LAPORAN PRAKTIKUM PENGEMBANGAN APLIKASI WEB

BAB	: WEB FRAMEWORK
NAMA	: Akwila Febryan Santoso
NIM	: 235150201111057
ASISTEN	: - NAJWA MUTHIA AZIZ - DAYANG ALYSSA RAISAPUTRI
TANGGAL PRAKTIKUM	: 15/05/2025

10.3. PROSEDUR/PELAKSANAAN PRAKTIKUM

a. Pilihlah salah satu web framework yang mengundang minat saudara.

b. Dengan framework tersebut, buatlah sebuah aplikasi sederhana dengan fitur dasar CRUD (Create, Read, Update, Delete) dan disertai autentikasi pengguna dengan ketentuan sebagai berikut:

1. Operasi read dapat dilakukan siapa saja

2. Operasi create, update, dan delete hanya dapat dilakukan oleh pengguna yang sudah login.

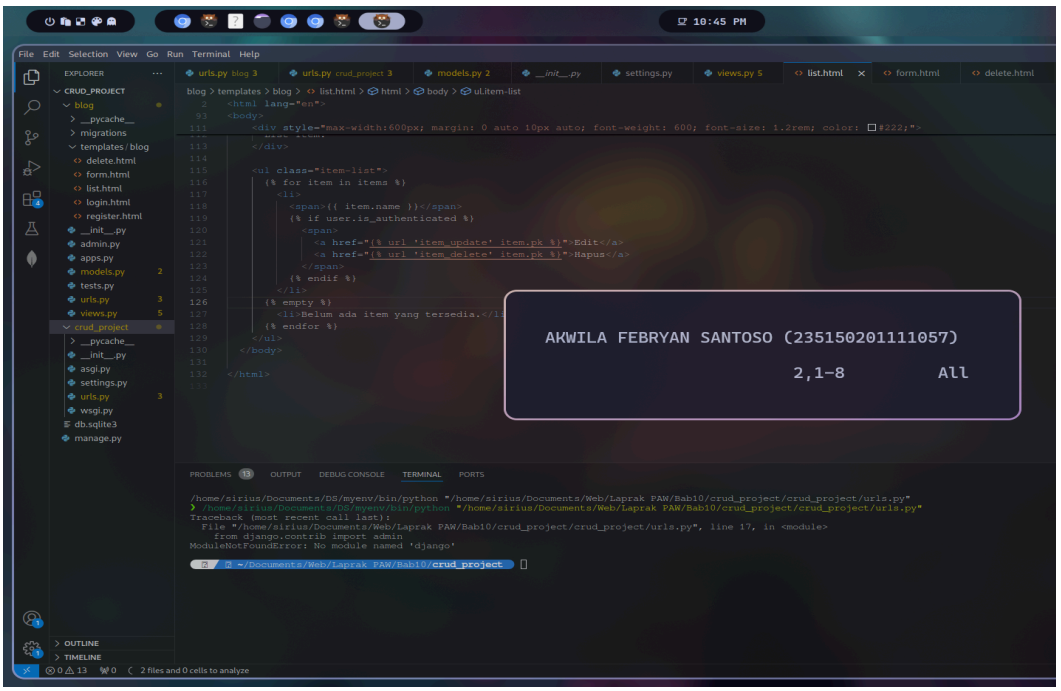
10.4. TUGAS ANALISIS HASIL PRAKTIKUM

A. Soal

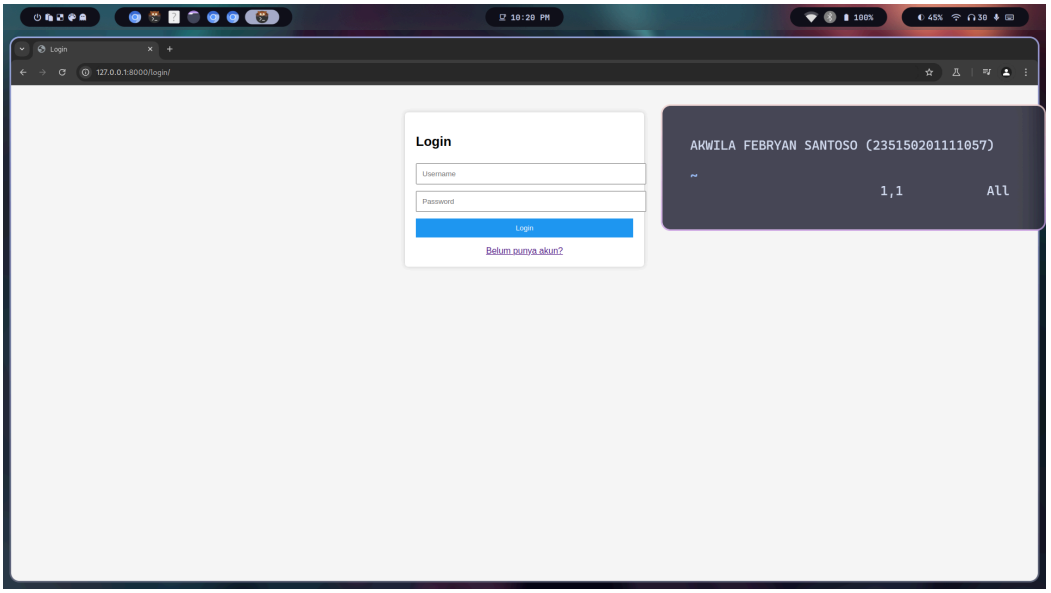
Setelah selesai membuat program, uji coba dengan melakukan operasi CRUD dan autentikasi pengguna.

Tempelkan seluruh screenshot output yang dihasilkan pada setiap operasi!

