn.close()

    return user['is\_admin'] == 1

# Dashboard Route

@app.route('/dashboard')

def dashboard():

    if 'user\_id' not in session:

        flash("Jums jāpieslēdzas!", "warning")

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

    admin\_status = is\_admin(session['user\_id'])

    sort\_option = request.args.get('sort', 'alphabet')

    search\_query = request.args.get('search', '').strip()

    try:

        conn = get\_db\_connection()

        query = "SELECT \* FROM inventory"

        params = []

        if search\_query:

            query += " WHERE name LIKE ?"

            params.append('%' + search\_query + '%')

        if sort\_option == 'asc':

            query += " ORDER BY quantity ASC"

        elif sort\_option == 'desc':

            query += " ORDER BY quantity DESC"

        else:

            query += " ORDER BY name ASC"

        items = conn.execute(query, tuple(params)).fetchall()

        # Process the items and return the response

        updated\_items = []

        for item in items:

            item\_dict = dict(item)

            if item\_dict['last\_modified\_by']:

                cursor = conn.execute("SELECT username FROM users WHERE id = ?", (item\_dict['last\_modified\_by'],))

                result = cursor.fetchone()

                item\_dict['last\_modified\_by\_username'] = result['username'] if result else "N/A"

            else:

                item\_dict['last\_modified\_by\_username'] = "N/A"

            if item\_dict.get('last\_modified\_at'):

                try:

                    last\_modified\_date = datetime.strptime(item\_dict['last\_modified\_at'], '%Y-%m-%d %H:%M:%S')

                    item\_dict['last\_modified\_at'] = last\_modified\_date

                except ValueError:

                    item\_dict['last\_modified\_at'] = None

            updated\_items.append(item\_dict)

        conn.close()

        return render\_template('dashboard.html', items=updated\_items, is\_admin=admin\_status, convert\_to\_local\_time=convert\_to\_local\_time)

    except Exception as e:

        app.logger.error(f"Error in dashboard route: {e}")

        return "An error occurred while processing your request.", 500

# Add Item Route

@app.route('/add\_item', methods=['POST'])

def add\_item():

    if 'user\_id' not in session:

        flash("Jums jāpieslēdzas!", "warning")

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

    name = request.form['name']

    quantity = request.form['quantity']

    user\_id = session['user\_id']

    conn = get\_db\_connection()

    cursor = conn.cursor()

    # Check if the item already exists

    cursor.execute("SELECT \* FROM inventory WHERE name = ?", (name,))

    existing\_item = cursor.fetchone()

    if existing\_item:

        new\_quantity = existing\_item['quantity'] + int(quantity)

        cursor.execute("UPDATE inventory SET quantity = ?, last\_modified\_by = ?, last\_modified\_at = CURRENT\_TIMESTAMP WHERE id = ?",

                       (new\_quantity, user\_id, existing\_item['id']))

    else:

        cursor.execute("INSERT INTO inventory (name, quantity, last\_modified\_by) VALUES (?, ?, ?)",

                       (name, quantity, user\_id))

    conn.commit()

    conn.close()

    flash("Prece pievienota vai atjaunināta!", "success")

    return redirect(url\_for('dashboard'))

# Delete Item Route

@app.route('/delete\_item/<int:item\_id>')

def delete\_item(item\_id):

    if 'user\_id' not in session:

        flash("Jums jāpieslēdzas!", "warning")

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

    conn = get\_db\_connection()

    cursor = conn.cursor()

    cursor.execute("DELETE FROM inventory WHERE id = ?", (item\_id,))

    conn.commit()

    conn.close()

    flash("Prece dzēsta!", "success")

    return redirect(url\_for('dashboard'))

