LAPORAN RESPONSI PRAKTIKUM

PEMROGRAMAN WEBSITE

Disusun oleh :
Awalisani umiyati (242104015)

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK DAN TEKNOLOGI INFORMASI
UNIVERSITAS JENDERAL ACHMAD YANI YOGYAKARTA
2025

Lembar pengesahan

Laporan

Praktikum pengembangan website

Daring

Yang dipersiapkan dan disusun oleh :

Awalisani umiyati
(242104015)

Dinyatakan telah memenuhi syarat untuk mengikuti ujian akhir semester

Praktikum pengembangan website periode 2025/2026

DAFTAR ISI

TUJUAN PEMBELAJARAN

Tujuan umum praktikum ini adalah membangun aplikasi web sistem manajemen laundry professional yang lengkap dan fungsional menggunakan framework flask, dengan menerapkan best practices dalam pengembangan web modern termasuk keamanan, manajemen database relasional, dan user interface yang responsif.

Tujuan khusus:

1. Virtual environment: memahami dan mengimplementasikan penggunaan virtual environment python untuk mengisolasi dependency proyek, sehingga tidak ada konflik dengan package sistem atau proyek lain. Praktikum ini melatih kita untuk konsisten menggunakan venv dalam setiap pengembangan aplikasi python.
2. Database relasional: merancang dan mengimplementasikan struktur database relasional yang efisien dengan minimal 4 tabel yang saling berelasi (roles, users, laundry_services, service_orders). Kita juga belajar menggunakan foreign key untuk menjaga integritas referensial data, dan memahami perbedaan penggunaan sqlite untuk development dan mysql untuk production.
3. Operasi crud: mengimplementasikan operasi create, read, update, dan delete pada entitas layanan laundry dan pesanan pelanggan. Setiap operasi harus berjalan lancar tanpa error, dilengkapi dengan validasi input dan notifikasi feedback ke user.
4. Autentikasi: membangun sistem login dan logout yang aman menggunakan flask-login, dengan password hashing menggunakan werkzeug. Sistem harus bisa membedakan user yang sudah login dan yang belum, serta memberikan feedback yang jelas saat login berhasil atau gagal.
5. Otorisasi: mengimplementasikan role-based access control (rbac) dengan dua level akses: admin/karyawan dan user/costumer. Admin/ kayawan memiliki akses penuh untuk mengelola layanan dan melihat semua pesanan, sementara user hanya bisa membuat dan melihat pesanannya sendiri. Sistem harus bisa melakukan redirect otomatis jika ada user yang coba akses halaman yang bukan haknya.
6. Tampilan web: membuat user interface yang responsif dan menarik menggunakan bootstrap 5.3, yang bisa beradaptasi dengan baik di berbagai ukuran layar (mobile, tablet, desktop). Tampilan harus konsisten, mudah digunakan, dan original (bukan copy-paste template mentah).

Demo aplikasi
login menggunakan 2 user customer/user dan admin/karyawan

1. User costumer

Name : miyas
Password : 1234567

2. User admin/karyawan

Name : karyawan
Password : karyawan123

Link website yang sudah di devloy
Responsi-prakweb-production.up.railway.app

Link git hub
https://github.com/awalisane/responsi-prakweb.git

BAB I

PENDAHULUAN

1.1 latar belakang

Di era digital seperti sekarang, bisnis laundry yang masih mengandalkan pencatatan manual menghadapi berbagai kendala serius. Bayangkan saja, pemilik laundry harus mencatat semua transaksi di buku besar, menghitung manual berapa banyak pelanggan yang datang, layanan apa saja yang paling laku, sampai harus ngecek satu-satu nota untuk tahu mana pesanan yang sudah selesai atau belum. Belum lagi kalau notanya hilang atau tulisannya gak kebaca, waduh ribet banget kan?

Masalah lainnya, data pelanggan dan transaksi yang tersimpan di kertas sangat rentan hilang atau rusak. Mau cari data transaksi bulan lalu? Harus bongkar-bongkar tumpukan nota. Mau bikin laporan penjualan? Hitung manual satu-satu pakai kalkulator. Ini jelas tidak efisien dan membuang-buang waktu yang bisa dipakai untuk fokus ke pelayanan pelanggan.

Oleh karena itu dalam praktikum ini kita bikin solusinya pakai aplikasi web berbasis flask. Kenapa flask? Karena flask itu framework python yang simpel tapi powerful. Cocok banget buat bikin aplikasi web yang skalanya menengah seperti sistem laundry ini. Flask juga fleksibel, jadi kita bisa custom sesuai kebutuhan tanpa ribet kayak framework besar lainnya.

Yang tidak kalah penting adalah soal keamanan data. Sistem laundry ini kan nyimpan data sensitif seperti informasi pelanggan, detail pesanan, dan data transaksi keuangan. Makanya kita implementasikan sistem autentikasi yang ketat dengan password hashing, jadi password pengguna tidak disimpan dalam bentuk plain text yang gampang bocor. Terus ada juga sistem otorisasi berbasis role, di mana admin/ karyawan punya akses penuh ke semua fitur, sementara user biasa cuma bisa akses fitur-fitur tertentu aja. Ini penting banget buat menjaga integritas data dan mencegah akses yang tidak sah.

1.2 alasan penggunaan framework flask

Flask dipilih sebagai framework dalam praktikum ini dengan pertimbangan sebagai berikut:

Pertama, flask merupakan micro-framework yang ringan dan mudah dipelajari, sehingga sangat cocok untuk tujuan pembelajaran. Filosofi "micro" dalam flask tidak berarti framework ini memiliki fungsionalitas yang terbatas, melainkan memberikan fleksibilitas kepada developer untuk memilih komponen yang benar-benar dibutuhkan.

Kedua, flask memiliki dokumentasi yang lengkap dan komunitas yang besar, memudahkan proses pembelajaran dan penyelesaian masalah yang mungkin dihadapi selama pengembangan.

Ketiga, flask menggunakan bahasa pemrograman python yang memiliki sintaks yang bersih dan mudah dibaca, sehingga memungkinkan fokus pada konsep pengembangan web tanpa terbebani oleh kompleksitas sintaks bahasa pemrograman.

Keempat, flask mendukung integrasi dengan berbagai library dan ekstensi seperti flask-sqlalchemy untuk manajemen database, flask-login untuk autentikasi, dan flask-wtf untuk pengelolaan form, yang semuanya diperlukan dalam praktikum ini.

1.3 gambaran umum aplikasi

Aplikasi yang dikembangkan dalam praktikum ini adalah website manajemen laundry. Sistem ini dirancang untuk mengelola data layanan jasa laundry dan pemesanan jasa dengan dua tingkat akses pengguna yaitu karyawan/admin dan user/costumer

admin memiliki hak akses penuh untuk mengelola data layanan (menambah, mengubah, dan menghapus), mengelola data member, serta melihat seluruh riwayat penggunaan jasa. Sedangkan costumer memiliki hak akses untuk melihat katalog layanan, melakukan pemesanan layanan jasa, dan melihat riwayat pesanan(transaksi) pribadi mereka.

sistem ini mengimplementasikan database relasional dengan empat tabel yang saling Berelasi, yaitu tabel roles, users, laundry_services, dan service_orders. Relasi antar Tabel memastikan integritas data dan memungkinkan pelacakan transaksi pesanan laundry secara akurat.

1.4 manfaat praktikum

Manfaat akademik:

Dari sisi akademik, praktikum ini memberikan pengalaman hands-on yang sangat berharga dalam mengintegrasikan berbagai konsep yang selama ini kita pelajari secara terpisah. Kita jadi paham bagaimana teori database relasional, pemrograman web, dan keamanan sistem informasi itu diterapkan secara bersamaan dalam satu proyek.

Praktikum ini juga melatih kemampuan problem-solving kita. Misalnya saat debugging error, merancang struktur database yang optimal, atau memutuskan flow aplikasi yang paling user-friendly.

Manfaat praktis:

Secara praktis, sistem yang kita bangun ini bisa langsung digunakan oleh bisnis laundry real. Pemilik laundry bisa mengelola berbagai jenis layanan (cuci kering, cuci setrika, dry cleaning, dll) dengan mudah. Mereka bisa menambah layanan baru, update harga, atau hapus layanan yang sudah tidak tersedia.

Untuk pelanggan atau customer laundry, mereka bisa langsung bikin pesanan baru lewat sistem, lihat status pesanan, dan mengelola pesanannya dengan lebih terorganisir. Semua data tersimpan aman di database, jadi tidak khawatir kehilangan data transaksi.

Manfaat pengembangan sistem:

Dari perspektif pengembangan sistem, proyek ini mengajarkan kita untuk berpikir skalabel dan maintainable. Struktur kode yang kita buat menggunakan pattern yang terorganisir: ada folder untuk models, views, templates, static files, semuanya terpisah dengan jelas. Ini penting banget supaya kalau nanti sistem mau dikembangkan lebih lanjut, kita atau developer lain yang melanjutkan tidak bingung.

Sistem ini juga sudah menerapkan prinsip separation of concerns. Database logic terpisah dari business logic, business logic terpisah dari presentation logic. Jadi kalau suatu saat mau ganti database dari sqllite ke mysql misalnya, kita tidak perlu ubah keseluruhan kode, cukup bagian konfigurasi database aja.

Penggunaan virtual environment juga mengajarkan kita best practice dalam manajemen dependency. Ini sangat penting saat kerja dalam tim atau saat deploy aplikasi ke production. Semua dependency yang diperlukan sudah terdokumentasi dengan jelas di requirements.txt, jadi siapapun yang mau run atau develop sistem ini bisa dengan mudah setup environment yang sama persis.

Terakhir, dengan mengimplementasikan autentikasi dan otorisasi yang proper, kita belajar bagaimana membangun sistem yang secure by design. Ini bukan cuma sekedar add-on atau fitur tambahan, tapi sudah built-in dari awal dalam arsitektur sistem. Skill ini sangat valuable karena security adalah aspek yang paling krusial dalam aplikasi web modern.

Dalam sistem ini, terdapat dua tingkat pengguna (role) yang dibedakan berdasarkan hak aksesnya:

1. Admin/karyawan: istilah "admin" dan "karyawan" merujuk pada role yang sama dalam sistem, yaitu pengguna dengan hak akses penuh untuk mengelola data layanan, memantau seluruh pesanan dari semua pelanggan, dan melakukan konfigurasi sistem. Role ini ditujukan untuk pemilik laundry, manajer, atau staf operasional yang bertanggung jawab mengelola bisnis.
2. User/customer: istilah "user", "customer", dan "member" merujuk pada role yang sama, yaitu pelanggan atau pengguna umum yang menggunakan layanan laundry. Role ini memiliki hak akses terbatas untuk membuat pesanan, melihat riwayat pesanan pribadi, dan mengelola profil mereka sendiri.

Untuk konsistensi dalam laporan ini, selanjutnya akan digunakan istilah "admin maupum administrator" untuk merujuk pada pengelola sistem(karyawan) dan "user" untuk merujuk pada pelanggan(costumer)

BAB II

LANDASAN TEORI

2.1 virtual environment python

Pengertian virtual environment

Virtual environment adalah sebuah lingkungan python yang terisolasi dan independen, yang memungkinkan kita untuk menginstal package dan dependency tertentu tanpa mempengaruhi instalasi python global di sistem operasi. Bayangin virtual environment ini seperti "kardus" tersendiri untuk setiap proyek python yang kita buat.

