**Project Documentation: MediLog - Personal Medication Tracker**

**1. Project Overview:**

**MediLog** is a secure **full-stack web application** designed to help users track their medications. Users can log details such as dosage, frequency, start date, and status (Active/Discontinued). The app ensures **user-specific data ownership**, meaning each user can only access and manage their own medication records.

**Objective:**

* Build a **Python-based** full-stack application with **authentication**.
* Implement **CRUD (Create, Read, Update, Delete)** operations for medication tracking.
* Provide **filtering, sorting, and summary statistics** for better usability.
* Ensure a **responsive UI** that works on different devices.

**2. Technology Stack:**

| **Category** | **Technologies Used** |
| --- | --- |
| **Frontend** | HTML, CSS, JavaScript, Bootstrap (for responsive design) |
| **Backend** | Python (Flask Framework) |
| **Database** | SQLite (for development) / MySQL (for production) |
| **Authentication** | Flask-Login (session-based) |
| **Deployment** | Render / PythonAnywhere / Heroku |

**3. Key Features:**

**A. User Authentication:**

* **Registration:** Users can sign up with email & password.
* **Login/Logout:** Secure session management using Flask-Login.
* **User-Specific Data:** Each user sees only their own medication records.

**B. Medication Management (CRUD Operations):**

| **Feature** | **Description** |
| --- | --- |
| **Create** | Add new medications (Name, Dosage, Frequency, Start Date, Notes, Status). |
| **Read** | View all medications in a list or see detailed info for a single medication. |
| **Update** | Edit medication details (e.g., change status from "Active" to "Discontinued"). |
| **Delete** | Remove a medication entry permanently. |

**C. Filtering & Summary:**

* **Filter by Status** (Active/Discontinued).
* **Sort by:**
  + Medication Name (A-Z)
  + Start Date (Newest-Oldest)
* **Summary Stats:**
  + *"Total Active Medications: X"*
  + *"Total Discontinued Medications: Y"*

**D. Responsive UI**

* Works on **desktops, tablets, and mobile devices**.
* Built with **Bootstrap** for consistent styling.

**4. Database Schema:**

**Tables**

1. users
   * id (Primary Key)
   * email (Unique)
   * password\_hash (Securely hashed)
2. medications
   * id (Primary Key)
   * user\_id (Foreign Key → users.id)
   * name (e.g., "Aspirin")
   * dosage (e.g., "500mg")
   * frequency (e.g., "Twice daily")
   * start\_date (Date started)
   * notes (Optional additional info)
   * status (Active / Discontinued)

**5. Setup & Installation:**

**Prerequisites**

* Python 3.x
* Flask (pip install flask flask-login flask-sqlalchemy)

**Steps to Run Locally**

1. **Clone the Repository** : git clone https://github.com/yourusername/medilog.git

cd medilog

1. **Install Dependencies** : pip install -r requirements.txt
2. **Set Up Database** : python init\_db.py # Creates SQLite database & tables
3. **Run the Application** : python app.py
   * Open http://localhost:5000 in browser.
4. SOURCE CODE:

**2.1. Backend Setup (**app.py**)**

python

Copy

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

from flask\_sqlalchemy import SQLAlchemy

from flask\_login import LoginManager, UserMixin, login\_user, login\_required, logout\_user, current\_user

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

from datetime import datetime

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

app.config['SECRET\_KEY'] = 'your-secret-key'

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///medilog.db'

app.config['SQLALCHEMY\_TRACK\_MODIFICATIONS'] = False

db = SQLAlchemy(app)

login\_manager = LoginManager(app)

login\_manager.login\_view = 'login'

# Database Models

class User(UserMixin, db.Model):

id = db.Column(db.Integer, primary\_key=True)

email = db.Column(db.String(100), unique=True)

password = db.Column(db.String(100))

medications = db.relationship('Medication', backref='user', lazy=True)

class Medication(db.Model):

id = db.Column(db.Integer, primary\_key=True)

name = db.Column(db.String(100))

dosage = db.Column(db.String(50))

frequency = db.Column(db.String(50))

start\_date = db.Column(db.Date)

notes = db.Column(db.Text)

status = db.Column(db.String(20)) # 'Active' or 'Discontinued'

user\_id = db.Column(db.Integer, db.ForeignKey('user.id'))