# Add user function

@app.route('/add\_user', methods=['GET', 'POST'])

def add\_user():

    if 'user\_id' not in session or not is\_admin(session['user\_id']):

        flash("Tikai administratori var pievienot lietotājus!", "danger")

        return redirect(url\_for('dashboard'))

    if request.method == 'POST':

        username = request.form['username'].strip().lower()

        password = request.form['password']

        admin = request.form.get('is\_admin')

        if admin == "on":

            admin = 1

        else:

            admin = 0

        # Check username requirements

        if len(username) < 3:

            flash("Lietotājvārdam jābūt vismaz 3 rakstzīmēm!", "danger")

            return redirect(url\_for('add\_user'))

        elif len(username) > 25:

            flash("Lietotājvārdam jābūt īsākam!", "danger")

            return redirect(url\_for('add\_user'))

        # Check password requirements

        if len(password) < 8 or not any(c.isdigit() for c in password) or not any(c in "!@#$%^&\*" for c in password):

            flash("Parolei jābūt vismaz 8 rakstzīmēm, vienam ciparam un vienam simbolam", "danger")

            return redirect(url\_for('add\_user'))

        elif len(password) > 25:

            flash("Parolei jābūt īsākai!", "danger")

            return redirect(url\_for('add\_user'))

        hashed\_password = generate\_password\_hash(password, method="pbkdf2:sha256")

        try:

            conn = get\_db\_connection()

            cursor = conn.cursor()

            # Check if username already exists

            cursor.execute("SELECT \* FROM users WHERE username = ?", (username,))

            if cursor.fetchone():

                flash("Lietotājvārds jau eksistē!", "danger")

                return redirect(url\_for('add\_user'))

            # Insert new user

            cursor.execute("INSERT INTO users (username, password, is\_admin) VALUES (?, ?, ?)", (username, hashed\_password, admin))

            conn.commit()

            flash("Jauns lietotājs pievienots!", "success")

            return redirect(url\_for('dashboard'))

        finally:

            conn.close()

    return render\_template('add\_user.html')

# View all users tab

@app.route('/view\_users')

def view\_users():

    if 'user\_id' not in session or not is\_admin(session['user\_id']):

        flash("Jums nav atļaujas skatīt šo lapu.", "danger")

        return redirect(url\_for('dashboard'))

    conn = get\_db\_connection()

    cursor = conn.cursor()

    cursor.execute("SELECT id, username, is\_admin FROM users")

    users = cursor.fetchall()

    conn.close()

    return render\_template('view\_users.html', users=users)

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

def change\_password():

    if 'user\_id' not in session:

        flash("Vajag pieslēgties lai mainītu paroli!", "warning")

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

    if request.method == 'POST':

        old\_password = request.form['old\_password']

        new\_password = request.form['new\_password']

        # Get current user's data

        conn = get\_db\_connection()

        cursor = conn.cursor()

        cursor.execute("SELECT \* FROM users WHERE id = ?", (session['user\_id'],))

        user = cursor.fetchone()

        conn.close()

        # Check if the old password is correct

        if not check\_password\_hash(user['password'], old\_password):

            flash("Vecā parole ir nepareiza!", "danger")

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

        # Check password requirements

        if len(new\_password) < 8 or not any(c.isdigit() for c in new\_password) or not any(c in "!@#$%^&\*" for c in new\_password):

            flash("Parolei jābūt vismaz 8 rakstzīmēm, vienam ciparam un vienam simbolam", "danger")

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

        elif len(new\_password) > 25:

            flash("Parolei jābūt īsākai!", "danger")

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

        # Update password

        hashed\_password = generate\_password\_hash(new\_password, method="pbkdf2:sha256")

        conn = get\_db\_connection()

        cursor = conn.cursor()

        cursor.execute("UPDATE users SET password = ? WHERE id = ?", (hashed\_password, session['user\_id']))

        conn.commit()

        conn.close()

        flash("Parole mainīta!", "success")

        return redirect(url\_for('dashboard'))