Misalnya gini, kita punya proyek a yang butuh flask versi 2.0, sementara proyek b butuh flask versi 3.0. Kalau kita install flask langsung ke sistem tanpa virtual environment, pasti konflik kan? Mau install yang versi 2.0 atau 3.0? dengan virtual environment, proyek a punya "kardus" sendiri dengan flask 2.0, dan proyek b punya "kardus" sendiri dengan flask 3.0. Mereka tidak saling ganggu.

Fungsi dan manfaat

Fungsi utama virtual environment adalah isolasi dependency. Setiap proyek punya set dependency-nya sendiri yang tidak akan bentrok dengan proyek lain atau dengan package sistem. Ini sangat penting terutama saat kita handle banyak proyek dengan requirement yang berbeda-beda.

Manfaat lainnya adalah reproducibility. Dengan virtual environment, kita bisa dengan mudah mendokumentasikan semua dependency yang dipakai dalam proyek lewat file requirements.txt. Jadi kalau ada orang lain atau tim yang mau develop atau run proyek yang sama, mereka tinggal bikin virtual environment baru dan install semua package dari requirements.txt. Environment mereka akan sama persis dengan environment kita, jadi tidak ada drama "di komputer gue jalan kok" tapi di komputer orang lain error.

Virtual environment juga menjaga kebersihan sistem. Package-package yang kita install untuk eksperimen atau project temporary tidak akan numpuk di instalasi python global. Kalau project-nya udah selesai atau tidak kepake lagi, tinggal delete folder virtual environment-nya, beres. Sistem python global tetap bersih.

Dari sisi keamanan, virtual environment juga membantu kita menghindari accidentally menggunakan package dengan versi yang vulnerable. Karena setiap project punya environment tersendiri, kita bisa lebih teliti mengontrol versi package yang dipake.

Alasan penggunaan venv pada proyek flask

Untuk proyek flask ini, penggunaan venv (module bawaan python untuk membuat virtual environment) itu bukan cuma recommended, tapi memang sudah jadi standar best practice. Flask sendiri dan berbagai extension-nya seperti flask-login, flask-sqlalchemy sering update dan kadang ada breaking changes antar versi. Dengan venv, kita bisa lock versi yang kita pakai dan memastikan aplikasi tetap stabil.

Proyek ini juga menggunakan beberapa library pihak ketiga seperti pymysql untuk koneksi mysql, werkzeug untuk password hashing, dan lain-lain. Semua library ini punya dependency mereka sendiri lagi. Tanpa virtual environment, bisa-bisa instalasi kita jadi bloated dengan puluhan package yang sebenernya tidak semua diperlukan.

Dalam konteks deployment, virtual environment memudahkan kita untuk setup environment yang identik antara development, staging, dan production. Server production bisa setup environment yang sama persis dengan apa yang kita pakai di development, mengurangi risiko "works on my machine" syndrome yang sering bikin frustasi developer.

2.2 framework flask

Konsep mvc pada flask

Flask mengadopsi pola arsitektur yang mirip dengan model-view-controller (mvc), meskipun implementasinya lebih fleksibel dibanding framework seperti django yang strict mvc. Mari kita bahas satu-satu:

Model adalah representasi data dan business logic. Di proyek kita, model ini didefinisikan sebagai class python yang merepresentasikan tabel database. Misalnya class user merepresentasikan tabel users, class laundryservice merepresentasikan tabel laundry_services, dan seterusnya. Model tidak cuma struktur data doang, tapi juga method-method untuk manipulasi data, validasi, dan relationship dengan model lain.

Controll dalam flask sebenernya adalah function atau class yang handle request dari user dan mengembalikan response. Ini beda dengan konsep view di framework lain yang biasanya lebih ke presentation layer. Di flask, view ini ada di route function kita. Misalnya saat user akses /dashboard, view function dashboard() akan dieksekusi, ngambil data dari model, dan nge-render template yang sesuai.

View adalah file html yang menentukan bagaimana data ditampilkan ke user. Flask menggunakan jinja2 sebagai template engine, yang memungkinkan kita untuk memasukkan logika sederhana dan variable python ke dalam html. Jadi template bukan cuma static html, tapi bisa dynamic tergantung data yang dikirim dari view.

Keunggulan pola mvc ini adalah separation of concerns yang jelas. Developer yang handle database bisa fokus ke model, yang handle business logic fokus ke controll, dan yang handle tampilan fokus ke view. Mereka bisa kerja paralel tanpa stepping on each other's toes.

Blueprint

Blueprint adalah fitur flask yang memungkinkan kita untuk modularisasi aplikasi. Bayangin aplikasi kita ini sebagai sebuah gedung besar, dan blueprint adalah masing-masing lantai atau wing yang punya fungsi terpisah.

Misalnya di proyek kita, kita bisa bikin blueprint untuk autentikasi (handle login/logout), blueprint untuk dashboard, blueprint untuk management layanan, blueprint untuk

pesanan, dan seterusnya. Setiap blueprint punya route-nya sendiri, template-nya sendiri, bahkan static files-nya sendiri kalau perlu.

Keuntungan utama menggunakan blueprint adalah maintainability. Kode jadi lebih organized dan mudah dikelola. Kalau ada bug di fitur pesanan, kita tau harus cari di blueprint orders. Mau nambahin fitur baru? Tinggal bikin blueprint baru tanpa ganggu blueprint yang sudah ada.

Blueprint juga memudahkan code reuse. Misalnya kita udah bikin blueprint autentikasi yang bagus untuk proyek ini, nanti blueprint yang sama bisa kita pakai lagi di proyek lain dengan sedikit modifikasi.

Routing dan template engine (jinja2)

Routing adalah mekanisme bagaimana flask memetakan url ke function yang harus dieksekusi. Di flask, kita menggunakan decorator @app.route() atau @blueprint.route() untuk mendefinisikan route.

Contohnya gini:

```
@app.route('/services')

Def list_services():

   # fungsi ini akan dieksekusi saat user akses /services
```

Flask mendukung dynamic routing juga, di mana kita bisa capture parameter dari url. Misalnya:

```
@app.route('/service/<int:service_id>')

Def view_service(service_id):

   # service_id adalah variable yang di-capture dari url
```

Routing juga bisa specify http method apa yang diterima. Misalnya untuk form submission biasanya pakai post:

```
@app.route('/service/add', methods=['get', 'post'])
Def add_service():
   if request.method == 'post':
      # handle form submission
   else:
      # show form
```

Setelah view function selesai process data, biasanya kita mau render html untuk ditampilkan ke user. Di sinilah jinja2 berperan. Jinja2 adalah template engine yang powerful dan aman.

Dengan jinja2, kita bisa:

- Menampilkan variable python di html: {{ user.username }}
- Menggunakan control structure seperti if-else dan loop:

```
{% if user.is_authenticated %}
  <p>welcome {{ user.username }}!</p>
{% else %}
  <p>please login</p>
{% endif %}
```

Inheritance template untuk menghindari duplikasi:

```
{% extends "base.html" %}
{% block content %}
  <!-- konten spesifik halaman -->
{% endblock %}
```

- Filter untuk manipulasi data: {{ price|format_currency }}

- Include template lain: {% include 'navbar.html' %}

Jinja2 juga otomatis escape html untuk mencegah xss (cross-site scripting) attack, jadi lebih aman.

2.3 Database relasional

Konsep database relasional

Database relasional adalah jenis database yang menyimpan data dalam bentuk tabel-tabel yang saling berelasi. Konsep dasarnya adalah data yang saling berhubungan disimpan terpisah di tabel berbeda, lalu dihubungkan melalui key.

Kebayang kalau semua data disimpan dalam satu tabel gede? Misal tabel pesanan yang nyimpen juga semua detail layanan, detail customer, semuanya dalam satu baris. Ini namanya redundancy, data yang sama tersimpan berkali-kali. Selain boros storage, ini juga bikin update data jadi nightmare. Bayangkan kalau mau update harga layanan "cuci setrika", harus update di ratusan atau ribuan baris pesanan yang pakai layanan itu.

Solusinya adalah normalisasi: pisahkan data ke dalam tabel-tabel yang logical. Tabel layanan nyimpen info layanan, tabel pesanan nyimpin info pesanan, lalu hubungkan mereka dengan foreign key. Jadinya kalau mau update harga layanan, cukup update satu baris di tabel layanan, dan semua pesanan yang reference ke layanan itu otomatis kebaca harga barunya (tergantung design-nya).

Database relasional juga memastikan data integrity melalui constraints seperti primary key, foreign key, unique, not null, dan lain-lain. Ini mencegah data yang invalid masuk ke database.

Primary key dan foreign key

Primary key (pk) adalah kolom atau kombinasi kolom yang uniquely identify setiap baris dalam tabel. Pk harus unique dan tidak boleh null. Biasanya kita pakai auto-increment integer sebagai pk karena simple dan efisien.

Di proyek ini:

- Tabel roles punya pk role_id
- Tabel users punya pk user_id
- Tabel laundry_services punya pk service_id
- Tabel service_orders punya pk order_id

Foreign key (fk) adalah kolom di satu tabel yang reference ke primary key di tabel lain. Fk inilah yang menciptakan relationship antar tabel.

Di proyek ini:

- Users.role_id adalah fk yang reference ke roles.role_id
- Service_orders.user_id adalah fk yang reference ke users.user_id
- Service_orders.service_id adalah fk yang reference ke laundry_services.service_id

Dengan fk, database bisa enforce referential integrity. Misalnya, kita tidak bisa insert user dengan role_id=5 kalau di tabel roles tidak ada role dengan id 5. Database akan reject operasi ini. Begitu juga kita tidak bisa delete role yang masih di-reference oleh user. Ini mencegah data jadi inconsistent.

One-to-many relationship

One-to-many adalah tipe relationship paling umum dalam database relasional. Konsepnya: satu record di tabel a bisa punya banyak record terkait di tabel b, tapi satu record di tabel b cuma bisa punya satu record terkait di tabel a.

Contoh di proyek kita:

1. Role to users (one-to-many):
    - Satu role (misal "admin") bisa punya banyak user
    - Tapi satu user cuma bisa punya satu role
    - Fk ada di tabel users (many side)
2. User to orders (one-to-many):
    - Satu user bisa punya banyak pesanan
    - Tapi satu pesanan cuma dimiliki oleh satu user
    - Fk ada di tabel service_orders (many side)
3. Service to orders (one-to-many):
    - Satu layanan (misal "cuci kering") bisa ada di banyak pesanan

- Tapi satu pesanan cuma untuk satu jenis layanan
- Fk ada di tabel service_orders (many side)

Cara implementasinya: fk selalu di sisi "many". Jadi tabel service_orders (many) punya fk user_id dan service_id yang point ke tabel users (one) dan laundry_services (one).

Untuk proyek miya laundry ini, saya memutuskan menggunakan sqlite sebagai database utamanya. Ada beberapa alasan kenapa database ini yang paling pas buat praktikum kali ini:

- tidak ribet (zero configuration): keunggulan utama sqlite itu kita tidak perlu install server database yang berat-berat kayak mysql atau postgresql. Sekali install library-nya di python, database langsung bisa jalan.
- File-nya ringkas: database sqlite itu cuma berupa satu file tunggal (di sini namanya miya_laundry_database.db).
- Cocok buat tahap belajar: karena datanya disimpan secara lokal, saya bisa cek isi tabelnya kapan aja lewat vs code pakai plugin sqlite viewer. Ini ngebantu banget pas lagi debugging atau mastiin data layanan laundry-nya masuk atau tidak.
- Ringan banget: sqlite tidak makan banyak ram di laptop.