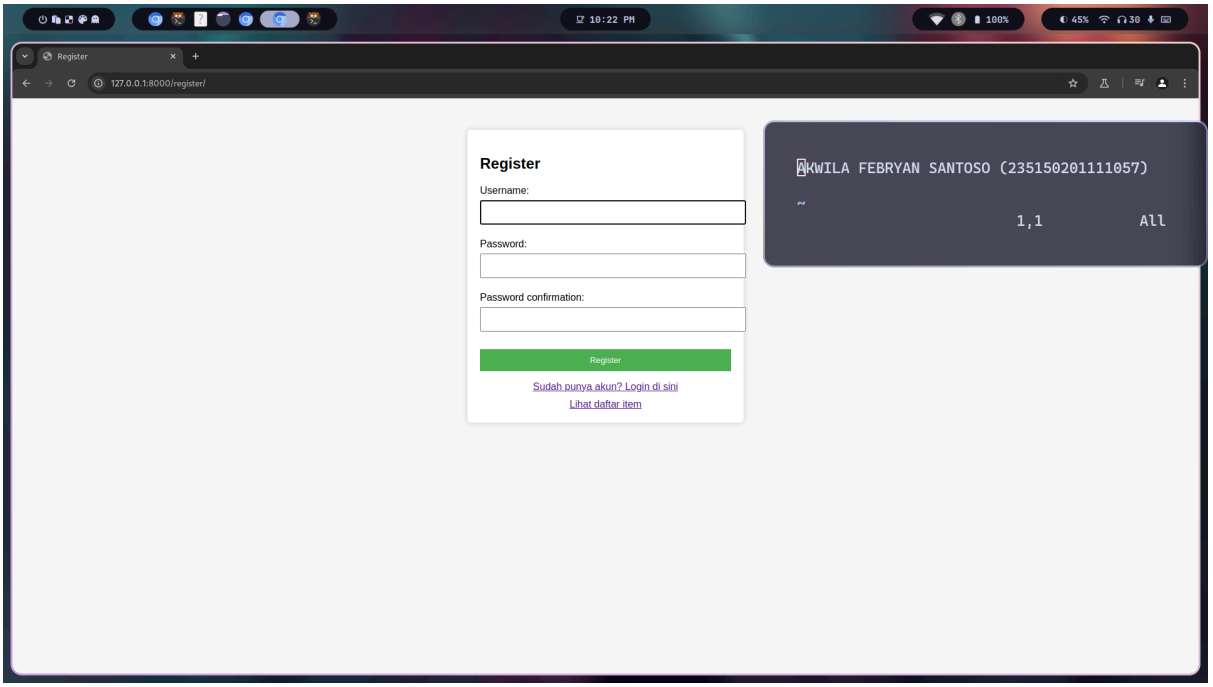
B. Screenshoot



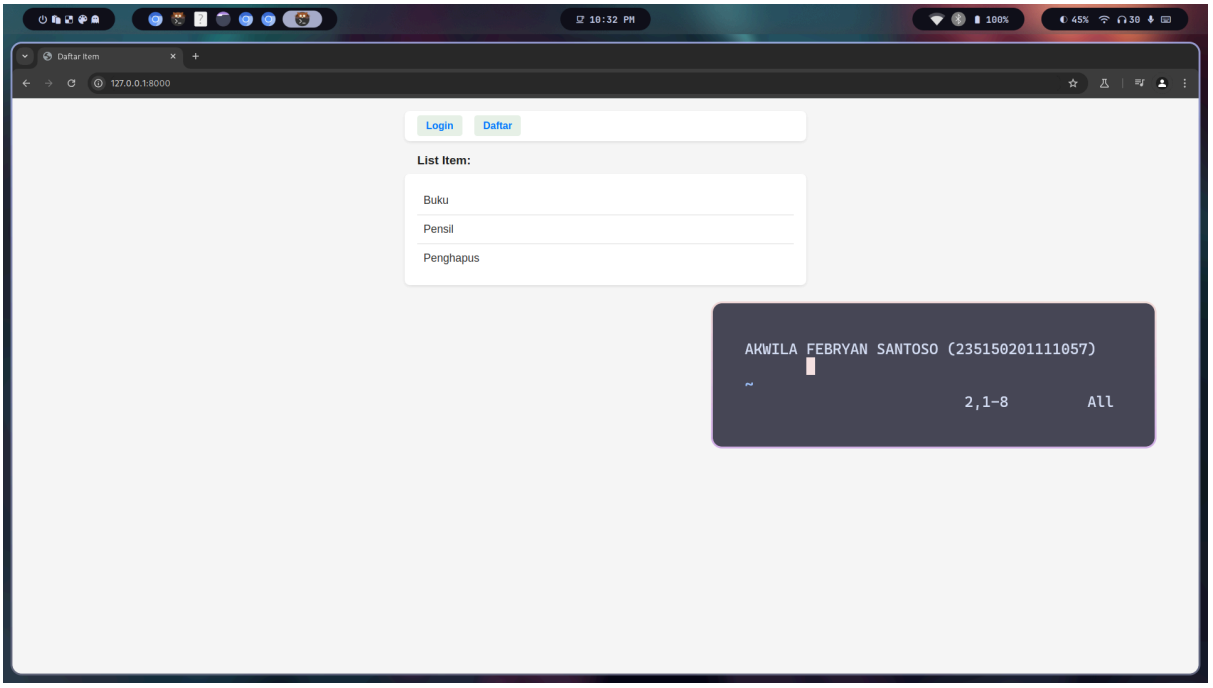
Gambar Struktur Kode



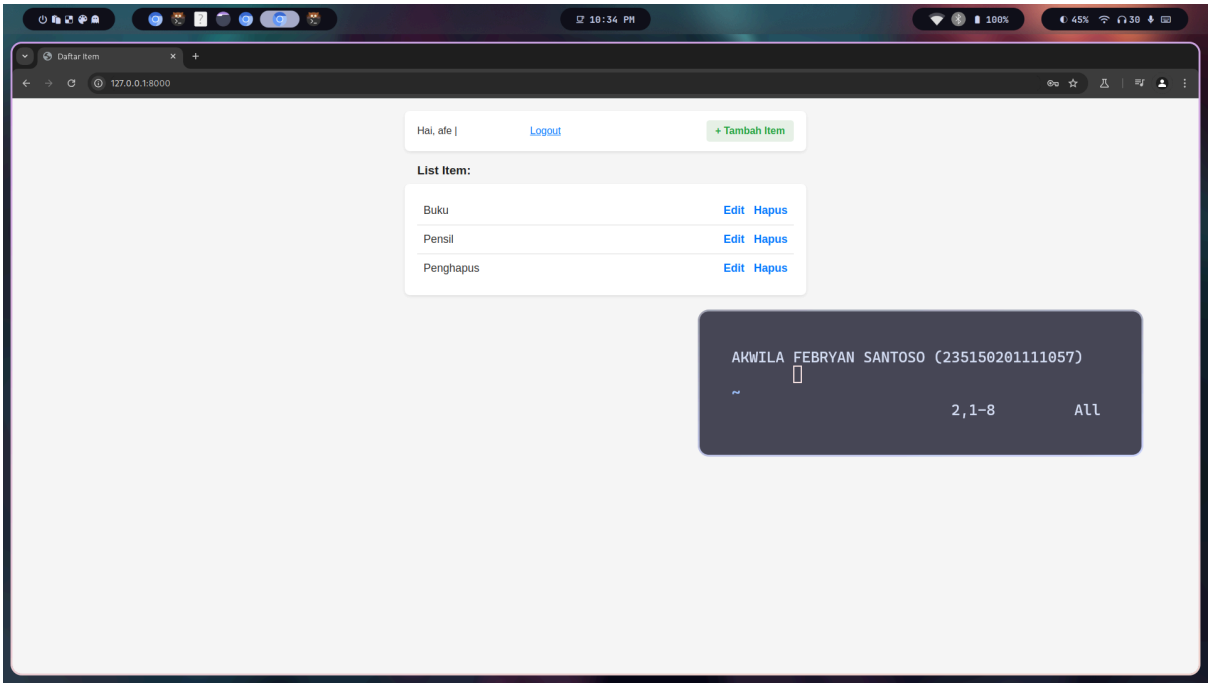
Gambar Halaman Login



Gambar Halaman Register

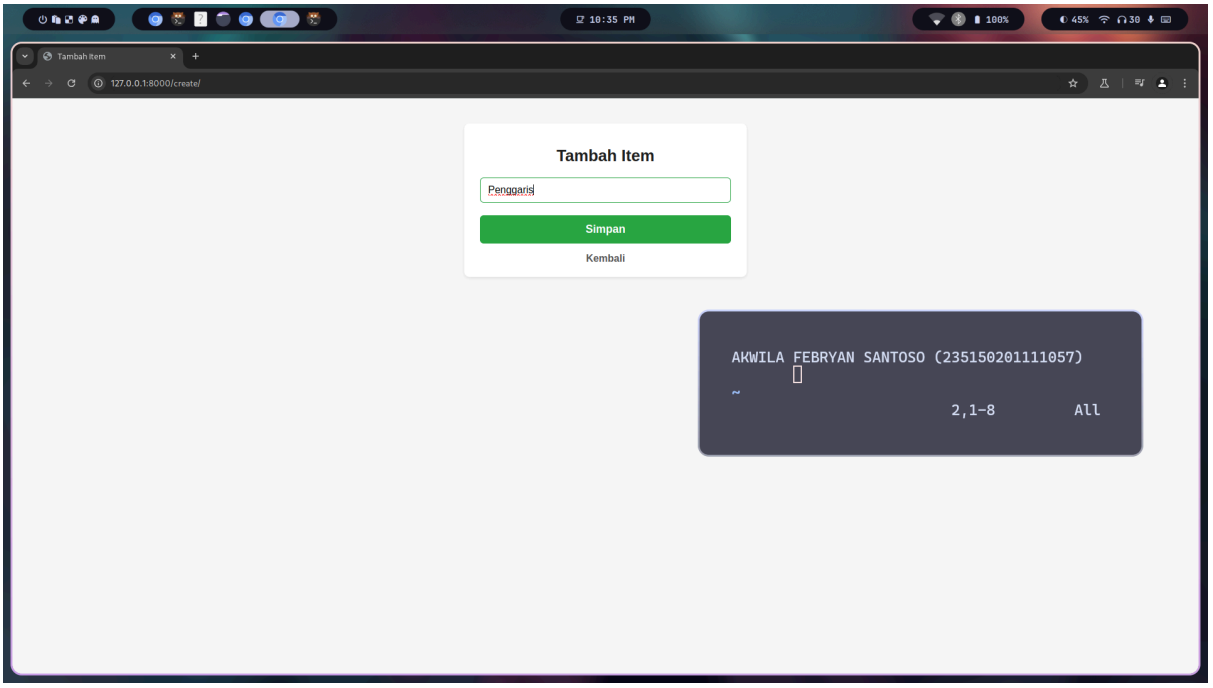