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

# Autocomplete

@app.route('/autocomplete', methods=['GET'])

def autocomplete():

    query = request.args.get('query', '')

    conn = get\_db\_connection()

    cursor = conn.cursor()

    cursor.execute("SELECT name FROM inventory WHERE name LIKE ? LIMIT 5", (f"%{query}%",))

    results = [row['name'] for row in cursor.fetchall()]

    conn.close()

    return jsonify(results)

# Item quantity updater

@app.route('/update\_quantity', methods=['POST'])

def update\_quantity():

    if 'user\_id' not in session:

        flash("Lūdzu, piesakieties, lai veiktu izmaiņas.", "warning")

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

    item\_id = request.form.get('item\_id')

    new\_quantity = request.form.get('quantity')

    user\_id = session['user\_id']

    if not item\_id or not new\_quantity.isdigit():

        flash("Nederīga ievade!", "danger")

        return redirect(url\_for('dashboard'))

    conn = get\_db\_connection()

    cursor = conn.cursor()

    cursor.execute("UPDATE inventory SET quantity = ?, last\_modified\_by = ?, last\_modified\_at = CURRENT\_TIMESTAMP WHERE id = ?",

                   (int(new\_quantity), user\_id, item\_id))

    conn.commit()

    conn.close()

    flash("Daudzums atjaunināts!", "success")

    return redirect(url\_for('dashboard'))

# Password change function

@app.route('/reset\_password', methods=['POST'])

def reset\_password():

    if 'user\_id' not in session or not is\_admin(session['user\_id']):

        flash("Jums nav atļaujas veikt šo darbību.", "danger")

        return redirect(url\_for('dashboard'))

    user\_id = request.form.get('user\_id')

    new\_password = request.form.get('new\_password')

    if not user\_id or not new\_password:

        flash("Lūdzu ievadiet jauno paroli.", "danger")

        return redirect(url\_for('view\_users'))

    hashed\_password = generate\_password\_hash(new\_password)

    conn = get\_db\_connection()

    cursor = conn.cursor()

    cursor.execute("UPDATE users SET password = ? WHERE id = ?", (hashed\_password, user\_id))

    conn.commit()

    conn.close()

    flash("Lietotāja parole atjaunināta!", "success")

    return redirect(url\_for('view\_users'))

# Session handler

@app.before\_request

def check\_session\_timeout():

    session.modified = True  # Updates session timer on user activity

    if 'user\_id' not in session:

        if request.endpoint not in ['login', 'static']:

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

def init\_db():

    conn = get\_db\_connection()

    cursor = conn.cursor()

    # Create Users Table

    cursor.execute('''

        CREATE TABLE IF NOT EXISTS users (

            id INTEGER PRIMARY KEY AUTOINCREMENT,

            username TEXT UNIQUE NOT NULL,

            password TEXT NOT NULL,

            is\_admin INT NOT NULL

        )

    ''')

    # Create Inventory Table

    cursor.execute('''

        CREATE TABLE IF NOT EXISTS inventory (

            id INTEGER PRIMARY KEY AUTOINCREMENT,

            name TEXT NOT NULL,

            quantity INTEGER NOT NULL,

            last\_modified\_by INTEGER,

            last\_modified\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

            FOREIGN KEY (last\_modified\_by) REFERENCES users (id)

        )

    ''')

    # Insert an admin user

    cursor.execute("SELECT \* FROM users WHERE username = ?", ("admin",))

    if not cursor.fetchone():

        hashed\_password = generate\_password\_hash("admin")

        cursor.execute("INSERT INTO users (username, password, is\_admin) VALUES (?, ?, ?)", ("admin", hashed\_password, 1))

    conn.commit()

    conn.close()

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

    init\_db()

    app.run(debug=True)

**add\_user.html:**

<!DOCTYPE html>

<html lang="lv">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Pievienot Lietotāju</title>