2.4 operasi crud

Konsep create, read, update, delete

Crud adalah akronim dari empat operasi dasar dalam manajemen data: create, read, update, dan delete. Hampir semua aplikasi yang berurusan dengan data pasti implement keempat operasi ini.

Create (insert): operasi menambahkan data baru ke database. Misalnya saat user mendaftar (insert data user baru), menambah layanan baru (insert data layanan), atau membuat pesanan baru (insert data order).

Proses create biasanya melibatkan:

- User mengisi form dengan data yang diperlukan
- Aplikasi melakukan validasi input (cek format, required field, dll)
- Jika valid, data di-insert ke database
- User diberi feedback bahwa data berhasil disimpan

Read (select): operasi membaca/mengambil data dari database. Ini operasi yang paling sering dilakukan. Setiap kali user buka halaman dan lihat data, itu adalah operasi read.

Read bisa macam-macam bentuknya:

- Read semua data (select * from ...)
- Read data tertentu dengan kondisi (select ... Where ...)
- Read dengan join multiple tables
- Read dengan sorting, filtering, pagination

Update (update): operasi mengubah data yang sudah ada. Misalnya user mau update profile, admin mau update harga layanan, atau update status pesanan dari "diproses" jadi "selesai".

Proses update melibatkan:

- Tampilkan form yang sudah terisi dengan data existing
- User modify data yang mau diubah
- Validasi input baru
- Update data di database berdasarkan id
- Feedback ke user

Delete (delete): operasi menghapus data dari database. Ini operasi yang paling "berbahaya" karena biasanya permanent (kecuali ada soft delete mechanism).

Delete harus dilakukan dengan hati-hati:

- Biasanya ada konfirmasi "are you sure?"
- Check dulu apakah data ini di-reference oleh data lain.
- Delete data dari database
- Feedback ke user

Implementasi crud pada aplikasi web

Dalam aplikasi web, crud biasanya di-map ke http methods dan url patterns:

- Create: post request ke url seperti /service/add
- Read: get request ke url seperti /services (list) atau /service/<id> (detail)
- Update: post/put request ke url seperti /service/<id>/edit
- Delete: post/delete request ke url seperti /service/<id>/delete

Kenapa kebanyakan pakai post? Karena html form by default cuma support get dan post. Untuk put dan delete, biasanya perlu javascript.

Flow typical crud di web app:

1. Create:
    - Get /service/add: tampilkan form kosong
    - Post /service/add: process form submission, insert data, redirect ke list
2. Read:
    - Get /services: query database, tampilkan list
    - Get /service/123: query data dengan id=123, tampilkan detail
3. Update:
    - Get /service/123/edit: query data dengan id=123, tampilkan form terisi
    - Post /service/123/edit: process form, update database, redirect
4. Delete:
    - Post /service/123/delete: delete data dengan id=123, redirect ke list

Dalam implementasi, kita juga perlu:

- Validasi: pastikan input valid sebelum ke database
- Error handling: handle kasus error dengan graceful
- Authorization: pastikan user punya hak untuk perform operasi tersebut
- Feedback: kasih tau user operasi berhasil atau gagal dengan message yang jelas
- Transaction: untuk operasi kompleks, wrap dalam transaction supaya atomic

2.5 autentikasi

Konsep login dan logout

Autentikasi adalah proses verifikasi identitas user. Intinya: membuktikan bahwa "kamu adalah kamu". Di aplikasi web, ini biasanya dilakukan melalui mekanisme login dengan username/email dan password.

Proses login secara detail:

1. User mengisi form login dengan username dan password
2. Aplikasi menerima data dan mengambil user record dari database berdasarkan username
3. Password yang diinput di-hash dengan algoritma yang sama seperti saat register
4. Hash hasil compare dengan hash yang tersimpan di database
5. Jika cocok: login berhasil, buat session untuk user
6. Jika tidak cocok: login gagal, tampilkan error message
7. Setelah login, user punya session yang identify mereka di request-request berikutnya

Session ini biasanya disimpan di cookie di browser user. Setiap request berikutnya akan include cookie ini, dan server bisa tau "oh ini user a yang udah login".

Proses logout:

1. User klik tombol logout
2. Aplikasi destroy session user
3. Redirect ke halaman login atau home
4. User sekarang kembali ke state "not authenticated"

Konsep penting dalam autentikasi adalah stateful session. Http adalah stateless protocol, artinya server tidak "ingat" siapa yang request. Session membuat aplikasi kita jadi stateful: server "remember" user yang sudah login dengan menyimpan session data (di-server side atau via encrypted cookie).

2.6 Password hashing

Password hashing adalah proses mengubah password plain text menjadi string random yang irreversible. Ini prinsip keamanan fundamental dalam aplikasi modern.

Kenapa password harus di-hash?

1. Keamanan data: kalau database kita ke-hack atau leak, password user tetap aman karena yang tersimpan bukan password asli tapi hash-nya. Hacker tidak bisa langsung pakai hash untuk login karena hash tidak bisa di-reverse jadi password asli.
2. Privacy: bahkan developer atau dba yang punya akses ke database tidak bisa tau password user. Ini penting untuk trust dan privacy.
3. Compliance: banyak regulation (seperti gdpr) yang require password disimpan dengan proper encryption/hashing.

Karakteristik hash yang baik:

- One-way function: tidak bisa di-reverse dari hash ke password asli
- Deterministic: password yang sama akan selalu menghasilkan hash yang sama
- Fast to compute: hashing harus cepat supaya tidak impact performance
- Slow enough: tapi tidak terlalu cepat sampai memudahkan brute force attack

Contoh:

- Password: "laundry123"
- Hash: "2b$12 k9vc8ht5.lq3dpx7yyxn6ew9...."
- Kalau user input "laundry123" lagi, akan generate hash yang sama
- Tapi tidak ada cara untuk reverse hash itu jadi "laundry123"

Keamanan password menggunakan werkzeug

Werkzeug adalah library python yang menyediakan utility untuk web development, termasuk secure password hashing. Flask sendiri internally menggunakan werkzeug.

Werkzeug menggunakan pbkdf2 (password-based key derivation function 2) dengan sha-256 untuk hashing. Ini adalah algoritma yang sudah battle-tested dan recommended.

Keunggulan werkzeug password hashing:

1. Salting: werkzeug otomatis menambahkan "salt" (random string) ke password sebelum di-hash. Salt ini berbeda untuk setiap password, sehingga dua user dengan password yang sama akan punya hash yang berbeda. Ini mencegah rainbow table attack.
2. Configurable iterations: pbkdf2 melakukan hashing berkali-kali (default 260,000 iterations). Ini membuat brute force attack jadi sangat lambat dan impractical.
3. Easy to use: api-nya simple

```
From werkzeug.security import generate_password_hash, check_password_hash

# saat register

Hashed = generate_password_hash("laundry123")

# saat login
```

```
If check_password_hash(hashed, "laundry123"):

   print("password benar!")
```

4. Future-proof: hash yang dihasilkan include information tentang method dan parameter yang dipakai, jadi kalau nanti kita mau upgrade algoritma, system bisa detect dan handle mixed formats.

Format hash werkzeug: Pbkdf2:sha256:260000$salt$hash

Method yang dipakai

- 260000: jumlah iterations
- Salt: random salt
- Hash: hash actual

Dengan sistem ini, bahkan kalau attacker dapat hash dari database, mereka tidak bisa praktis crack password karena computational cost-nya terlalu tinggi.

2.7 otorisasi

Role-based access control (RBAC)

Kalau autentikasi adalah "siapa kamu?", maka otorisasi adalah "apa yang boleh kamu lakukan?". Rbac adalah pendekatan otorisasi di mana permission di-assign berdasarkan role, bukan individual user.

Konsep dasar rbac:

- User: individu yang pakai sistem
- Role: jabatan atau posisi yang punya set permission tertentu
- Permission: hak untuk melakukan action tertentu
- User assigned to role: user punya satu atau lebih role
- Role has permissions: role punya set permission

Flow-nya: user -> role -> permissions

Keuntungan rbac:

1. Scalability: kalau ada user baru dengan posisi yang sama, tinggal assign role yang sama. tidak perlu set permission satu-satu.

2. Maintainability: mau ubah permission untuk semua admin? Tinggal ubah role admin, semua user dengan role admin otomatis kebaca permission baru.

3. Clarity: lebih mudah dipahami dan di-audit. "user x adalah admin, jadi punya permission a, b, c"

4. Separation of duties: bisa enforce bahwa certain permissions tidak boleh dikombinasikan dalam satu role untuk security reasons.

Di sistem kita, implementasinya simple tapi effective: kita punya tabel roles yang menyimpan available roles, dan users punya foreign key ke role mereka.

Perbedaan roles karyawan dan costumer disistem laundry kita sebagai berikut:

Role: karyawan

- Tujuan: mengelola sistem dan bisnis laundry secara keseluruhan
- Permissions:
    - Akses penuh ke dashboard admin
    - Kelola layanan laundry (crud services): tambah layanan baru, edit harga, hapus layanan
    - Lihat semua pesanan dari semua customer
    - Kelola user
    - Akses laporan pendapatan
- Use case: pemilik laundry, manager, atau staff karyawan

Role: customer

- Tujuan: menggunakan sistem untuk mengelola pesanan mereka sendiri
- Permissions:
    - Akses dashboard user
    - Buat pesanan baru
    - Lihat pesanan mereka sendiri
    - Edit/cancel pesanan mereka sendiri
    - tidak bisa akses management layanan
    - tidak bisa lihat pesanan user lain
- Use case: customer laundry atau operator yang input pesanan

Perbedaan key:

- Karyawan/admin: system-wide access, fokus ke management
- costumer/user: personal access, fokus ke operational

Separation ini penting untuk:

- Security: user biasa tidak bisa ngubah-ngubah data penting
- Data privacy: user tidak bisa lihat data user lain
- User experience: interface yang ditampilkan sesuai dengan kebutuhan role, tidak overwhelming

Pembatasan akses halaman

Pembatasan akses adalah implementasi praktis dari otorisasi. Kita harus ensure bahwa user cuma bisa akses halaman dan fitur yang sesuai dengan role mereka.

Teknik pembatasan akses:

1. Decorator-based protection:

```
From functools import wraps

Def admin_required(f):
   @wraps(f)
   def decorated_function(args, kwargs):
      if not current_user.is_authenticated:
         return redirect('/login')
      if current_user.role.role_name != 'admin':
         flash('akses ditolak!')
         return redirect('/dashboard')
      return f(args, kwargs)
   return decorated_function

@app.route('/admin/services')
@admin_required
Def manage_services():
   # hanya admin yang bisa akses
```

2. Template-level protection

```
{% if current_user.role.role_name == 'admin' %}

   <a href="/admin/services">kelola layanan</a>

{% endif %}
```

3. Data-level protection:

```
# user cuma bisa lihat pesanan sendiri
If current_user.role.role_name == 'user':
   orders = order.query.filter_by(user_id=current_user.user_id).all()
Else:
   # admin bisa lihat semua
   orders = order.query.all(
```

Redirect otomatis: kalau user coba akses halaman yang tidak boleh, jangan tampilkan error 403 yang ugly. Better user experience adalah redirect mereka ke halaman yang appropriate dengan message yang informatif:

```
If not authorized:
   flash('anda tidak memiliki akses ke halaman tersebut', 'warning')
   return redirect(url_for('dashboard'))
```

2.8 responsive web design

Responsive web design adalah pendekatan design di mana website secara otomatis adjust layout, size, dan functionality-nya untuk provide optimal viewing experience across berbagai devices dan screen sizes.