Gambar Halaman List Item Tanpa Pengguna Login

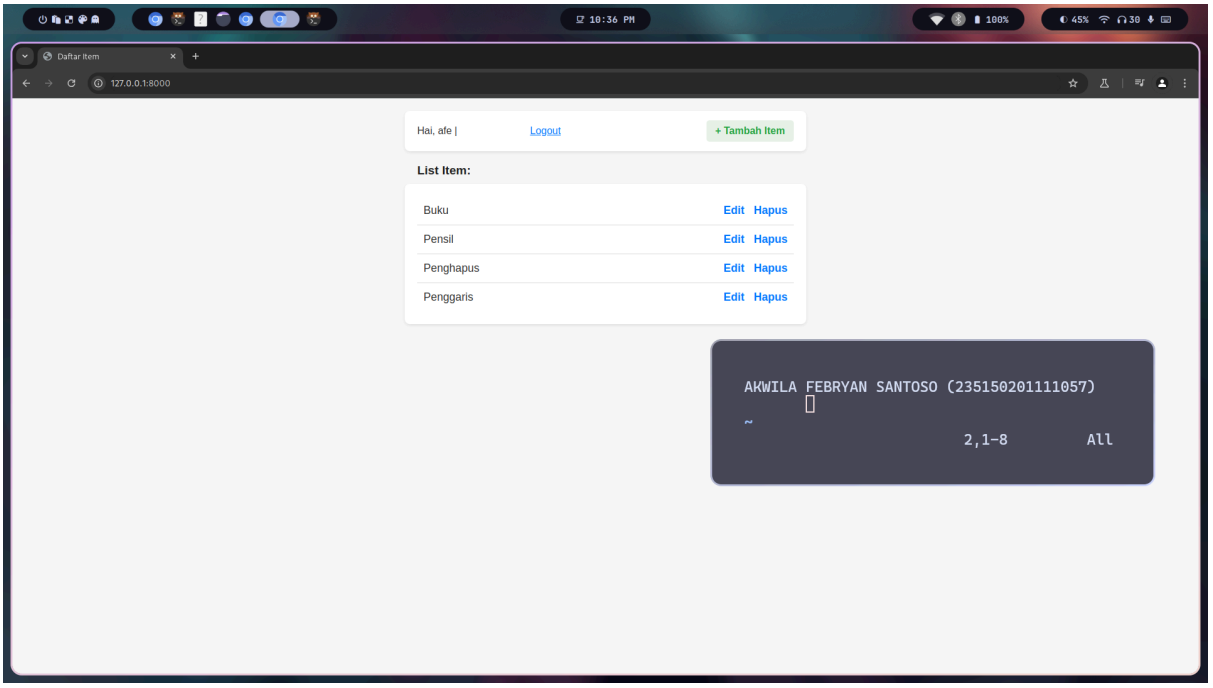


Gambar Halaman List Item Ketika Pengguna Sudah Login

Skenario Menambah Item

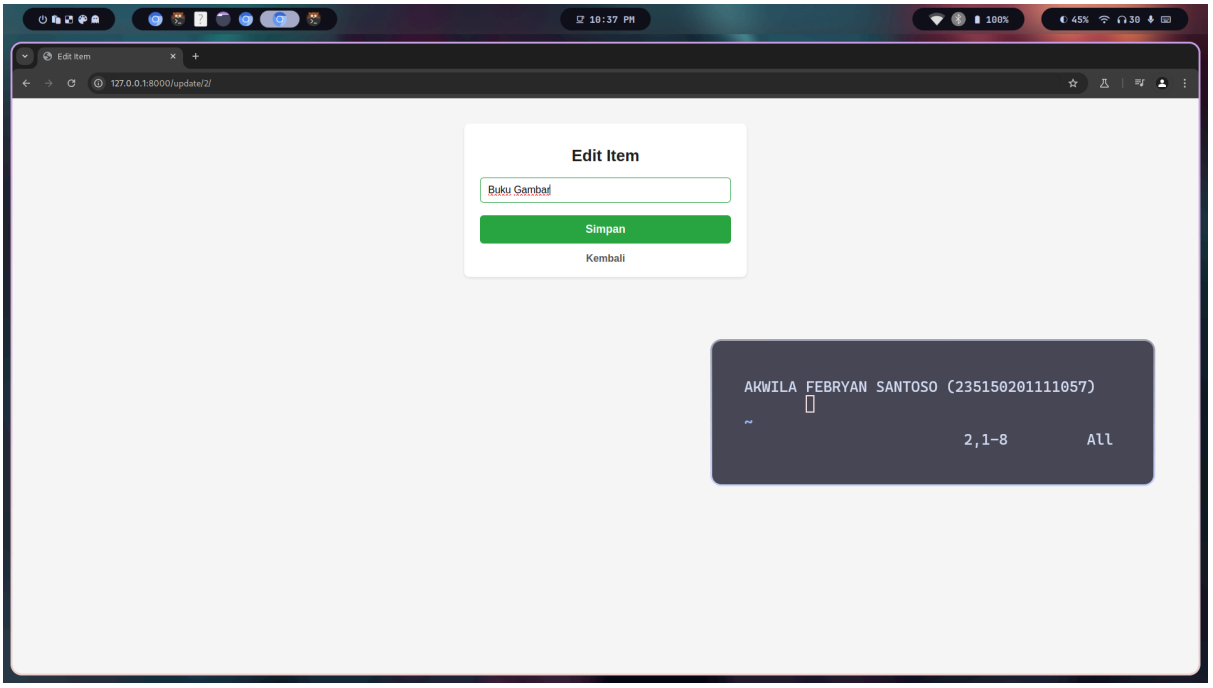


Gambar Halaman Menambah Item (Disini tambah item berupa penggaris)