    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">

</head>

<body class="d-flex justify-content-center align-items-center vh-100 bg-light">

    <div class="card p-4 shadow" style="width: 400px;">

        <h3 class="text-center">Pievienot Lietotāju</h3>

        {% with messages = get\_flashed\_messages(with\_categories=true) %}

            {% if messages %}

                {% for category, message in messages %}

                    <div class="alert alert-{{ category }} text-center">{{ message }}</div>

                {% endfor %}

            {% endif %}

        {% endwith %}

        <form method="POST">

            <input type="hidden" name="csrf\_token" value="{{ csrf\_token() }}">

            <div class="mb-3">

                <label for="username" class="form-label">Lietotājvārds:</label>

                <input type="text" class="form-control" id="username" name="username" required>

            </div>

            <div class="mb-3">

                <label for="password" class="form-label">Parole:</label>

                <input type="password" class="form-control" id="password" name="password" required>

            </div>

            <div class="mb-3 form-check">

                <input type="checkbox" class="form-check-input" id="is\_admin" name="is\_admin">

                <label class="form-check-label" for="is\_admin">Piešķirt administratora tiesības</label>

            </div>

            <button type="submit" class="btn btn-success w-100">Pievienot</button>

        </form>

        <div class="text-center mt-3">

            <a href="{{ url\_for('dashboard') }}" class="btn btn-secondary">Atpakaļ</a>

        </div>

    </div>

</body>

</html>

**change\_password.html**

<!DOCTYPE html>

<html lang="lv">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Mainīt paroli</title>

    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">

</head>

<body>

    <div class="container mt-5">

        <h2 class="mb-4">Paroles Maiņa</h2>

        {% with messages = get\_flashed\_messages(with\_categories=true) %}

            {% if messages %}

                {% for category, message in messages %}

                    <div class="alert alert-{{ category }} text-center">{{ message }}</div>

                {% endfor %}

            {% endif %}

        {% endwith %}

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

            <input type="hidden" name="csrf\_token" value="{{ csrf\_token() }}">

            <div class="mb-3">

                <label for="old\_password" class="form-label">Vecā parole</label>

                <input type="password" name="old\_password" id="old\_password" class="form-control" required>

            </div>

            <div class="mb-3">

                <label for="new\_password" class="form-label">Jaunā parole</label>

                <input type="password" name="new\_password" id="new\_password" class="form-control" required>

            </div>

            <button type="submit" class="btn btn-primary">Mainīt paroli</button>

        </form>

        <div class="mt-3">

            <a href="{{ url\_for('dashboard') }}" class="btn btn-secondary">Atpakaļ</a>

        </div>

    </div>

</body>

**dashboard.html**

<!DOCTYPE html>

<html lang="lv">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Inventāra pārvaldība</title>

    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">

    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

</head>

<body class="container mt-4">

    <header class="d-flex justify-content-between align-items-center">

        <h1>Inventāra pārvaldība</h1>

        <div class="header-right">

            {% if session['user\_id'] %}

                <a href="{{ url\_for('logout') }}" class="btn btn-danger">Iziet</a>

                <a href="{{ url\_for('change\_password') }}" class="btn btn-secondary">Mainīt paroli</a>

                {% if is\_admin %}

                    <a href="{{ url\_for('add\_user') }}" class="btn btn-primary">Pievienot lietotāju</a>

                    <a href="{{ url\_for('view\_users') }}" class="btn btn-warning">Skatīt lietotājus</a>

                {% endif %}

            {% endif %}

        </div>

    </header>

    {% with messages = get\_flashed\_messages(with\_categories=true) %}

        {% if messages %}

            {% for category, message in messages %}

                <div class="alert alert-{{ category }}">{{ message }}</div>

            {% endfor %}

        {% endif %}

    {% endwith %}

    <form action="{{ url\_for('add\_item') }}?sort={{ request.args.get('sort', 'alphabet') }}" method="POST" class="mb-4" id="inventory-form">