# Flask-Login Loader

@login\_manager.user\_loader

def load\_user(user\_id):

return User.query.get(int(user\_id))

# Routes

@app.route('/')

@login\_required

def index():

medications = Medication.query.filter\_by(user\_id=current\_user.id).all()

active\_meds = Medication.query.filter\_by(user\_id=current\_user.id, status='Active').count()

return render\_template('index.html', medications=medications, active\_meds=active\_meds)

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

def login():

if request.method == 'POST':

email = request.form.get('email')

password = request.form.get('password')

user = User.query.filter\_by(email=email).first()

if user and check\_password\_hash(user.password, password):

login\_user(user)

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

flash('Invalid email or password!')

return render\_template('login.html')

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

def register():

if request.method == 'POST':

email = request.form.get('email')

password = generate\_password\_hash(request.form.get('password'))

new\_user = User(email=email, password=password)

db.session.add(new\_user)

db.session.commit()

flash('Account created! Please log in.')

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

return render\_template('register.html')

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

@login\_required

def add\_medication():

if request.method == 'POST':

name = request.form.get('name')

dosage = request.form.get('dosage')

frequency = request.form.get('frequency')

start\_date = datetime.strptime(request.form.get('start\_date'), '%Y-%m-%d').date()

notes = request.form.get('notes')

status = request.form.get('status')

new\_med = Medication(

name=name,

dosage=dosage,

frequency=frequency,

start\_date=start\_date,

notes=notes,

status=status,

user\_id=current\_user.id

)

db.session.add(new\_med)

db.session.commit()

flash('Medication added successfully!')

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

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

@app.route('/logout')

@login\_required

def logout():

logout\_user()

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

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

with app.app\_context():

db.create\_all()

app.run(debug=True)

**2.2. Frontend (HTML Templates)**

templates/index.html**(Dashboard)**

html

Copy

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>MediLog - Dashboard</title>

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

</head>

<body>

<div class="container mt-4">

<h2>Your Medications</h2>

<p>Total Active Medications: {{ active\_meds }}</p>

<a href="{{ url\_for('add\_medication') }}" class="btn btn-primary mb-3">Add New Medication</a>