Zaman sekarang, user akses web dari berbagai device: smartphone, tablet, laptop, desktop monitor gede, bahkan tv. Screen size-nya vary dari 4 inch sampai 50+ inch. Kalau website cuma designed untuk desktop, bakal kelihatan jelek atau bahkan unusable di mobile. Sebaliknya, kalau cuma designed untuk mobile, bakal kerasa cramped di desktop.

Prinsip-prinsip responsive design:

1. Fluid grids: layout menggunakan relative units (%, em, rem) instead of fixed units (px). Jadi element-element bisa grow or shrink sesuai container-nya.
2. Flexible images: images juga responsive. Width set ke max 100% dari container, jadi tidak overflow keluar container di screen kecil.
3. Media queries: css technique untuk apply different styles based on device characteristics (terutama screen width).

```
Css
/* mobile first */
.container { padding: 10px; }

/* tablet dan lebih gede */
@media (min-width: 768px) {
  .container { padding: 20px; }
}

/* desktop */
@media (min-width: 1024px) {
  .container { padding: 30px; }
}
```

4. Mobile-first approach: start design untuk mobile (smallest screen), lalu progressively enhance untuk screen yang lebih gede. Ini lebih baik daripada desktop-first karena lebih mudah add features untuk screen gede daripada strip away features untuk screen kecil.

Keuntungan responsive design:

- Better ux: user bisa access dan navigate website dengan nyaman dari device apapun
- Single codebase: tidak perlu maintain separate mobile site (m.example.com)

- Seo friendly: google prefer responsive design dan bahkan prioritize mobile-friendly sites di ranking
- Cost effective: daripada build mobile app + web app terpisah
- Future proof: adaptable untuk new devices dengan screen size baru

Bootstrap dan grid system

Bootstrap adalah css framework yang paling popular untuk building responsive websites. Bootstrap menyediakan pre-built components, utilities, dan grid system yang membuat development jadi jauh lebih cepat.

Keunggulan bootstrap:

1. Pre-styled components: button, form, navbar, card, modal, dll sudah styled dengan baik. Tinggal pakai dengan add classes.
2. Responsive grid system: ini adalah core dari bootstrap. Grid system membagi layout jadi 12 kolom. Element bisa occupy beberapa kolom tergantung screen size.
3. Utility classes: classes untuk spacing (margin, padding), typography, colors, display, flexbox, dll. Sangat convenient untuk quick styling.
4. Javascript components: carousel, dropdown, modal, tooltip, dll yang sudah include javascript functionality.
5. Customizable: bisa customize theme dengan sass variables atau bahkan build custom version.

Bootstrap grid system deep dive: grid system bootstrap based on flexbox dan punya 6 breakpoints:

- Xs: < 576px (extra small - phones)
- Sm: ≥ 576px (small - large phones, portrait tablets)
- Md: ≥ 768px (medium - tablets)
- Lg: ≥ 992px (large - desktops)
- Xl: ≥ 1200px (extra large - large desktops)
- Xxl: ≥ 1400px (extra extra large - larger desktops)

Structure grid:

```
Html
<div class="container"> <!-- container untuk center content -->
  <div class="row"> <!-- row untuk horizontal grouping -->
    <div class="col-md-6"> <!-- column -->
      content 1
    </div>
    <div class="col-md-6">
      content 2
    </div>
```

```
  </div>
</div>
```

Class naming: col-{breakpoint}-{columns}

- Col-12: full width di semua screen
- Col-md-6: half width di medium screen ke atas
- Col-lg-4: 1/3 width di large screen ke atas

Bisa combine multiple classes untuk different behavior di different screen sizes:

Html

```
<div class="col-12 col-md-6 col-lg-4">
  <!-- full width di mobile
     half width di tablet
     1/3 width di desktop -->
</div>
```

Praktik terbaik:

- gunakan `container` atau `container-fluid` sebagai wrapper
- columns harus always inside `row`
- content goes inside columns, not directly inside row
- total columns dalam row should add up to 12

Dengan bootstrap, kita bisa build responsive layout yang complex dengan markup yang relatively simple, tanpa perlu nulis banyak custom css. Ini significantly speed up development process dan ensure consistency across aplikasi.

BAB III

IMPLEMENTASI

Bab ini menjelaskan perancangan dan implementasi praktik sistem secara menyeluruh, meliputi alur kerja aplikasi, struktur database relasional, dan arsitektur folder aplikasi.

Alat dan bahan

a. Perangkat keras:

- processor: intel core i5 atau setara
- ram: minimal 4 gb
- storage: minimal 2 gb ruang kosong

b. Perangkat lunak:

- python 3.11.x
- visual studio code 1.85
- db browser for sqlite 3.12
- web browser (chrome/firefox)
- git 2.42

c. Library python (requirements.txt):

- flask==3.0.0
- flask-sqlalchemy==3.1.1
- flask-login==0.6.3
- pymysql==1.1.0
- werkzeug==3.0.1

3.1 gambaran sederhana miya laundry

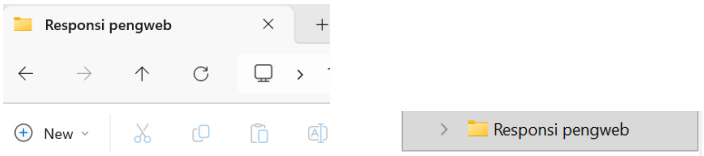
Sistem manajemen laundry professional ini dirancang untuk digitalisasi dan mengotomasi proses bisnis laundry, mulai dari pengelolaan layanan, pencatatan pesanan, sampai manajemen user dengan sistem keamanan yang proper.

3.2 pembuatan virtual environment

Virtual environment adalah langkah pertama dan paling penting sebelum mulai development. Ini memastikan dependency project kita terisolasi dan tidak bentrok dengan project lain atau package sistem.

Step 1: persiapan direktori project

buat folder untuk project kita. Buka terminal atau command prompt maupun file explore, lalu navigate ke lokasi di mana kamu mau simpan project:



direktori project

saat kita ada di dalam folder responsi pengweb yang masih kosong. Ini akan jadi root directory project kita.

Step 2: membuat virtual environment

Jalankan command berikut untuk membuat virtual environment dengan nama `venv`:

Bash di windows

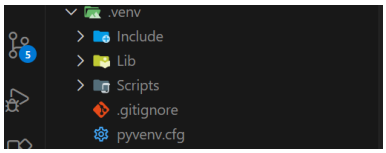
Python -m venv venv

syntax pembuatan venv

Command ini akan membuat folder `venv` di dalam project folder kita. Proses ini mungkin memakan waktu beberapa detik sampai satu menit tergantung speed komputer.

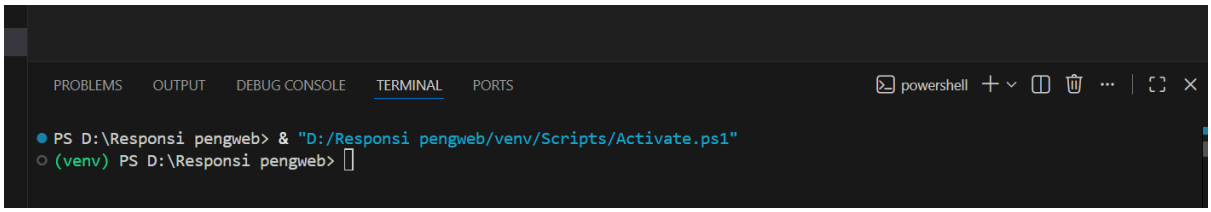
Setelah itu python membuat folder `venv` dengan struktur lengkap yang berisi

- Copy dari python interpreter
- Pip (package installer)
- Setuptools dan wheel (untuk instalasi package)
- Folder untuk libraries yang akan kita install



library.venv

Step 3: mengaktifkan venv



mengaktifkan venv

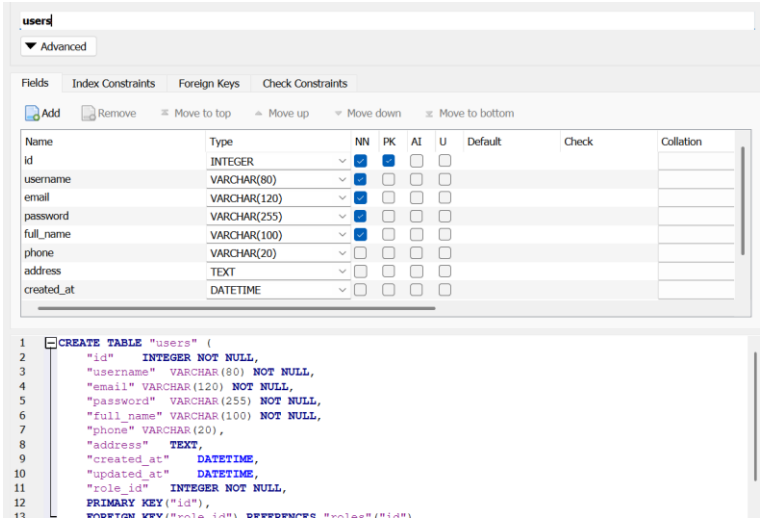
Setiap kita akan melakukan perubahan, penambahan program(develovment) dll kita mesti mengaktifkan venv, contoh jika kita ingin menginstal library flask maka kita perlu mengaktifkan venv kita

3.3 Pembuatan database

Perancangan dan pembuatan database langkah
1: membuat database sqlite

2. Buka db browser for sqlite
3. Klik menu file → new database
4. Simpan file
5. Pilih lokasi: `responsi_pengweb/instance/

1. Tabel user



Tabel saat user registrasi atau penambahan user

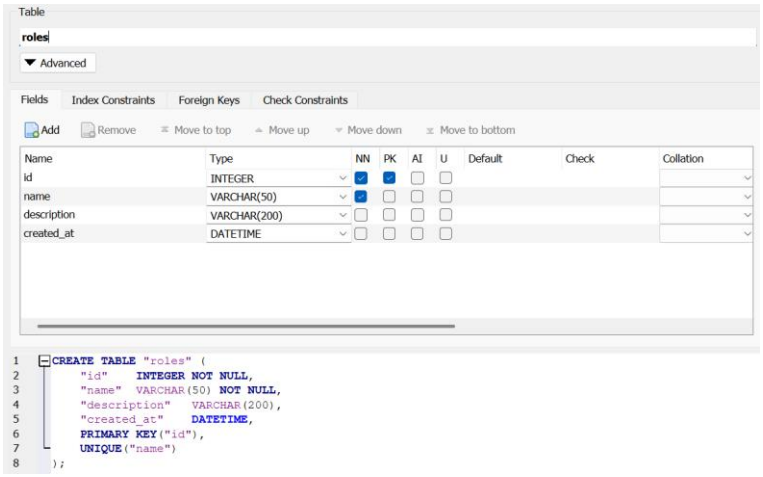
Deskripsi kolom:

- Id: integer, primary key, not null. Berfungsi sebagai identitas unik bagi setiap pengguna untuk memastikan tidak ada data akun yang tumpang tindih.
- Username: varchar(80), not null. Menyimpan nama unik yang digunakan pengguna untuk masuk (login) ke dalam sistem.
- Email: varchar(120), not null. Menyimpan alamat surat elektronik pengguna sebagai sarana komunikasi dan verifikasi akun.
- Password: varchar(255), not null. Menyimpan kata sandi dalam bentuk terenkripsi untuk mengamankan akses ke akun pengguna.
- Full_name: varchar(100), not null. Menyimpan nama lengkap pengguna sesuai identitas asli untuk keperluan administratif atau laporan.
- Phone: varchar(20). Menyimpan nomor telepon pengguna yang dapat dihubungi untuk keperluan konfirmasi atau layanan.
- Address: text. Tempat menyimpan alamat lengkap tempat tinggal pengguna untuk mempermudah proses penjemputan atau pengiriman.
- Created_at: datetime. Catatan waktu otomatis saat akun pengguna pertama kali didaftarkan ke dalam sistem.
- Updated_at: datetime. Catatan waktu otomatis saat terjadi perubahan data pada profil atau informasi akun pengguna.
- Role_id: integer, not null. Berfungsi sebagai foreign key yang menghubungkan pengguna dengan tabel peran (roles) untuk menentukan hak akses mereka (seperti admin, kurir, atau pelanggan).