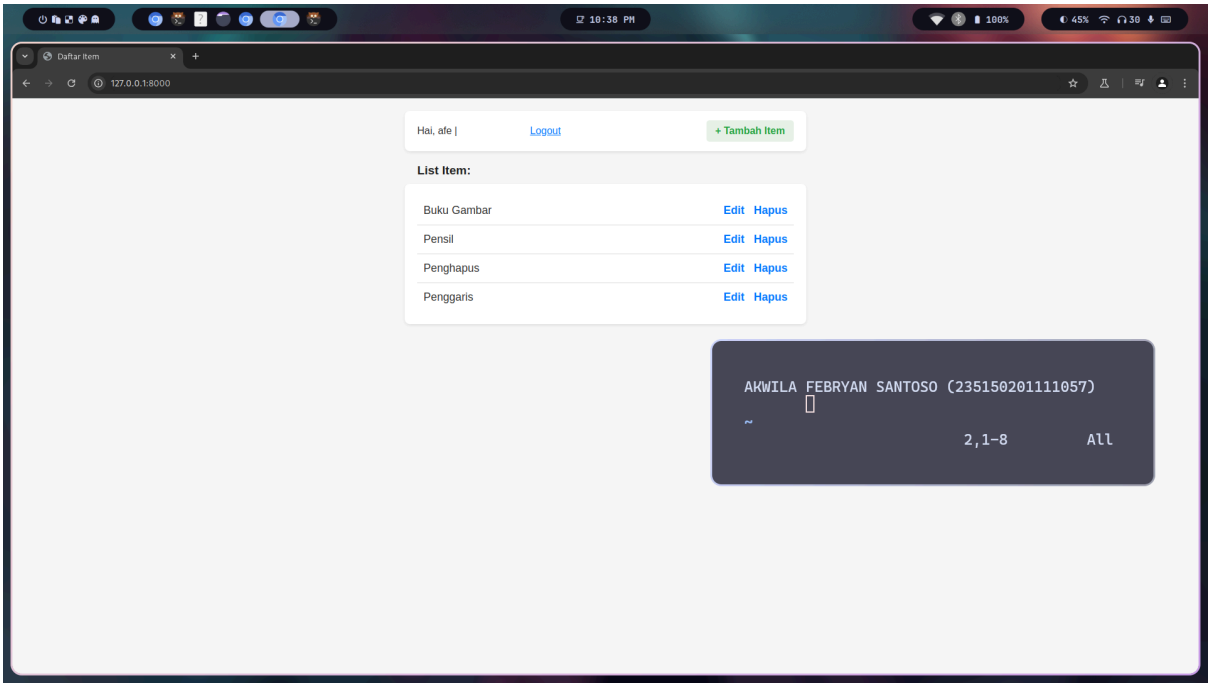


Gambar Halaman List Item Setelah Pengguna Menambah Item

Skenario Mengubah Item

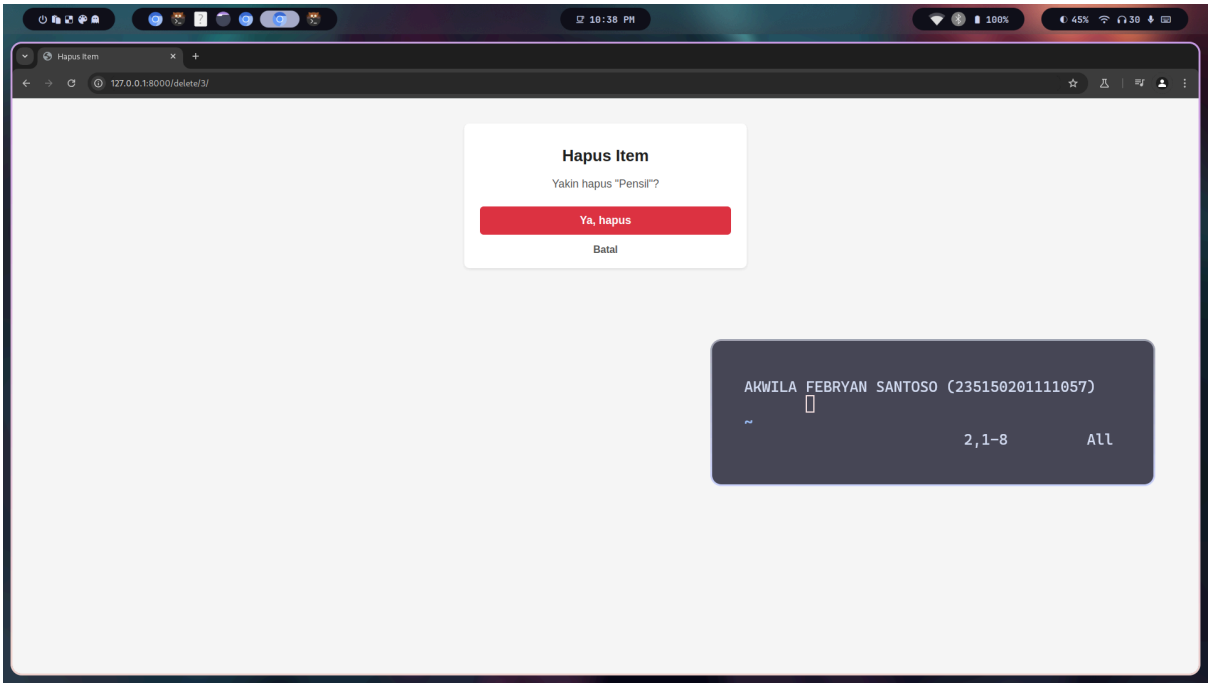


Gambar Halaman Edit Item (Disini kita