<table class="table">

<thead>

<tr>

<th>Name</th>

<th>Dosage</th>

<th>Frequency</th>

<th>Status</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

{% for med in medications %}

<tr>

<td>{{ med.name }}</td>

<td>{{ med.dosage }}</td>

<td>{{ med.frequency }}</td>

<td>{{ med.status }}</td>

<td>

<a href="#" class="btn btn-sm btn-info">Edit</a>

<a href="#" class="btn btn-sm btn-danger">Delete</a>

</td>

</tr>

{% endfor %}

</tbody>

</table>

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

</div>

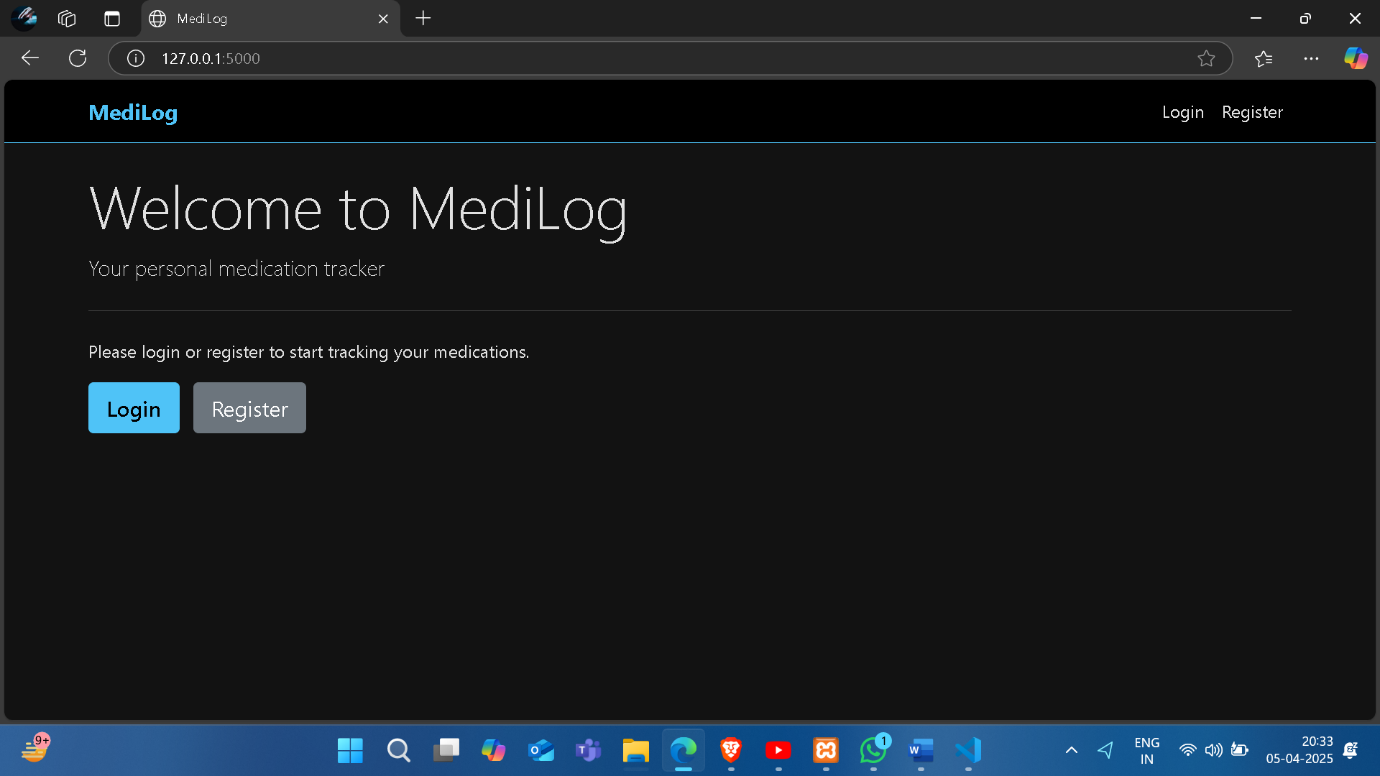
</body>

</html>

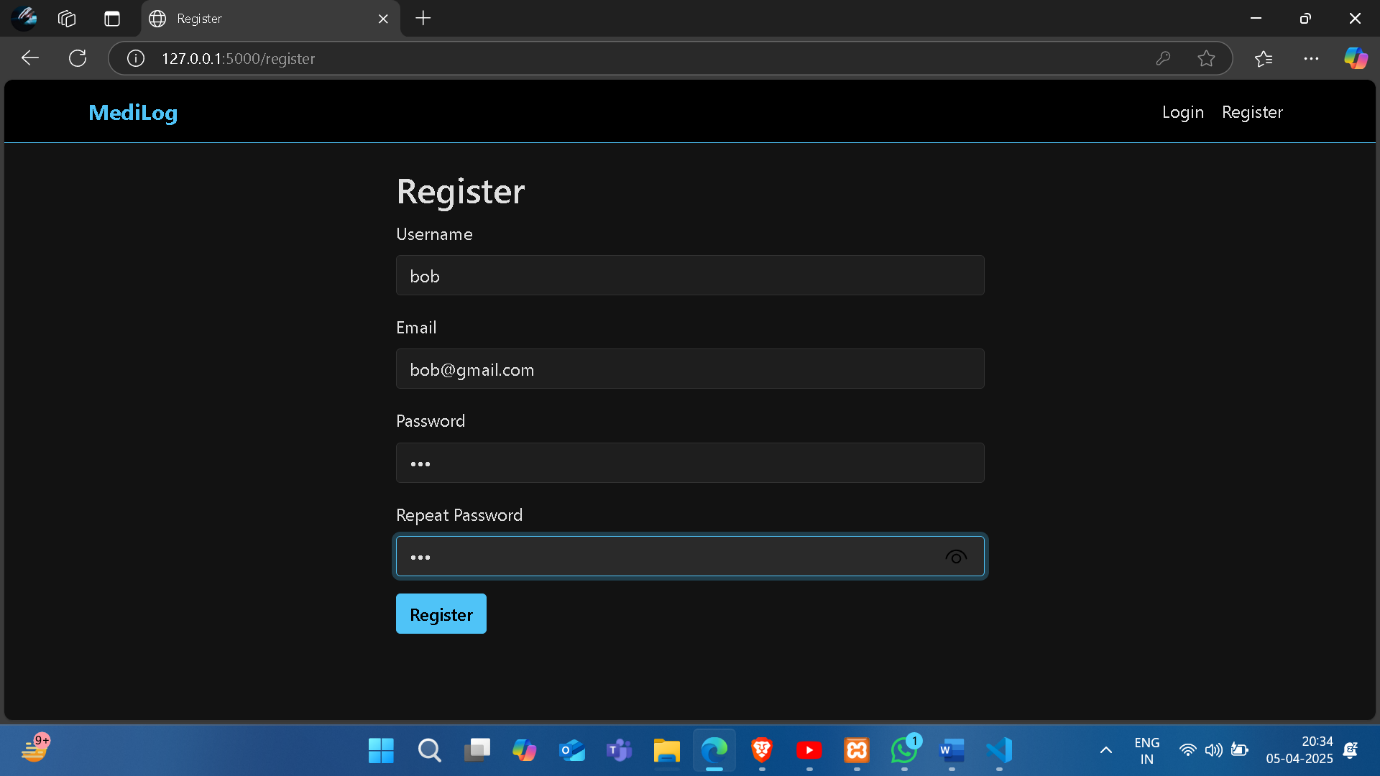
Run HTML

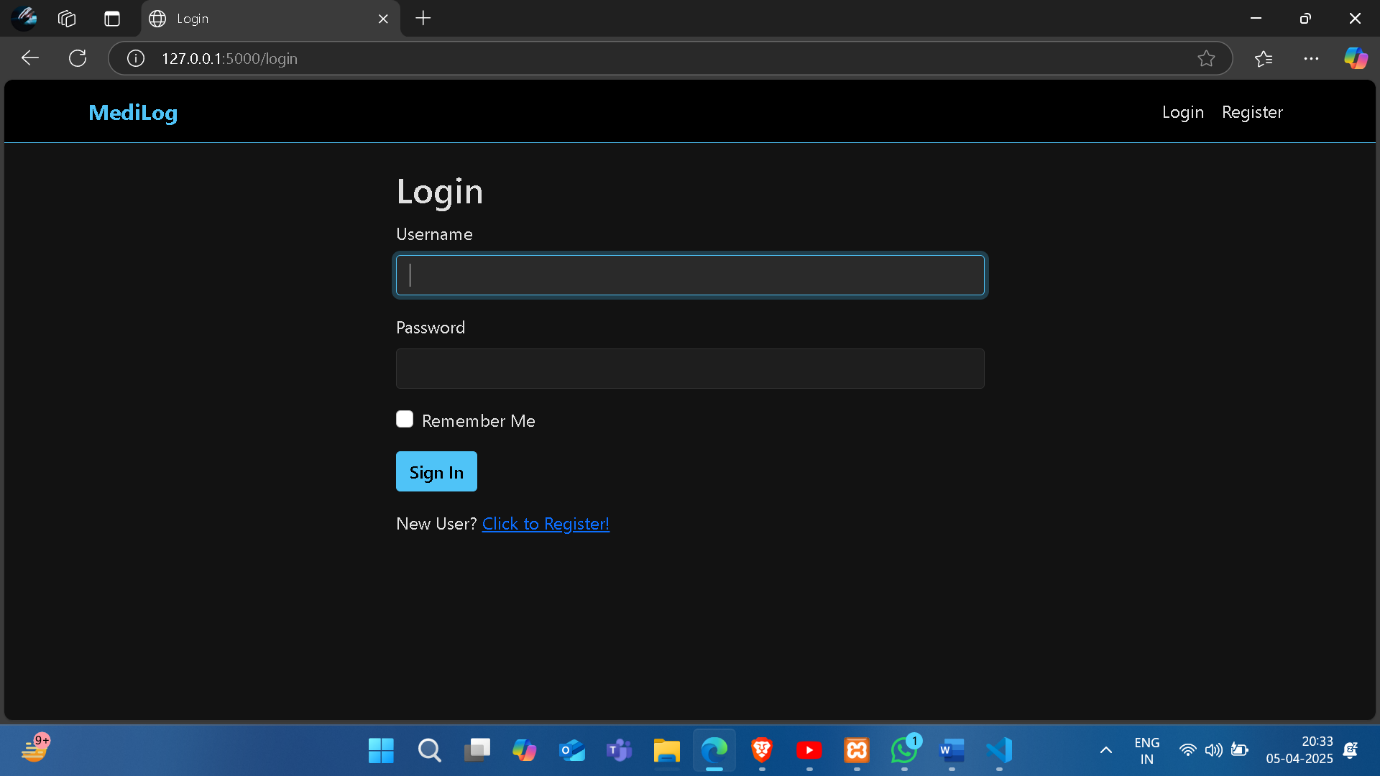
**6. Screenshots:**

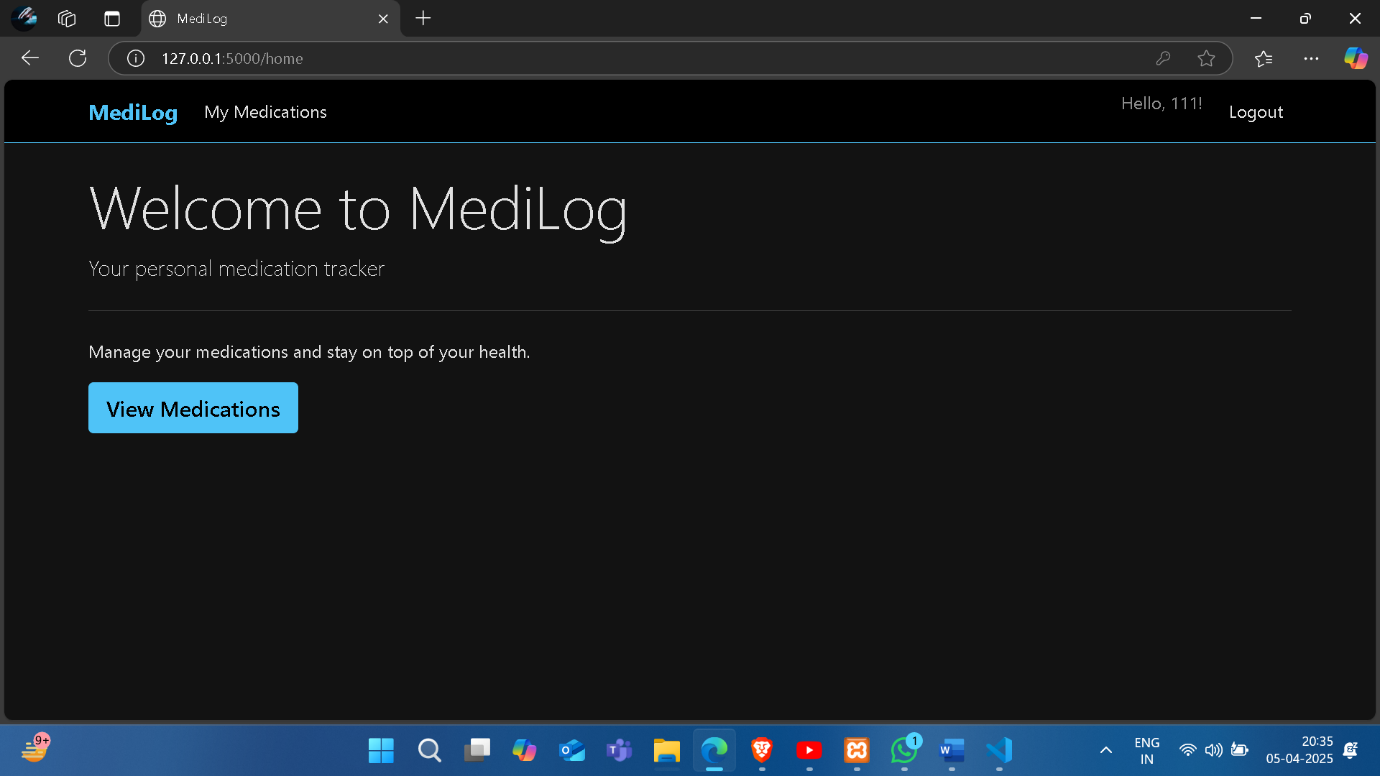
* **welcome Page:**

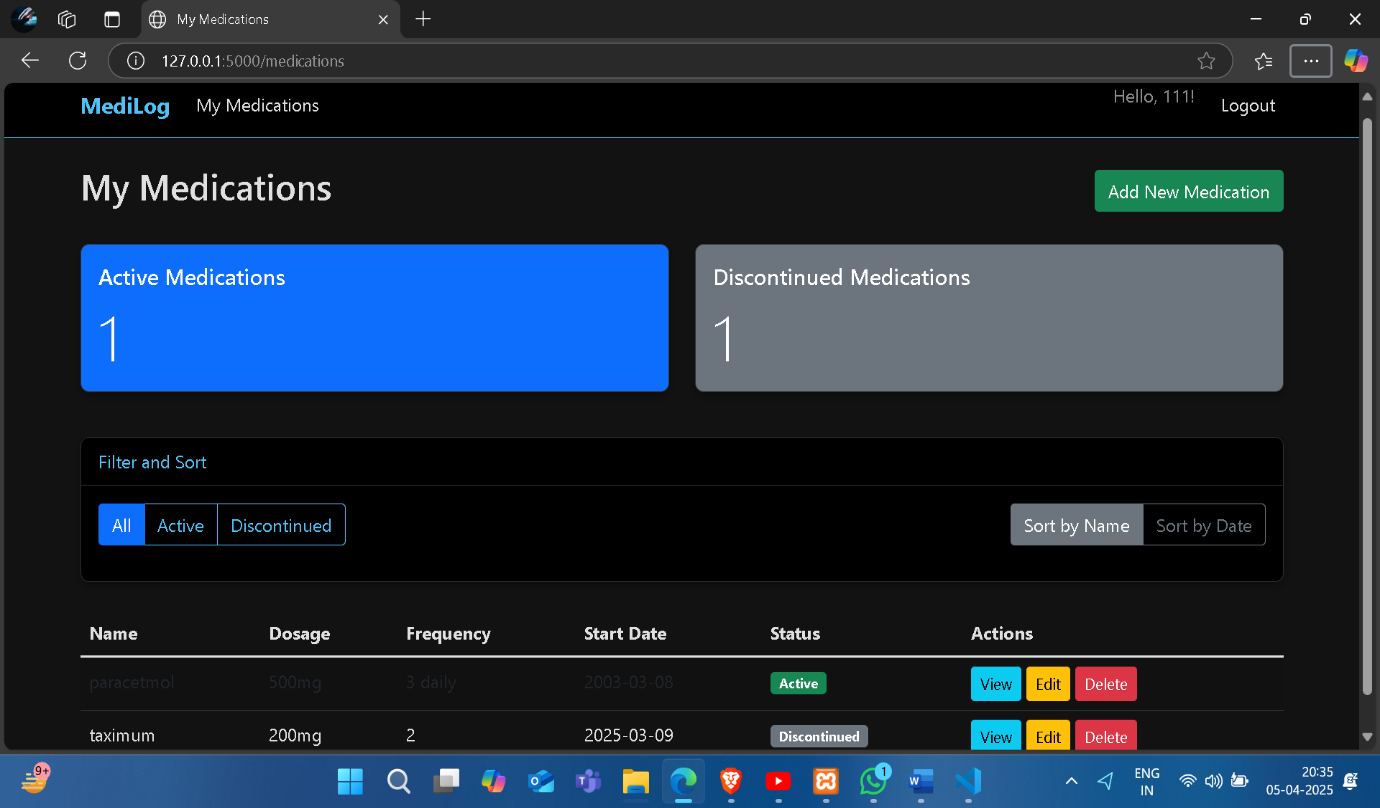