        <input type="hidden" name="csrf\_token" value="{{ csrf\_token() }}">

        <div class="mb-3">

            <label for="name" class="form-label">Preces nosaukums:</label>

            <input type="text" class="form-control" id="name" name="name" required autocomplete="off">

            <div id="autocomplete-list" class="list-group position-absolute"></div>

        </div>

        <div class="mb-3">

            <label for="quantity" class="form-label">Daudzums:</label>

            <input type="number" class="form-control" name="quantity" required>

        </div>

        <button type="submit" class="btn btn-success">Pievienot</button>

    </form>

    <div class="d-flex justify-content-between align-items-center mb-3">

        <h4 class="me-3" style="white-space: nowrap;">Inventāra saraksts</h4>

            <input type="text" id="search-input" class="form-control form-control-sm me-3" placeholder="Meklēt" value="{{ request.args.get('search', '') }}" style="flex-grow: 1;">

            <button id="search-btn" class="btn btn-primary me-3" style="height: 36px;">Meklēt</button>

        <div style="min-width: 115px;">

            <label for="sort" class="form-label me-2">Kārtot:</label>

            <select id="sort" class="form-select" onchange="window.location.href=this.value;">

                <option value="{{ url\_for('dashboard', sort='alphabet') }}" {% if request.args.get('sort') == 'alphabet' %} selected {% endif %}>Alfabēta</option>

                <option value="{{ url\_for('dashboard', sort='asc') }}" {% if request.args.get('sort') == 'asc' %} selected {% endif %}>Augošā</option>

                <option value="{{ url\_for('dashboard', sort='desc') }}" {% if request.args.get('sort') == 'desc' %} selected {% endif %}>Dilstošā</option>

            </select>

        </div>

    </div>

    <ul class="list-group" id="item-list">

        {% for item in items %}

            <li class="list-group-item d-flex justify-content-between align-items-center">

                <div>

                    <strong>{{ item.name }}</strong> ({{ item.quantity }})

                    <br>

                    <small class="text-muted">

                        Pēdējo izmaiņu veicis: {{ item.last\_modified\_by\_username or 'N/A' }}

                        {% if item.last\_modified\_at %}

                            {{ convert\_to\_local\_time(item.last\_modified\_at) }}

                        {% else %}

                            N/A

                        {% endif %}

                    </small>

                </div>

                <div class="d-flex">

                    <form method="POST" action="{{ url\_for('update\_quantity') }}" class="d-flex align-items-center">

                        <input type="hidden" name="csrf\_token" value="{{ csrf\_token() }}">

                        <input type="hidden" name="item\_id" value="{{ item.id }}">

                        <input type="number" name="quantity" class="form-control form-control-sm me-2" value="{{ item.quantity }}" required style="width: 70px;">

                        <button type="submit" class="btn btn-primary btn-sm">Atjaunināt</button>

                    </form>

                    <a href="{{ url\_for('delete\_item', item\_id=item.id) }}" class="btn btn-danger btn-sm ms-2">Dzēst</a>

                </div>

            </li>

        {% endfor %}

    </ul>

    <script>

        $(document).ready(function() {

            $('#search-btn').on('click', function() {

                var query = $('#search-input').val();

                window.location.href = "{{ url\_for('dashboard') }}?search=" + query;  // Perform the search

            });

        });

        $("#name").on("input", function() {

            let query = $(this).val();

            if (query.length > 1) {

                $.get("{{ url\_for('autocomplete') }}", { query: query }, function(data) {

                    let autocompleteList = $("#autocomplete-list");

                    autocompleteList.empty();

                    data.forEach(function(item) {

                        autocompleteList.append(`<div class="list-group-item list-group-item-action">${item}</div>`);

                    });

                    $(".list-group-item-action").on("click", function() {

                        $("#name").val($(this).text());

                        autocompleteList.empty();

                    });

                });

            };

        });