Tabel ini berfungsi sebagai basis data utama untuk mengelola informasi pengguna dan keamanan sistem (otentikasi). Dengan adanya kolom role_id, tabel ini mendukung sistem

hak akses yang terorganisir, sementara kolom username dan password memastikan bahwa hanya pengguna sah yang dapat masuk ke aplikasi.

2. Tabel roles



Tabel .roles terdspst admin/karyawan dan user/costumer

a. Deskripsi kolom:

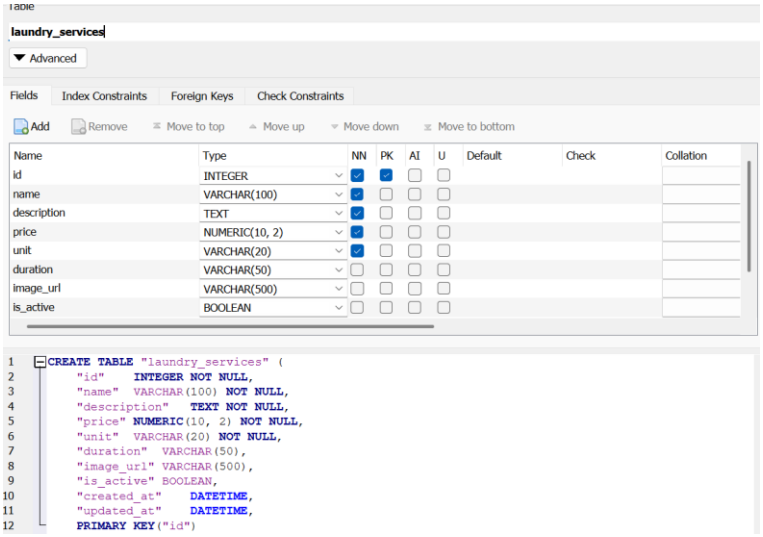
- Id (integer): kunci utama (primary key). Catatan: di aplikasi kamu, saat insert data baru, id harus dimasukkan secara manual (karena auto increment tidak aktif).
- Name (varchar 50): nama role. Bersifat unique (tidak boleh ada nama role yang sama) dan not null (wajib diisi).
- Description (varchar 200): penjelasan singkat mengenai fungsi role tersebut.
- Created_at (datetime): mencatat waktu pembuatan data.

b. Relasi

- Fk mapping: kolom role_id di tabel users menyambung ke kolom id di tabel roles.
- Constraint: jika kamu menyetel on delete restrict, maka kamu tidak akan bisa menghapus data di tabel roles selama masih ada user yang memakainya. Ini sudah benar untuk menjaga keamanan data.

Tabel ini berfungsi sebagai master table untuk manajemen hak akses berbasis peran (role-based access control atau rbac). Semua jenis tingkatan pengguna yang tersedia di dalam sistem didefinisikan secara terpusat di sini.

3. Tabel laundry_services



tabel. Laundry service

deskripsi kolom:

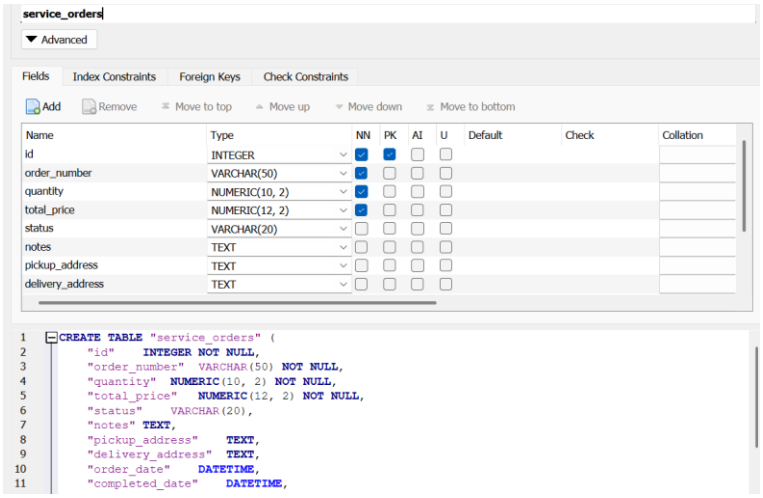
- Id: integer, primary key, not null. Berfungsi sebagai identitas unik untuk setiap jenis layanan laundry.
- Name: varchar(100), not null. Menyimpan nama layanan laundry (contoh: "cuci komplit").
- Description: text, not null. Penjelasan detail mengenai apa saja yang termasuk dalam layanan tersebut.
- Price: numeric(10, 2), not null. Harga layanan dengan presisi dua angka di belakang koma untuk akurasi nilai mata uang.
- Unit: varchar(20), not null. Satuan hitung layanan, seperti "kg", "pcs", atau "meter".
- Duration: varchar(50). Estimasi waktu pengerjaan (contoh: "2 hari" atau "6 jam").
- Image_url: varchar(500). Link atau path file gambar untuk ikon atau foto layanan.
- Is_active: boolean. Status apakah layanan ini sedang aktif dan tersedia untuk dipesan atau tidak.
- Created_at: datetime. Catatan waktu otomatis saat data layanan pertama kali dibuat.
- Updated_at: datetime. Catatan waktu otomatis saat data layanan terakhir kali diubah.

Tabel ini berfungsi menjadi acuan (reference) bagi tabel transaksi atau pesanan. Dengan adanya kolom is_active, kamu bisa menonaktifkan layanan tertentu tanpa harus menghapus datanya, sehingga riwayat transaksi lama tetap aman.

Contoh data

- service_id=1, service_name="cuci kering", price_per_unit=15000.00, unit="kg"
- service_id=2, service_name="cuci setrika", price_per_unit=18000.00, unit="kg"
- dan seterusnya sampai minimal 20+ layanan untuk memenuhi requirement

4. Tabel service_orders



Tabel service order

Deskripsi kolom:

- Id: integer, primary key, not null. Berfungsi sebagai identitas unik untuk setiap transaksi pesanan agar data tidak tertukar.
- Order_number: varchar(50), not null. Menyimpan kode unik atau nomor invoice pesanan sebagai referensi pelacakan bagi pelanggan dan admin.
- Quantity: numeric(10, 2), not null. Menyimpan jumlah pesanan (bisa berupa angka desimal) untuk mengakomodasi satuan seperti berat (kg) atau luas.
- Total_price: numeric(12, 2), not null. Menyimpan total nilai transaksi yang harus dibayarkan untuk pesanan tersebut.
- Status: varchar(20). Menyimpan status proses pesanan (contoh: "pending", "proses", atau "selesai") untuk memantau progres layanan.
- Notes: text. Tempat menyimpan catatan tambahan atau instruksi khusus dari pelanggan terkait pesanan mereka.
- Pickup_address: text. Menyimpan detail lokasi atau alamat lengkap tempat pengambilan barang milik pelanggan.
- Delivery_address: text. Menyimpan detail lokasi atau alamat tujuan pengiriman barang setelah layanan selesai dikerjakan.
- Order_date: datetime. Catatan tanggal dan waktu saat pesanan pertama kali dibuat oleh pelanggan.
- Completed_date: datetime. Catatan tanggal dan waktu saat pesanan dinyatakan selesai atau telah sukses dikirim ke pelanggan.

Tabel ini berfungsi sebagai pusat penyimpanan data transaksi pesanan. Dengan adanya kolom pickup_address dan delivery_address, tabel ini mampu menangani alur kerja logistik, sementara kolom status dan catatan waktu (date) memungkinkan pihak penyedia jasa untuk memantau kecepatan layanan serta riwayat transaksi secara akurat.

a. Foreign keys:

1. Role_id references roles.id

- On delete restrict: tidak bisa menghapus role (peran) jika masih ada pengguna yang terdaftar dengan peran tersebut.
- On update cascade: jika id pada tabel roles berubah, maka role_id pada tabel users akan ikut berubah secara otomatis.
- Fungsi: memastikan setiap pengguna memiliki tingkat akses yang valid. Fk ini menjaga agar tidak ada pengguna yang memiliki peran "fiktif".

2. User_id references users.id (pada tabel service_orders)
   - On delete cascade: jika sebuah akun pengguna dihapus, maka seluruh riwayat pesanan milik pengguna tersebut akan ikut terhapus otomatis.
   - On update cascade: jika id pengguna berubah, otomatis data user_id pada tabel pesanan akan diperbarui.
   - Fungsi: mengaitkan pesanan dengan pemiliknya. Fk ini memastikan setiap transaksi layanan selalu terikat pada profil pelanggan yang terdaftar.
3. Service_id references laundry_services.id (pada tabel service_orders)
   - On delete restrict: layanan laundry tidak bisa dihapus jika masih ada data pesanan aktif yang merujuk pada layanan tersebut.
   - On update cascade: jika id layanan di tabel master berubah, otomatis data pada tabel pesanan akan menyesuaikan.
   - Fungsi: menjamin bahwa setiap pesanan merujuk pada katalog layanan yang tersedia dan valid.

b. Relasi antar tabel:

1. Relasi: roles → users (one-to-many)

- Penjelasan: setiap satu peran (role) dalam sistem dapat dimiliki oleh banyak pengguna (users), namun setiap pengguna hanya bisa terikat pada satu peran tertentu saja.
- Implementasi: kolom role_id pada tabel users bertindak sebagai foreign key yang merujuk pada id di tabel roles.
- Fungsi: memisahkan data profil pengguna dengan hak aksesnya, sehingga pengaturan izin akses (seperti admin atau pelanggan) menjadi lebih terpusat dan rapi.

2. Relasi: users → service_orders (one-to-many)

- Penjelasan: satu pengguna (user) dapat melakukan banyak pesanan layanan (service_orders) seiring berjalannya waktu, tetapi satu pesanan hanya boleh dimiliki oleh satu pengguna.
- Implementasi: kolom user_id pada tabel service_orders merujuk pada id di tabel users.
- Fungsi: mengaitkan setiap transaksi pesanan dengan pelanggan yang memesan, sehingga histori transaksi setiap pengguna dapat dilacak dengan akurat.

3. Relasi: laundry_services → service_orders (one-to-many)

- Penjelasan: satu jenis layanan laundry dapat dipilih atau digunakan dalam banyak pesanan yang berbeda, namun satu detail pesanan pada baris tersebut merujuk pada satu jenis layanan induk.