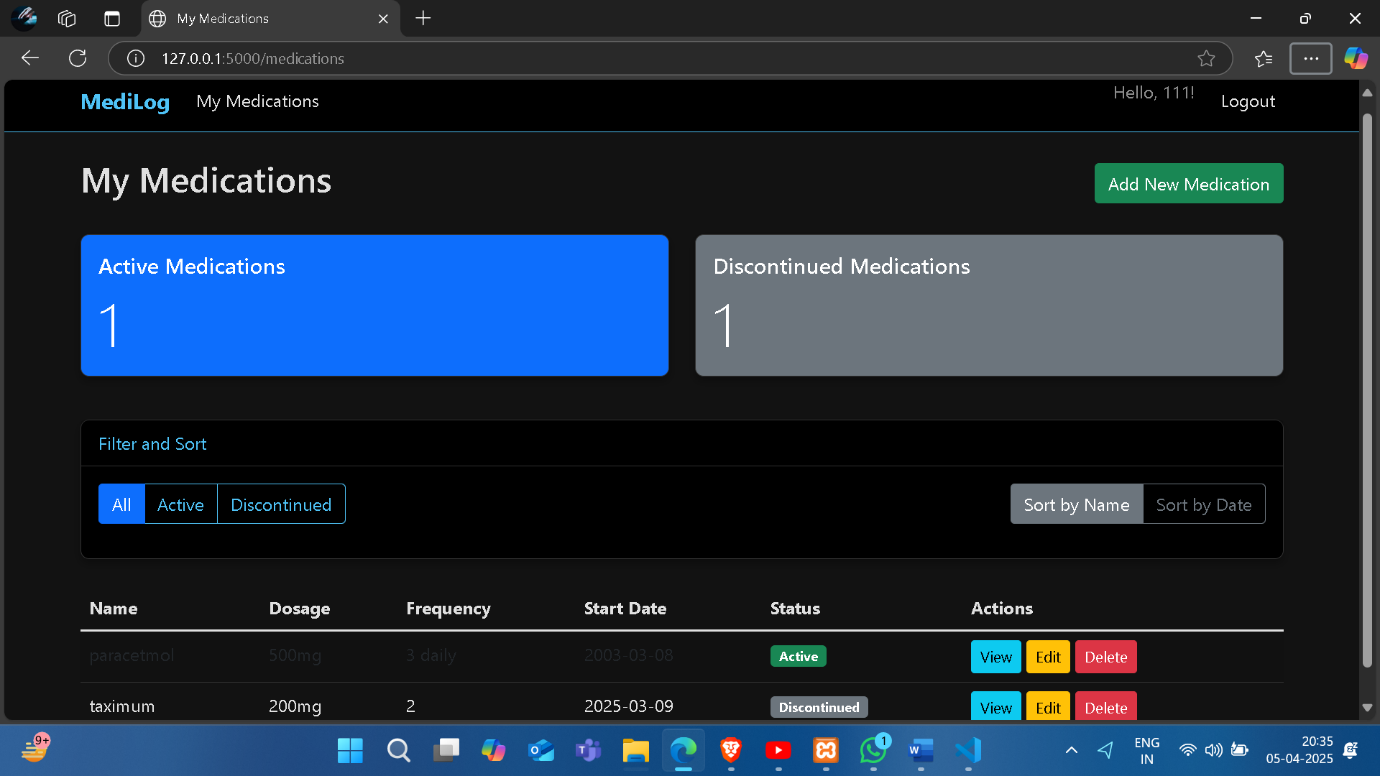
* Register page:



* Login page:
* 
* Home page:



* **Add/Edit Medication Form:**
* 
* Final output:



**7. Future Enhancements:**

* **Reminder System** (Email/SMS notifications for medication schedules).
* **Data Export** (Download medication list as PDF/CSV).
* **Multi-User Access** (Family members tracking medications for elderly patients).
* **API Integration** (Fetch medication details from a pharmacy database).

**8. Conclusion:**

**MediLog** successfully delivers a **secure, user-friendly medication tracker** with core CRUD operations, filtering, and summary statistics. Built with **Python (Flask)** and designed for **responsiveness**, it serves as a practical tool for personal healthcare management.

🔗 **GitHub Repository:** https://github.com/Akhilesh-dot76/submittechh

**Appendix: Code Structure**

mediilog/

├── app.py # Flask backend (routes, auth, CRUD logic)

├── models.py # Database models (User, Medication)

├── templates/ # HTML files (Jinja2 templates)

│ ├── login.html

│ ├── dashboard.html

│ ├── add\_medication.html

├── static/ # CSS/JS files

│ ├── style.css

│ └── script.js

├── requirements.txt # Dependencies

└── README.md # Setup guide

THANK YOU