        // Caps first letter

        document.getElementById("inventory-form").addEventListener("submit", function(event) {

        let itemNameInput = document.getElementById("name");

        let itemName = itemNameInput.value.trim();

        if (itemName) {

            itemNameInput.value = itemName.charAt(0).toUpperCase() + itemName.slice(1);

        }});

    </script>

</body>

</html>

**login.html**

<!DOCTYPE html>

<html lang="lv">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Pieslēgties</title>

    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">

</head>

<body class="d-flex justify-content-center align-items-center vh-100 bg-light">

    <div class="card p-4 shadow" style="width: 300px;">

        <h3 class="text-center">Pieslēgties</h3>

        {% with messages = get\_flashed\_messages(with\_categories=true) %}

            {% if messages %}

                {% for category, message in messages %}

                    <div class="alert alert-{{ category }} text-center">{{ message }}</div>

                {% endfor %}

            {% endif %}

        {% endwith %}

        <form method="POST">

            <input type="hidden" name="csrf\_token" value="{{ csrf\_token() }}">

            <div class="mb-3">

                <label for="username" class="form-label">Lietotājvārds:</label>

                <input type="text" class="form-control" id="username" name="username" required>

            </div>

            <div class="mb-3">

                <label for="password" class="form-label">Parole:</label>

                <input type="password" class="form-control" id="password" name="password" required>

            </div>

            <button type="submit" class="btn btn-primary w-100">Pieslēgties</button>

        </form>

    </div>

</body>

</html>

**rate\_limited.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Mēginājumu ierobežojums pārsniegts</title>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

</head>

<body class="container text-center mt-5">

    <h1 class="text-danger">Mēginājumu ierobežojums pārsniegts</h1>

    <p>Pārāk daudz mēģinājumi! Mēģini vēlreiz vēlāk.</p>

    <a href="{{ url\_for('login') }}" class="btn btn-primary">Atpakaļ</a>

</body>

</html>

**view\_users.html**

<!DOCTYPE html>

<html lang="lv">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Lietotāju saraksts</title>

    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">

</head>

<body class="container mt-4">

    <h2>Lietotāju saraksts</h2>

    <a href="{{ url\_for('dashboard') }}" class="btn btn-secondary mb-3">Atpakaļ</a>

    <table class="table">

        <thead>

            <tr>

                <th>ID</th>

                <th>Lietotājvārds</th>

                <th>Darbība</th>

            </tr>

        </thead>

        <tbody>

            {% for user in users %}

            <tr>

                <td>{{ user.id }}</td>

                <td>{{ user.username }}</td>

                <td>

                    {% if user.is\_admin %}

                        <button class="btn btn-secondary btn-sm" disabled>Nav pieejams</button>

                    {% else %}

                        <button class="btn btn-warning btn-sm" onclick="showResetForm('{{ user.id }}')">Mainīt paroli</button>

                    {% endif %}

                </td>

            </tr>

            <tr id="reset-form-{{ user.id }}" style="display: none;">

                <td colspan="3">

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

                        <input type="hidden" name="csrf\_token" value="{{ csrf\_token() }}">

                        <input type="hidden" name="user\_id" value="{{ user.id }}">

                        <div class="mb-2">

                            <label for="new\_password" class="form-label">Jaunā parole:</label>

                            <input type="password" name="new\_password" class="form-control" required>

                        </div>

                        <button type="submit" class="btn btn-success btn-sm">Saglabāt</button>

                        <button type="button" class="btn btn-secondary btn-sm" onclick="hideResetForm('{{ user.id }}')">Atcelt</button>

                    </form>

                </td>

            </tr>

            {% endfor %}

        </tbody>

    </table>

    <script>

        function showResetForm(userId) {

            document.getElementById("reset-form-" + userId).style.display = "table-row";

        }

        function hideResetForm(userId) {

            document.getElementById("reset-form-" + userId).style.display = "none";

        }

    </script>

</body>

</html>