merubah item buku menjadi buku gambar)

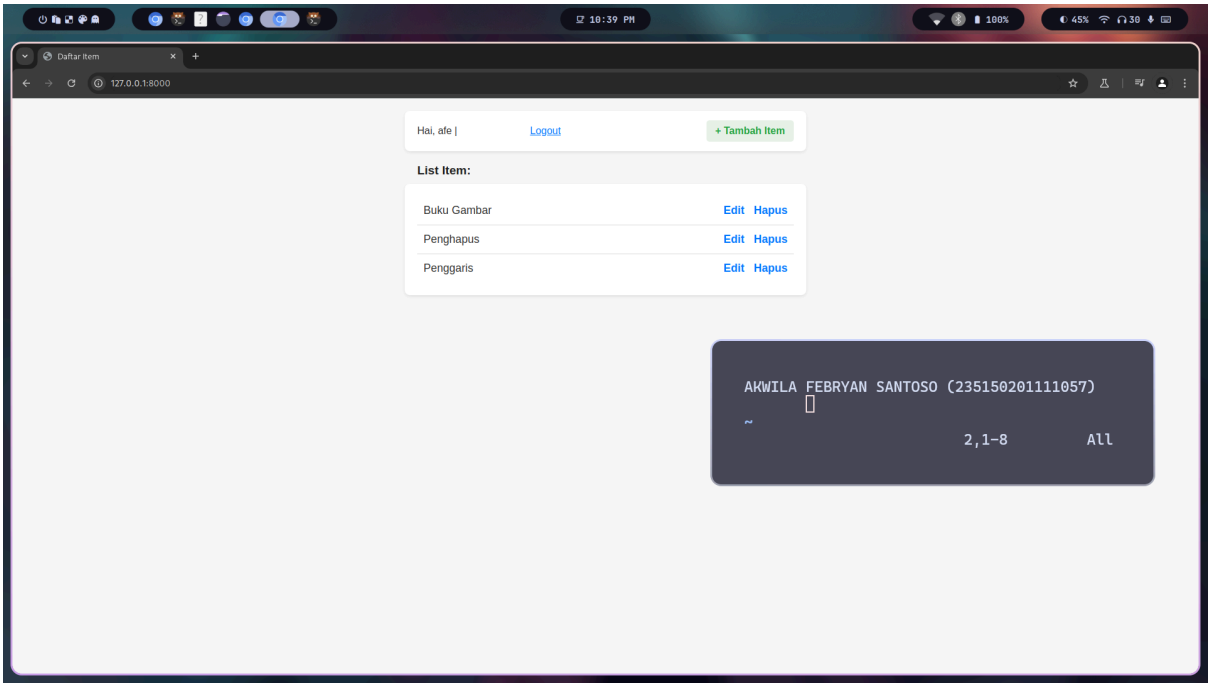


Skenario Menghapus Item

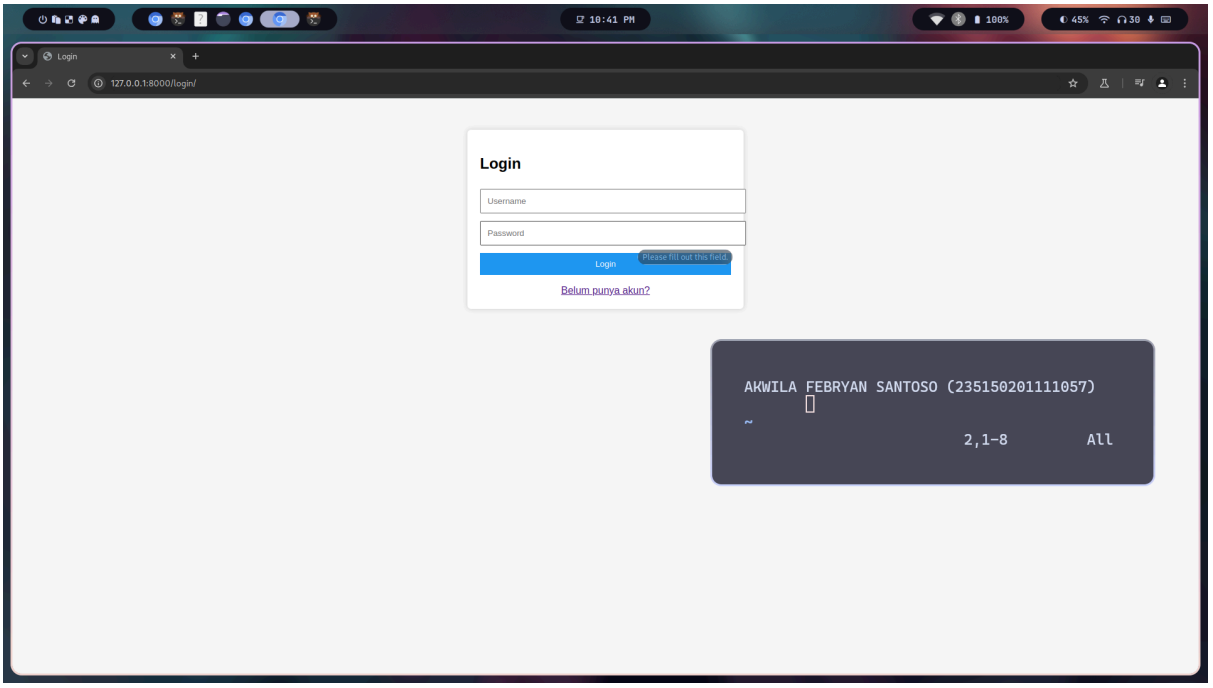
Gambar Halaman List Item Setelah Pengguna Mengubah Item