- Implementasi: kolom service_id pada tabel service_orders merujuk pada id di tabel laundry_services.
- Fungsi: memastikan setiap pesanan memiliki referensi harga, satuan, dan nama layanan yang valid dari katalog layanan laundry yang tersedia.

c. Integritas referensial & aturan (constraints):

Database ini menggunakan aturan khusus untuk menjaga konsistensi data di seluruh tabel:

- Pencegahan data yatim (orphan data): dengan adanya foreign key, sistem tidak akan mengizinkan pembuatan data di tabel service_orders jika user_id atau service_id yang dimasukkan tidak ada di tabel master.
- Mekanisme cascade:
  - Jika data pengguna dihapus dari tabel users, maka secara otomatis seluruh pesanan yang terkait dengan pengguna tersebut di tabel service_orders akan ikut terhapus untuk menghindari sampah data.
  - Jika terdapat perubahan id pada tabel master (roles, users, atau laundry_services), perubahan tersebut akan otomatis diperbarui ke tabel-tabel yang merujuk kepadanya agar relasi tidak terputus.
- Mekanisme restrict:
  - Layanan di tabel laundry_services tidak dapat dihapus jika layanan tersebut masih tercatat dalam histori pesanan aktif di tabel service_orders, guna menjaga integritas laporan keuangan.

Contoh query relasional

```
Sql
-- get semua pesanan user tertentu dengan detail layanan
Select
  o.order_id,
  o.order_date,
  o.quantity,
  o.total_price,
  o.status,
  s.service_name,
  s.price_per_unit,
  s.unit
From service_orders o
Join laundry_services s on o.service_id = s.service_id
Where o.user_id = 5;


-- get semua pesanan dengan info user dan layanan
```

```
Select
   o.order_id,
   u.username,
   u.full_name,
   s.service_name,
   o.quantity,
   o.total_price,
   o.status
From service_orders o
Join users u on o.user_id = u.user_id
Join laundry_services s on o.service_id = s.service_id
Order by o.order_date desc;

-- get total revenue per layanan
Select
   s.service_name,
   count(o.order_id) as total_orders,
   sum(o.total_price) as total_revenue
From laundry_services s
Left join service_orders o on s.service_id = o.service_id
Group by s.service_id, s.service_name;
```

3.3 struktur folder aplikasi

a. Struktur proyek aplikasi miya laundry

Berdasarkan hasil pengembangan yang telah dilakukan, aplikasi "miya laundry" dibangun menggunakan arsitektur yang terorganisir untuk memisahkan antara logika bisnis, tampilan, dan data. Berikut adalah rincian strukturnya:

- Penyimpanan data (database): aplikasi ini menggunakan database sqlite dengan file bernama miya_laundry_database.db yang tersimpan di dalam folder instance. Folder ini berfungsi sebagai tempat penyimpanan data lokal yang bersifat privat bagi aplikasi. Di dalamnya terdapat tabel-tabel utama seperti laundry_services, roles, service_orders, dan users.
- Struktur model (models): di dalam folder app/models/, terdapat file-file python seperti user.py, service.py, order.py, dan role.py. File-file ini berfungsi untuk mendefinisikan skema tabel database (menggunakan sqlalchemy). Jadi, setiap data yang terlihat di tampilan sqlite tadi diatur aturannya di sini.

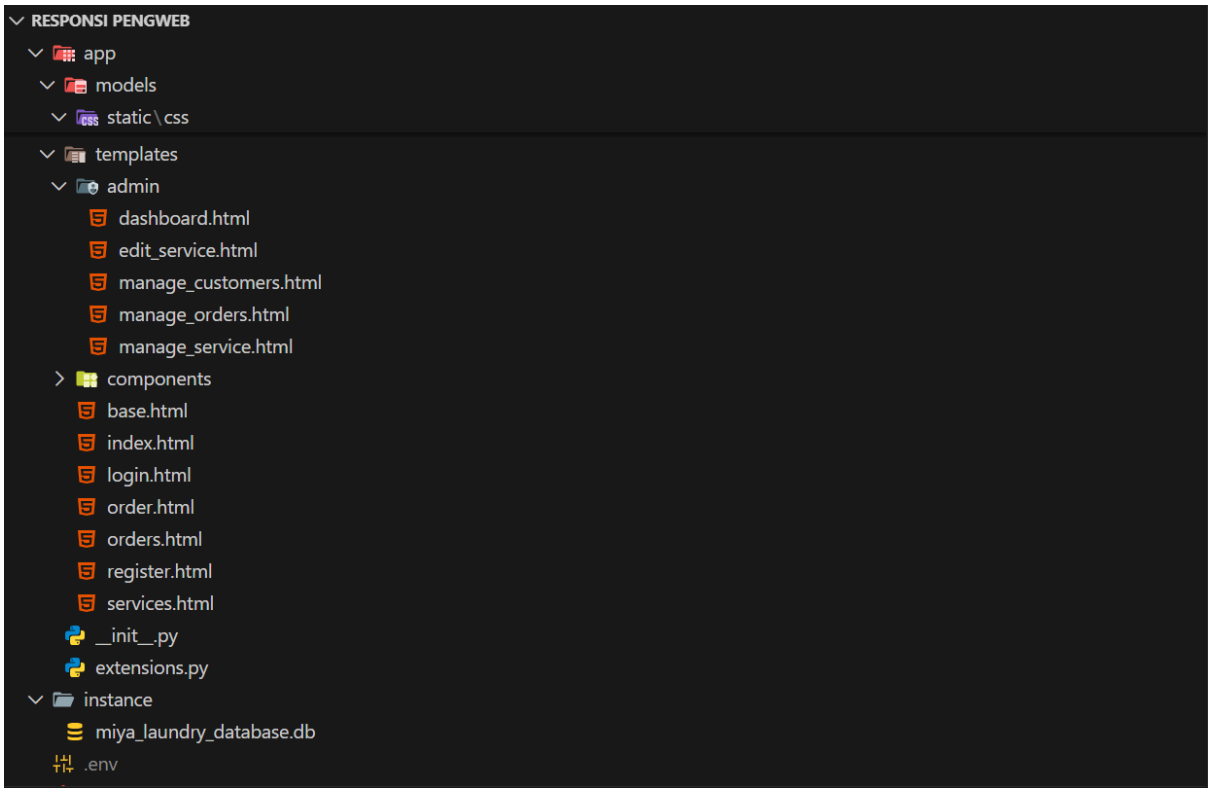
- Logika navigasi (routes): folder app/routes/ berisi alur navigasi aplikasi, yang dipisahkan menjadi beberapa bagian:
    - Auth.py: mengatur fitur login dan daftar akun.
    - Admin.py: khusus untuk mengatur halaman manajemen laundry (tambah/edit layanan).
    - Main.py: mengatur halaman utama yang bisa diakses oleh pelanggan.
- Tampilan (templates): semua file html disimpan di folder app/templates/. Struktur ini dibagi lagi agar lebih rapi:
    - Folder admin/: berisi halaman khusus pengelola, seperti manage_service.html dan dashboard.html.
    - Folder components/: berisi potongan kecil tampilan yang dipakai berulang kali, seperti navbar.html dan footer.html.
    - File base.html: berfungsi sebagai kerangka utama (template induk) agar tampilan web seragam di semua halaman.
- Konfigurasi dan eksekusi:
    - Config.py: tempat menyimpan pengaturan rahasia dan koneksi database.
    - Run.py: file utama yang dijalankan untuk menyalakan server aplikasi.
    - Requirements.txt: daftar semua library (bahan baku) yang dibutuhkan agar aplikasi bisa berjalan di komputer

```
Responsi_pengweb/
│
├── .venv/                  # virtual environment python
│                       # menyimpan library flask, sqlalchemy, dll
│
├── app/                    # package utama aplikasi flask
│   │
│   ├── __init__.py             # inisialisasi flask app
│   │                       # registrasi database, login manager, dan blueprint
│   │
│   ├── models/                 # folder model database (orm)
│   │   ├── __init__.py         # inisialisasi models
│   │   ├── user.py             # model user (akun pengguna)
│   │   ├── role.py             # model role (admin / user)
│   │   ├── service.py          # model service (layanan laundry)
│   │   └── order.py            # model order (pesanan laundry)
│   │
│   ├── routes/                 # folder routing (blueprint flask)
│   │   ├── __init__.py         # inisialisasi blueprint
│   │   ├── main.py             # route halaman utama (home, index)
│   │   ├── auth.py             # route login, register, logout
```

```
│   │   └── admin.py            # route khusus admin (dashboard & manajemen)
│   │
│   ├── templates/              # folder template html (jinja2)
│   │   │
│   │   ├── components/         # komponen template yang dipakai ulang
│   │   │   ├── navbar.html     # navbar (menu navigasi)
│   │   │   └── footer.html     # footer website
│   │   │
│   │   ├── admin/              # template khusus admin
│   │   │   ├── dashboard.html     # dashboard admin
│   │   │   ├── manage_customers.html # kelola data pelanggan
│   │   │   ├── manage_orders.html    # kelola pesanan laundry
│   │   │   ├── manage_service.html   # kelola layanan laundry
│   │   │   └── edit_service.html     # edit data layanan
│   │   │
│   │   ├── base.html           # template dasar (layout utama)
│   │   │                       # berisi navbar, footer, dan block content
│   │   ├── index.html          # halaman utama website
│   │   ├── login.html          # halaman login user
│   │   ├── register.html       # halaman registrasi user
│   │   ├── services.html       # daftar layanan laundry
│   │   ├── orders.html         # daftar pesanan user
│   │   └── order.html          # form pemesanan laundry
│   │
│   ├── static/                 # file statis
│   │   └── css/
│   │       └── custom.css      # css tambahan untuk tampilan website
│
├── instance/                   # folder data runtime
│   └── miya_laundry_database.db   # database sqlite
│
├── .env                        # environment variable (secret key, config db)
├── .gitignore                  # file/folder yang diabaikan git
├── config.py                   # konfigurasi aplikasi flask
├── migrate_db.py               # script migrasi / inisialisasi database
```

```
├── prepare_deploy.py          # script persiapan deployment
├── procfile                   # konfigurasi deployment (railway)
├── railway.json               # konfigurasi deploy railway
├── requirements.txt           # daftar library python yang dibutuhkan
├── readme.md                  # dokumentasi project
├── run.py                     # file utama untuk menjalankan aplikasi
└── test_db.py                 # file testing koneksi & struktur database
```

 # implementasi di vscode



stryktur project

b. Penjelasan detail setiap file utama:

1. folder model, fungsi dari setiap file model adalah sebagai berikut:

1. __init__.py
   File ini berfungsi sebagai pusat pengelolaan impor seluruh model yang ada dalam folder models. Dengan mendefinisikan __all__, file ini memudahkan pemanggilan model dari bagian lain aplikasi serta menjaga struktur kode tetap rapi dan terorganisir.
2. user.py
   File ini mendefinisikan model User yang digunakan untuk menyimpan data pengguna sistem, seperti username, email, password, dan informasi profil lainnya. Model ini juga terintegrasi dengan Flask-Login melalui UserMixin untuk mendukung manajemen sesi dan autentikasi pengguna. Selain itu, model User memiliki relasi dengan tabel Role dan ServiceOrder.
3. role.py
   File ini berisi model Role yang digunakan untuk mengatur peran pengguna dalam sistem, seperti Admin, Karyawan, dan Customer. Model ini mendukung penerapan Role-Based Access Control (RBAC), di mana satu role dapat dimiliki oleh banyak user, sehingga memudahkan pengaturan hak akses dalam aplikasi.
4. service.py
   File ini mendefinisikan model LaundryService yang menyimpan data layanan laundry yang ditawarkan, seperti nama layanan, deskripsi, harga, satuan layanan, dan estimasi waktu pengerjaan. Model ini memungkinkan sistem untuk mengelola daftar layanan secara dinamis serta memiliki relasi langsung dengan pesanan laundry.
5. order.py
   File ini berisi model ServiceOrder yang berfungsi mencatat transaksi atau pesanan laundry. Model ini menyimpan informasi penting seperti nomor pesanan, jumlah layanan, total harga, status pesanan, alamat penjemputan dan pengantaran, serta waktu pemesanan dan penyelesaian. Model ini menjadi penghubung utama antara user dan layanan laundry.