Gambar Halaman Hapus Item (Disini kita menghapus item pensil)



Gambar Halaman List Item Setelah Pengguna Menghapus Item



Gambar Halaman Login Setelah Pengguna Logout

C. Syntax

Folder templates

Kode login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Login</title>
  <style>
    body { font-family: sans-serif; background: #f7f7f7; }
    .container { max-width: 400px; margin: 50px auto; background: white;
padding: 20px; border-radius: 6px; box-shadow: 0 0 8px #ccc; }
    input, button { width: 100%; padding: 10px; margin: 6px 0; }
    button { background-color: #2196F3; color: white; border: none;
cursor: pointer; }
    a { display: block; margin-top: 10px; text-align: center; }
  </style>
</head>
<body>
  <div class="container">
    <h2>Login</h2>
    <form method="post" action="{% url 'login' %}">
      {% csrf_token %}
      <input type="text" name="username" placeholder="Username" required
/>
      <input type="password" name="password" placeholder="Password"
required />
      <button type="submit">Login</button>
    </form>
    <a href="{% url 'register' %}">Belum punya akun?</a>
  </div>
</body>
</html>
```

Kode register.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Register</title>
  <style>
    body { font-family: sans-serif; background: #f7f7f7; }
    .container { max-width: 400px; margin: 50px auto; background: white;
padding: 20px; border-radius: 6px; box-shadow: 0 0 8px #ccc; }
    input, button { width: 100%; padding: 10px; margin: 6px 0; }
    button { background-color: #4CAF50; color: white; border: none;
cursor: pointer; }
    a { display: block; margin-top: 10px; text-align: center; }
  </style>
</head>
<body>
  <div class="container">
    <h2>Register</h2>
    <form method="post" action="{% url 'register' %}">
      {% csrf_token %}
      <p>
        {{ form.username.label_tag }}
        {{ form.username }}
      </p>
      <p>
        {{ form.password1.label_tag }}
        {{ form.password1 }}
      </p>
      <p>
        {{ form.password2.label_tag }}
        {{ form.password2 }}
      </p>
      <button type="submit">Register</button>
    </form>

    <a href="{% url 'login' %}">Sudah punya akun? Login di sini</a>
    <a href="{% url 'item_list' %}">Lihat daftar item</a>
  </div>
</body>
</html>
```

Kode form.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>{% if item %}Edit Item{% else %}Tambah Item{% endif %}</title>
  <style>
    body {
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      background-color: #f9f9f9;
      margin: 20px;
      color: #333;
    }
    .container {
      max-width: 400px;
      margin: 40px auto;
      background: white;
      padding: 20px 25px;
      border-radius: 8px;
      box-shadow: 0 2px 6px rgba(0,0,0,0.1);
    }
    h1 {
      margin-bottom: 20px;
      font-weight: 700;
      font-size: 1.6rem;
      color: #222;
      text-align: center;
    }
    input[type="text"] {
      width: 100%;
      padding: 10px 12px;
      margin-bottom: 20px;
      border: 1.5px solid #ccc;
      border-radius: 6px;
```

```
      font-size: 1rem;
      box-sizing: border-box;
      transition: border-color 0.3s ease;
    }
    input[type="text"]:focus {
      border-color: #28a745;
      outline: none;
    }
    button {
      width: 100%;
      padding: 12px;
      background-color: #28a745;
      border: none;
      border-radius: 6px;
      color: white;
      font-size: 1.1rem;
      font-weight: 600;
      cursor: pointer;
      transition: background-color 0.3s ease;
    }
    button:hover {
      background-color: #218838;
    }
    .cancel-link {
      display: block;
      margin-top: 15px;
      text-align: center;
      color: #555;
      text-decoration: none;
      font-weight: 600;
      transition: color 0.3s ease;
    }
    .cancel-link:hover {
      color: #000;
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>{% if item %}Edit Item{% else %}Tambah Item{% endif %}</h1>
    <form method="post">
      {% csrf_token %}
      <input type="text" name="name" value="{{
item.name|default_if_none:'' }}" placeholder="Nama item" required />
      <button type="submit">Simpan</button>
    </form>
    <a href="{% url 'item_list' %}" class="cancel-link">Kembali</a>
  </div>
</body>
</html>
```

Kode list.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Daftar Item</title>
  <style>
    body {
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      background-color: #f9f9f9;
      margin: 20px;
      color: #333;
    }
    .header {
      max-width: 600px;
      margin: 0 auto 20px auto;
      padding: 15px 20px;
      background-color: #fff;
      border-radius: 8px;
      box-shadow: 0 2px 6px rgba(0,0,0,0.1);
      display: flex;
      justify-content: space-between;
      align-items: center;
    }
```

```
    .header p {
      margin: 0;
      font-size: 1rem;
    }
    .header form button {
      background: none;
      border: none;
      color: #007bff;
      cursor: pointer;
      font-size: 1rem;
      padding: 0;
      font-family: inherit;
      text-decoration: underline;
    }
    .header a {
      text-decoration: none;
      color: #28a745;
      font-weight: 600;
      padding: 8px 14px;
      border-radius: 5px;
      background-color: #e6f4ea;
      transition: background-color 0.3s ease;
    }
    .header a:hover {
      background-color: #c7e6cd;
    }
    .auth-links a {
      margin-right: 15px;
      color: #007bff;
      text-decoration: none;
      font-weight: 600;
    }
    .auth-links a:hover {
      text-decoration: underline;
    }
    ul.item-list {
      max-width: 600px;
      margin: 0 auto;
      background-color: #fff;
      border-radius: 8px;
      box-shadow: 0 2px 6px rgba(0,0,0,0.1);
      padding: 20px;
      list-style: none;
    }
    ul.item-list li {
      padding: 12px 10px;
      border-bottom: 1px solid #ddd;
      display: flex;
      justify-content: space-between;
      align-items: center;
      font-size: 1.1rem;
    }
    ul.item-list li:last-child {
      border-bottom: none;
    }
    ul.item-list li a {
      margin-left: 10px;
      text-decoration: none;
      color: #007bff;
      font-weight: 600;
    }
    ul.item-list li a:hover {
      text-decoration: underline;
    }



  </style>
</head>
<body>
    <div class="header">
      {% if user.is_authenticated %}
        <p>Hai, {{ user.username }} |
          <form method="post" action="{% url 'logout' %}"
style="display:inline;">
            {% csrf_token %}
            <button type="submit">Logout</button>
          </form>
        </p>
```

```
        <a href="{% url 'item_create' %}">+ Tambah Item</a>
      {% else %}
        <div class="auth-links">
          <a href="{% url 'login' %}">Login</a>
          <a href="{% url 'register' %}">Daftar</a>
        </div>
      {% endif %}
    </div>

    <div style="max-width:600px; margin: 0 auto 10px auto; font-weight:
600; font-size: 1.2rem; color: #222;">
      List Item:
    </div>

    <ul class="item-list">
      {% for item in items %}
        <li>
          <span>{{ item.name }}</span>
          {% if user.is_authenticated %}
            <span>
              <a href="{% url 'item_update' item.pk %}">Edit</a>
              <a href="{% url 'item_delete' item.pk %}">Hapus</a>
            </span>
          {% endif %}
        </li>
      {% empty %}
        <li>Belum ada item yang tersedia.</li>
      {% endfor %}
    </ul>
  </body>

</html>
```

Kode delete.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Hapus Item</title>
  <style>
    body {
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      background-color: #f9f9f9;
      margin: 20px;
      color: #333;
    }
    .container {
      max-width: 400px;
      margin: 40px auto;
      background: white;
      padding: 20px 25px;
      border-radius: 8px;
      box-shadow: 0 2px 6px rgba(0,0,0,0.1);
      text-align: center;
    }
    h1 {
      margin-bottom: 20px;
      font-weight: 700;
      font-size: 1.6rem;
      color: #222;
    }
    p.confirmation-text {
      font-size: 1.1rem;
      margin-bottom: 25px;
      color: #555;
    }
    button {
      width: 100%;
      padding: 12px;
      background-color: #dc3545;
      border: none;
      border-radius: 6px;
      color: white;
      font-size: 1.1rem;
      font-weight: 600;
```

```
      cursor: pointer;
      transition: background-color 0.3s ease;
      margin-bottom: 15px;
    }
    button:hover {
      background-color: #b02a37;
    }
    .cancel-link {
      display: inline-block;
      color: #555;
      text-decoration: none;
      font-weight: 600;
      transition: color 0.3s ease;
      cursor: pointer;
    }
    .cancel-link:hover {
      color: #000;
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>Hapus Item</h1>
    <p class="confirmation-text">Yakin hapus &quot;{{ item.name
}}&quot;?</p>
    <form method="post">
      {% csrf_token %}
      <button type="submit">Ya, hapus</button>
    </form>
    <a href="{% url 'item_list' %}" class="cancel-link">Batal</a>
  </div>
</body>
</html>
```

Folder blog

Kode models.py

```
from django.db import models

class Item(models.Model):
    name = models.CharField(max_length=100)

    def __str__(self):
        return self.name
```

Kode urls.py

```
from django.contrib.auth import views as auth_views
from django.urls import path
from . import views

urlpatterns = [
    path('', views.item_list, name='item_list'),
    path('create/', views.item_create, name='item_create'),
    path('update/<int:pk>/', views.item_update, name='item_update'),
    path('delete/<int:pk>/', views.item_delete, name='item_delete'),

    # Auth
    path('login/',
auth_views.LoginView.as_view(template_name='blog/login.html'),
name='login'),
    path('logout/', auth_views.LogoutView.as_view(next_page='login'),
name='logout'),
    path('register/', views.register, name='register'),
]
```

Kode views.py

```
from django.shortcuts import render, redirect, get_object_or_404
```

```
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
from django.contrib.auth.decorators import login_required
from django.contrib.auth import login as auth_login

from .models import Item

# READ - untuk semua orang
def item_list(request):
    items = Item.objects.all()
    return render(request, 'blog/list.html', {'items': items})


# CREATE
@login_required
def item_create(request):
    if request.method == 'POST':
        name = request.POST.get('name')
        Item.objects.create(name=name)
        return redirect('item_list')
    return render(request, 'blog/form.html')

# UPDATE
@login_required
def item_update(request, pk):
    item = get_object_or_404(Item, pk=pk)
    if request.method == 'POST':
        item.name = request.POST.get('name')
        item.save()
        return redirect('item_list')
    return render(request, 'blog/form.html', {'item': item})

# DELETE
@login_required
def item_delete(request, pk):
    item = get_object_or_404(Item, pk=pk)
    if request.method == 'POST':
        item.delete()
        return redirect('item_list')
    return render(request, 'blog/delete.html', {'item': item})

# REGISTER
def register(request):
    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('login')
    else:
        form = UserCreationForm()
    return render(request, 'blog/register.html', {'form': form})

# LOGIN
def user_login(request):
    if request.method == 'POST':
        form = AuthenticationForm(data=request.POST)
        if form.is_valid():
            user = form.get_user()
            auth_login(request, user)
            return redirect('item_list')
    else:
        form = AuthenticationForm()
    return render(request, 'blog/login.html', {'form': form})

# LOGOUT
def logout_view(request):
    logout(request)
    return redirect('item_list')
```

Folder crud_project

Kode settings.py

```
from pathlib import Path
```

```
# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/5.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY =
'django-insecure-0jxo)9p_$wpcv-+!nz4^#-ocri-g2d+=ihqz55vj^(%r*8a^*k'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'blog',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'crud_project.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'crud_project.wsgi.application'


# Database
# https://docs.djangoproject.com/en/5.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'crud_db',
        'USER': 'django_user',
        'PASSWORD': 'passwordku',
        'HOST': 'localhost',
        'PORT': '3306',
    }
}