Dengan pemisahan model ke dalam file terpisah, aplikasi menjadi lebih modular, mudah dipelihara, dan mudah dikembangkan. Struktur ini juga mendukung praktik pengembangan aplikasi web yang baik, khususnya dalam membangun sistem berbasis database dengan Flask dan SQLAlchemy.

2. Folder route, fungsi dari setiap file route sebagai berikut

Folder routes pada aplikasi Sistem Manajemen Laundry berbasis Flask berfungsi sebagai pengatur alur navigasi dan logika aplikasi yang menghubungkan interaksi pengguna dengan sistem dan database. Folder ini memanfaatkan konsep Blueprint Flask untuk memisahkan routing berdasarkan fungsi utama aplikasi, sehingga struktur kode menjadi lebih modular, terorganisir, dan mudah dikembangkan.

Secara rinci, fungsi dari setiap file dalam folder routes adalah sebagai berikut:

1. __init__.py
   File ini berfungsi sebagai pusat pengelolaan seluruh blueprint yang digunakan dalam aplikasi, yaitu auth_blueprint, main_blueprint, dan admin_blueprint. Dengan adanya file ini, proses registrasi routing pada aplikasi utama menjadi lebih sederhana serta menjaga struktur routing tetap rapi dan terpusat.
2. auth.py
   File ini menangani seluruh proses autentikasi pengguna, meliputi login, registrasi, dan logout. Pada file ini diterapkan validasi kredensial, pengamanan password menggunakan hashing, serta pengelolaan sesi pengguna dengan Flask-Login. Selain itu, file ini juga mengatur pengalihan halaman berdasarkan role pengguna setelah berhasil login.
3. admin.py
   File ini berisi routing khusus untuk karyawan atau admin yang bertugas mengelola sistem. Di dalamnya terdapat fitur dashboard, manajemen layanan laundry, manajemen pelanggan, serta pengelolaan pesanan. File ini juga menerapkan pembatasan akses berbasis role melalui middleware karyawan_required untuk memastikan hanya pengguna dengan hak akses tertentu yang dapat mengakses halaman administrasi.
4. main.py
   File ini mengatur routing utama yang dapat diakses oleh pengguna umum dan customer, seperti halaman beranda, daftar layanan, pemesanan laundry, daftar pesanan, pembatalan pesanan, serta halaman informasi aplikasi. File ini menjadi penghubung utama antara customer dan sistem dalam melakukan transaksi layanan laundry.

Dengan pemisahan routing ke dalam file yang berbeda sesuai dengan fungsinya, folder routes mendukung prinsip separation of concerns, meningkatkan keamanan melalui kontrol akses berbasis role, serta memudahkan pengelolaan dan pengembangan aplikasi di masa mendatang.

3. folder costum , fungsi costum sebagai berikut

Folder custom pada aplikasi Sistem Manajemen Laundry berbasis Flask berfungsi sebagai pengatur tampilan antarmuka (user interface) dan pengalaman pengguna (user experience) melalui penggunaan Custom CSS. File CSS dalam folder ini dirancang untuk

menghasilkan tampilan aplikasi yang modern, responsif, konsisten, dan mudah digunakan pada berbagai ukuran perangkat.

Secara rinci fungsi folder costum sebagai berikut:

1. Pengaturan Variabel Global (:root)
   Bagian ini mendefinisikan variabel CSS global seperti warna utama, warna sekunder, tipografi, jarak (spacing), radius sudut, bayangan (shadow), dan transisi animasi. Penggunaan variabel ini bertujuan untuk menjaga konsistensi desain dan mempermudah pemeliharaan tampilan aplikasi.
2. Global Styles
   Bagian ini mengatur gaya dasar elemen HTML seperti body, heading, paragraf, dan link. Pengaturan ini memastikan tampilan teks yang rapi, nyaman dibaca, serta konsisten di seluruh halaman aplikasi.
3. Komponen Navigasi (Navbar)
   CSS pada bagian ini mengatur tampilan navbar agar terlihat modern dengan gradasi warna, efek hover, serta posisi sticky. Navbar dirancang agar tetap responsif dan mudah digunakan baik pada desktop maupun perangkat mobile.
4. Hero Section dan Button Styles
   Bagian ini berfungsi untuk mempercantik halaman utama dengan hero section yang menarik, dilengkapi dengan tombol interaktif yang memiliki efek animasi dan transisi, sehingga meningkatkan daya tarik visual aplikasi.
5. Card, Table, dan Form Styles
   CSS ini digunakan untuk menata tampilan kartu layanan, tabel data, serta form input. Desain dibuat dengan sudut membulat, bayangan halus, dan efek hover untuk meningkatkan kenyamanan dan kejelasan informasi bagi pengguna.
6. Alert, Badge, dan Dashboard Styles
   Bagian ini mengatur tampilan notifikasi, label status, serta kartu statistik pada dashboard admin. Warna dan ikon digunakan untuk membedakan status seperti sukses, peringatan, dan kesalahan secara visual.
7. Footer dan Komponen Pendukung
   CSS pada bagian footer dan elemen tambahan seperti pagination, modal, breadcrumb, dan loading spinner berfungsi untuk melengkapi tampilan aplikasi agar terlihat profesional dan konsisten di seluruh halaman.
8. Responsif, Animasi, dan Aksesibilitas
   File CSS ini juga mengimplementasikan desain responsif menggunakan media query, animasi sederhana untuk transisi elemen, serta dukungan aksesibilitas seperti focus style dan high contrast mode agar aplikasi dapat digunakan oleh berbagai jenis pengguna.

Dengan adanya folder custom yang berisi CSS terstruktur dan terorganisir, aplikasi tidak hanya berfungsi dengan baik secara teknis, tetapi juga memiliki tampilan yang menarik, ramah pengguna, serta siap digunakan pada berbagai perangkat. Hal ini mendukung kualitas aplikasi secara keseluruhan baik dari sisi fungsional maupun visual.

4. folder template

Folder template pada aplikasi Sistem Manajemen Laundry berbasis Flask berfungsi sebagai fondasi utama aplikasi, tempat seluruh komponen inti sistem diinisialisasi dan dikonfigurasikan sebelum aplikasi dijalankan. Folder ini memastikan aplikasi dapat berjalan dengan baik melalui pengaturan konfigurasi, koneksi database, manajemen autentikasi, serta penyediaan data awal sistem.

Secara rinci, fungsi dari setiap file dalam folder template adalah sebagai berikut:

1. __init__.py
   File ini berfungsi sebagai pusat pembuatan aplikasi Flask menggunakan Application Factory Pattern. Di dalam file ini dilakukan konfigurasi aplikasi, inisialisasi database dan sistem login, pendaftaran seluruh blueprint, serta pembuatan tabel database. Selain itu, file ini juga bertugas menyediakan data awal seperti role pengguna, akun default, dan data layanan laundry, sehingga aplikasi siap digunakan sejak pertama kali dijalankan.
2. extensions.py
   File ini berfungsi sebagai tempat inisialisasi seluruh extension yang digunakan oleh aplikasi, seperti SQLAlchemy untuk pengelolaan database dan LoginManager untuk manajemen autentikasi pengguna. Dengan memisahkan extension ke dalam file khusus, struktur kode menjadi lebih rapi, modular, dan terhindar dari masalah ketergantungan antar modul.

Dengan adanya folder template, aplikasi memiliki struktur awal yang kuat dan terorganisir, mendukung pengembangan berkelanjutan, serta memastikan seluruh komponen sistem terintegrasi dengan baik sejak proses inisialisasi hingga aplikasi siap digunakan oleh pengguna.

5. Folder instance, fungsinya sebagai berikut :

Folder instance pada aplikasi Sistem Manajemen Laundry berbasis Flask berfungsi sebagai penyimpanan data runtime aplikasi, khususnya data yang bersifat dinamis dan dihasilkan saat aplikasi berjalan. Folder ini umumnya tidak berisi kode program, melainkan menyimpan file penting yang dibutuhkan oleh aplikasi selama proses operasional.

Secara rinci, fungsi dari file yang terdapat dalam folder instance adalah sebagai berikut:

1. miya_laundry_database.db
   File ini merupakan database SQLite yang digunakan oleh aplikasi untuk menyimpan seluruh data sistem. Di dalam file ini tersimpan data pengguna, role, layanan laundry, serta data pesanan layanan. Database ini menjadi pusat penyimpanan informasi sehingga seluruh proses transaksi, autentikasi, dan pengelolaan layanan dapat berjalan secara terintegrasi dan konsisten.

Dengan adanya folder instance, aplikasi dapat memisahkan kode program dengan data aplikasi, sehingga meningkatkan keamanan, kerapihan struktur proyek, serta memudahkan proses pengelolaan dan pemeliharaan sistem. Folder ini mendukung praktik pengembangan aplikasi Flask yang baik dengan menjaga data penting tetap terisolasi dari logika program utama.

6. Folder env

    1. penggunaan environment variables pada aplikasi Sistem Manajemen Laundry berbasis Flask memiliki peranan yang sangat penting dalam mengatur konfigurasi aplikasi secara terpusat dan aman. File konfigurasi ini digunakan untuk menyimpan pengaturan utama yang mendukung jalannya aplikasi tanpa harus menuliskannya langsung di dalam kode program.
    2. Pada baris SECRET_KEY, saya menetapkan kunci rahasia yang digunakan oleh Flask untuk keperluan keamanan, seperti pengelolaan sesi (session management), perlindungan data pengguna, serta mekanisme keamanan lainnya. Dengan adanya secret key ini, aplikasi menjadi lebih aman dari risiko manipulasi data sesi.
    3. Selanjutnya, pada bagian DATABASE_URL, saya mendefinisikan alamat koneksi database yang digunakan oleh aplikasi. Dalam praktikum ini, database yang digunakan adalah SQLite dengan nama miya_laundry_database.db. Penggunaan SQLite dipilih karena bersifat ringan, mudah dikonfigurasi, dan sangat cocok untuk tahap pengembangan serta pengujian aplikasi tanpa memerlukan instalasi database server tambahan.
    4. Variabel FLASK_ENV diset ke nilai development, yang menandakan bahwa aplikasi sedang dijalankan dalam mode pengembangan. Mode ini memungkinkan saya sebagai pengembang untuk melakukan perubahan kode secara fleksibel tanpa harus khawatir terhadap pembatasan yang biasanya diterapkan pada lingkungan produksi.
    5. Kemudian, pada variabel FLASK_DEBUG yang bernilai True, saya mengaktifkan fitur debug pada Flask. Dengan fitur ini, ketika terjadi kesalahan pada aplikasi, sistem akan menampilkan pesan error secara detail di browser. Hal ini sangat membantu saya dalam proses identifikasi kesalahan dan perbaikan bug selama pengembangan aplikasi berlangsung.

    Secara keseluruhan, penerapan environment variables ini membuat konfigurasi aplikasi menjadi lebih rapi, aman, dan mudah dikelola. Dengan memisahkan pengaturan lingkungan dari kode program utama, aplikasi Sistem Manajemen Laundry menjadi lebih terstruktur, mudah dikembangkan, serta siap untuk diimplementasikan pada lingkungan yang berbeda di masa mendatang.

7. Folder admin dalam folder template memiliki fungsi sebagai berikut

    Berdasarkan hasil perancangan dan implementasi antarmuka administrator pada Sistem Manajemen Laundry Miya Laundry, folder admin di dalam direktori templates berfungsi sebagai kumpulan halaman khusus yang digunakan oleh admin untuk mengelola seluruh aktivitas sistem. Seluruh file di dalam folder ini saling terintegrasi

dan menggunakan template inheritance dari base.html, sehingga tampilan aplikasi menjadi konsisten, rapi, dan mudah dipahami.