LOGIN_URL = '/login/'
LOGIN_REDIRECT_URL = '/'
LOGOUT_REDIRECT_URL = 'login'
```

D. Penjelasan

Pada program diatas saya menggunakan framework django dengan bahasa pemrograman python untuk membuat aplikasi web sederhana yang memiliki fitur CRUD (Create, Read, Update, Delete) untuk mengelola daftar item. Selain itu, saya juga menambahkan fitur autentikasi sehingga pengguna harus login terlebih dahulu untuk dapat menambah, mengedit, atau menghapus item. Jika pengguna belum login mereka hanya bisa melihat daftar item dan diberikan opsi untuk login atau mendaftar akun.

Program tersebut memiliki struktur proyek Django standar yang terdiri dari dua folder utama yaitu crud_project/ sebagai konfigurasi utama proyek dan blog/ sebagai aplikasi inti yang menangani fitur CRUD. File models.py di dalam folder blog berfungsi sebagai bagian Model dalam pola arsitektur MVC (Model-View-Controller) yang merepresentasikan struktur data dari item yang dikelola. File views.py bertindak sebagai Controller yang memproses logika aplikasi dan menentukan data apa yang akan ditampilkan. Sementara itu, folder templates/blog/ berisi file HTML seperti list.html, form.html, dan delete.html yang berperan sebagai View, menampilkan data kepada pengguna. Routing dari URL ditangani oleh urls.py baik di level proyek maupun aplikasi, yang mengarahkan permintaan ke view yang sesuai.

Folder template merupakan agian dari antarmuka pengguna untuk aplikasi web sederhana berbasis Django yang memiliki fitur autentikasi dan manajemen item. Terdapat halaman login dan register yang digunakan untuk mengatur proses masuk dan pendaftaran akun pengguna. Formulir pada kedua halaman ini telah dilengkapi dengan token CSRF untuk keamanan. Setelah pengguna berhasil login, mereka akan diarahkan ke halaman list item, yang menampilkan daftar item yang bisa diedit atau dihapus, namun hanya jika pengguna telah terautentikasi. Tautan untuk logout, login, dan register ditampilkan secara kondisional berdasarkan status autentikasi pengguna.

Selain itu, terdapat halaman form.html yang digunakan untuk menambahkan atau mengedit item, serta halaman delete.html untuk mengonfirmasi penghapusan item tertentu. Semua tampilan ini didesain dengan gaya CSS sederhana yang responsif dan bersih, memanfaatkan HTML dasar dan sintaks Django templating seperti {% csrf_token %}, {% if %}, dan {% url %} untuk menghubungkan logika backend ke antarmuka frontend. Sistem ini memungkinkan pengelolaan data item secara aman dan user-friendly dengan pengalaman pengguna yang terstruktur dan intuitif.

Kemudian terdapat folder blog yang berguna untuk CRUD yang memungkinkan pengguna untuk ntuk mengelola daftar item setelah login. Model Item hanya memiliki satu field name, yang mewakili nama item, dan digunakan sebagai representasi string dari objek tersebut. Routing diatur dalam urls.py, yang menghubungkan berbagai path seperti tampilan daftar item, pembuatan, pembaruan, dan penghapusan item, serta fitur login, logout, dan register. Untuk otentikasi, digunakan LoginView dan LogoutView bawaan Django, sementara registrasi pengguna ditangani melalui fungsi khusus register.

Pada bagian views.py, logika utama aplikasi didefinisikan. Fungsi item_list menampilkan semua item yang ada, sedangkan item_create, item_update, dan item_delete mengatur manipulasi data dan dilindungi oleh dekorator @login_required agar hanya pengguna yang sudah login yang bisa mengaksesnya. Fungsi register memungkinkan pengguna baru untuk membuat akun menggunakan UserCreationForm. Selain itu, disediakan juga fungsi

user_login dan logout_view untuk mengatur proses masuk dan keluar secara manual, meskipun login dan logout sebenarnya sudah ditangani oleh views bawaan di urls.py.

File settings.py ini merupakan pengaturan utama untuk proyek Django bernama crud_project, yang mendukung aplikasi blog. File ini mengatur berbagai konfigurasi penting seperti INSTALLED_APPS, middleware, dan pengaturan template. Salah satu bagian krusial adalah konfigurasi DATABASES, yang menunjukkan bahwa proyek ini menggunakan MySQL sebagai database backend, dengan nama database crud_db, user django_user, dan password passwordku. Pengaturan ini memungkinkan Django menyimpan dan mengelola data Item serta informasi autentikasi pengguna secara permanen di MySQL. Selain itu, konfigurasi LOGIN_URL, LOGIN_REDIRECT_URL, dan LOGOUT_REDIRECT_URL mengatur alur autentikasi pengguna, memastikan bahwa pengguna diarahkan ke halaman login, daftar item, atau login kembali setelah logout.

Terlihat dari hasil screenshot diatas aplikasi telah berjalan dengan baik dan skenario percobaan CRUD yang telah dilakukan menunjukan bahwa sistem berhasil membatasi hak akses pengguna sesuai yang diharapkan. Ketika pengguna belum login, mereka hanya bisa melihat daftar item tanpa dapat menambahkan, mengedit, atau menghapus data. Setelah login, pengguna memperoleh akses penuh terhadap fitur CRUD. Proses penambahan item berjalan dengan baik, dibuktikan dengan munculnya item baru seperti "penggaris" di halaman list setelah di submit melalui form. Ini menunjukkan bahwa proses penyimpanan data ke database berjalan dengan lancar.

Selanjutnya, saat pengguna melakukan perubahan pada item seperti mengganti "buku" menjadi "buku gambar", sistem berhasil memperbarui data dan menampilkannya di list item. Penghapusan item seperti "pensil" juga berhasil dilakukan, dan item tersebut tidak lagi muncul dalam daftar setelah konfirmasi penghapusan. Dari seluruh rangkaian pengujian, bisa disimpulkan bahwa alur CRUD (Create, Read, Update, Delete) berjalan sesuai fungsinya, dan sistem telah berhasil mengelola data dengan baik berdasarkan autentikasi pengguna. Ini menunjukkan bahwa implementasi Django dengan pembatasan login dan integrasi ke database MySQL telah berjalan efektif.

LATIHAN 2

A. Soal

Apa saja kendala yang saudara hadapi saat mengerjakan modul praktikum ini dan

bagaimana saudara bisa mengatasinya?

B. Penjelasan

Pada praktikum kali ini relatif tidak ada kendala ataupun masalah saat pengerjaan modul. Pengerjaan modul berjalan dengan lancar dan kode program berhasil berjalan dengan baik