Secara lebih rinci, fungsi dari setiap file dalam folder admin dapat dijelaskan sebagai berikut.

1. File Dashboard Admin berfungsi sebagai halaman utama yang pertama kali diakses oleh administrator setelah login. Pada halaman ini saya menampilkan ringkasan kondisi sistem secara keseluruhan, seperti jumlah layanan, jumlah pengguna, total pesanan, serta total pendapatan. Selain itu, dashboard juga menyediakan menu aksi cepat dan tabel layanan terbaru, sehingga admin dapat langsung mengetahui kondisi sistem dan mengambil tindakan tanpa harus berpindah halaman terlalu banyak.
2. File Tambah/Edit Layanan digunakan untuk mengelola data layanan laundry. Halaman ini bersifat dinamis karena dapat digunakan baik untuk menambahkan layanan baru maupun mengedit layanan yang sudah ada. Saya memanfaatkan kondisi pada template untuk membedakan kedua fungsi tersebut. Form yang disediakan mencakup nama layanan, deskripsi, harga, satuan, estimasi waktu, hingga URL gambar, sehingga data layanan dapat dikelola secara lengkap dan terstruktur.
3. File Kelola Pelanggan berfungsi untuk menampilkan data seluruh pelanggan yang terdaftar dalam sistem. Pada halaman ini, admin dapat melihat informasi penting seperti username, nama lengkap, email, nomor telepon, jumlah pesanan, serta tanggal bergabung. Halaman ini dirancang sebagai halaman monitoring, sehingga tidak menyediakan fitur pengeditan langsung demi menjaga keamanan dan integritas data pelanggan.
4. File Manajemen Pesanan digunakan oleh admin untuk mengelola seluruh pesanan laundry yang masuk. Pada halaman ini, saya menampilkan daftar pesanan lengkap beserta statusnya, mulai dari menunggu, diproses, siap diambil, hingga dikirim. Admin juga dapat memperbarui status pesanan secara langsung melalui dropdown, sehingga alur kerja laundry dapat dikelola dengan lebih efisien dan terkontrol.
5. File Kelola Layanan berfungsi sebagai halaman utama untuk manajemen CRUD layanan laundry. Pada halaman ini, admin dapat melihat seluruh layanan yang tersedia, lengkap dengan gambar, deskripsi singkat, harga, dan status layanan. Tersedia pula fitur edit dan hapus layanan dengan konfirmasi, sehingga meminimalkan kesalahan dalam pengelolaan data.

Secara keseluruhan, folder admin pada templates dirancang untuk mendukung kebutuhan administrator dalam mengelola sistem laundry secara menyeluruh. Setiap file memiliki peran yang jelas dan saling melengkapi, mulai dari monitoring sistem, pengelolaan layanan, pelanggan, hingga pesanan. Dengan struktur ini, sistem menjadi lebih mudah digunakan, terorganisir, dan mencerminkan penerapan konsep pengembangan aplikasi web yang baik.

8. folder component didalam folder template dengan fungsinya sebagai berikut

Berdasarkan implementasi antarmuka pada Sistem Manajemen Laundry Miya Laundry, folder components di dalam direktori templates berfungsi sebagai kumpulan halaman inti dan komponen utama tampilan aplikasi. Folder ini menjadi fondasi antarmuka pengguna karena hampir seluruh halaman dalam sistem bergantung pada struktur dan komponen yang disediakan di dalamnya.

Seluruh file di folder ini dirancang dengan pendekatan template inheritance menggunakan base.html, sehingga tampilan aplikasi menjadi konsisten, mudah dikembangkan, dan efisien dalam pemeliharaan kode. Berikut adalah penjelasan fungsi masing-masing file berdasarkan implementasinya.

1. File base.html merupakan template utama atau kerangka dasar dari seluruh halaman web dalam aplikasi. Pada file ini saya mendefinisikan struktur HTML secara lengkap mulai dari DOCTYPE, <head>, hingga <body>.

   Di dalam bagian <head>, saya mengatur metadata seperti karakter encoding, viewport, dan deskripsi website. Saya juga mengintegrasikan Bootstrap 5, Bootstrap Icons, serta Google Fonts (Poppins) agar tampilan website modern, responsif, dan nyaman dilihat. Selain itu, saya mendefinisikan CSS global menggunakan variabel warna agar konsistensi desain tetap terjaga di seluruh halaman.

   Pada bagian body, base.html memuat navbar, flash message, konten utama, dan footer dengan menggunakan {% include %} dan {% block %}. Dengan cara ini, halaman lain cukup mewarisi base.html tanpa harus menulis ulang struktur dasar. File ini berfungsi sebagai tulang punggung tampilan aplikasi.

2. File index.html berfungsi sebagai halaman beranda (homepage) aplikasi Miya Laundry. Halaman ini dirancang sebagai halaman pertama yang dilihat pengguna.

   Di dalamnya terdapat hero section yang menampilkan deskripsi singkat layanan, tombol navigasi utama, serta gambar ilustrasi. Selain itu, halaman ini menampilkan fitur unggulan, layanan unggulan, dan call to action (CTA) yang menyesuaikan kondisi pengguna apakah sudah login atau belum.

   Fungsi utama file ini adalah sebagai media promosi layanan sekaligus pengarah pengguna untuk mendaftar, melihat layanan, atau langsung memesan laundry.

3. File login.html digunakan sebagai halaman autentikasi pengguna untuk masuk ke dalam sistem. Halaman ini menyediakan form input username dan password yang terhubung langsung ke route auth.login.

   Pada file ini juga saya menambahkan fitur toggle password menggunakan JavaScript sederhana agar pengguna dapat melihat atau menyembunyikan password yang dimasukkan. Halaman ini dibuat dengan desain yang bersih dan fokus pada kenyamanan pengguna.

Fungsi utama login.html adalah mengamankan akses sistem dan memastikan hanya pengguna terdaftar yang dapat menggunakan fitur lanjutan seperti pemesanan dan manajemen pesanan.

4. File register.html berfungsi sebagai halaman pendaftaran akun baru. Pada halaman ini, pengguna dapat mengisi data seperti username, email, nama lengkap, nomor telepon, serta password.

   Saya menambahkan validasi dasar seperti panjang password dan konfirmasi password, serta fitur toggle visibility password agar pengguna lebih mudah memastikan input yang dimasukkan. Halaman ini berperan penting dalam proses onboarding pengguna baru ke dalam sistem.

5. File service.html digunakan untuk menampilkan seluruh layanan laundry yang tersedia. Data layanan ditampilkan dalam bentuk grid kartu yang berisi gambar, nama layanan, deskripsi, harga, satuan, durasi, dan status ketersediaan.

   Halaman ini bersifat dinamis karena tombol aksi akan berubah tergantung kondisi pengguna, apakah sudah login, belum login, atau merupakan karyawan. Fungsi utama file ini adalah sebagai etalase layanan laundry yang memudahkan pengguna memilih layanan yang sesuai.

6. File order.html berfungsi sebagai halaman pemesanan layanan. Pada halaman ini, pengguna dapat melihat detail layanan yang dipilih, mengisi jumlah, alamat penjemputan, alamat pengiriman, serta catatan tambahan.

   Saya juga menambahkan perhitungan total harga secara dinamis menggunakan JavaScript agar pengguna langsung mengetahui estimasi biaya. Halaman ini menjadi inti dari proses transaksi karena menghubungkan layanan dengan data pesanan pengguna.

7. File orders.html digunakan sebagai halaman riwayat dan manajemen pesanan. Halaman ini bersifat fleksibel karena tampilannya menyesuaikan peran pengguna.

   Jika pengguna adalah pelanggan, halaman ini menampilkan riwayat pesanan beserta statusnya. Jika pengguna adalah karyawan/admin, halaman ini berfungsi sebagai dashboard pengelolaan pesanan, termasuk pembaruan status pesanan secara langsung.

   Fungsi utama file ini adalah untuk monitoring dan kontrol proses laundry dari awal hingga selesai.

Secara keseluruhan, folder components pada templates berperan sebagai pusat antarmuka utama Sistem Manajemen Laundry Miya Laundry. Setiap file memiliki fungsi yang jelas dan saling terintegrasi, mulai dari tampilan dasar, autentikasi, informasi layanan, hingga proses pemesanan dan pengelolaan pesanan.

Dengan penggunaan base.html sebagai template utama, sistem menjadi lebih terstruktur, konsisten, dan mudah dikembangkan. Implementasi ini menunjukkan

penerapan konsep template inheritance, user experience, dan modularisasi tampilan yang baik dalam pengembangan aplikasi web berbasis Flask.

9. Gitignore, fungsinya sebagai berikut

Berdasarkan implementasinya, file .gitignore pada proyek Sistem Manajemen Laundry Miya Laundry berfungsi sebagai mekanisme penyaring yang memastikan hanya file penting dan relevan saja yang masuk ke dalam repository Git.

Dengan mengabaikan file hasil kompilasi, cache, database lokal, virtual environment, konfigurasi editor, serta file environment, pengelolaan proyek menjadi, lebih aman rapi, dan profesional. Selain itu, penggunaan .gitignore ini menunjukkan bahwa proyek telah dikembangkan dengan memperhatikan best practice pengembangan aplikasi Flask, khususnya dalam hal keamanan data, efisiensi kolaborasi, dan kemudahan maintenance. File ini membantu mencegah kebocoran data sensitif serta menghindari konflik file yang tidak diperlukan antar developer. Secara keseluruhan, .gitignore berperan penting dalam menjaga kualitas dan kerapian struktur proyek Miya Laundry, sehingga aplikasi dapat dikembangkan dan dikelola dengan lebih baik.

10. config, fungsinya sebagai berikut

Berdasarkan konfigurasi yang telah saya terapkan pada file Config, dapat disimpulkan bahwa pengaturan aplikasi Sistem Manajemen Laundry Miya Laundry telah dirancang dengan memperhatikan aspek keamanan, fleksibilitas, dan kesiapan deployment.

Class Config berfungsi sebagai pusat pengaturan aplikasi yang mengelola berbagai kebutuhan penting, mulai dari keamanan aplikasi melalui SECRET_KEY, pengelolaan database menggunakan SQLAlchemy, hingga pengaturan sesi login pengguna. Dengan memanfaatkan environment variable, saya memastikan bahwa data sensitif seperti kunci rahasia dan koneksi database tidak ditulis secara langsung di dalam kode, sehingga lebih aman dan mudah disesuaikan pada berbagai lingkungan pengembangan maupun produksi.

Penggunaan SQLite pada tahap development mempermudah proses pengujian dan pengembangan aplikasi, sementara dukungan untuk PostgreSQL dan MySQL menunjukkan bahwa aplikasi telah disiapkan untuk kebutuhan skala produksi. Selain itu, pengaturan sesi seperti batas waktu login, cookie secure, HTTPOnly, dan SameSite menunjukkan bahwa aplikasi ini telah menerapkan praktik keamanan web yang baik.

Secara keseluruhan, konfigurasi ini membuat aplikasi Miya Laundry menjadi lebih terstruktur, aman, dan profesional. Dengan pengaturan yang terpusat dan jelas, aplikasi tidak hanya mudah dikembangkan, tetapi juga siap untuk digunakan secara nyata dalam lingkungan produksi.

11. Migrasi database

Berdasarkan implementasi script migrasi database yang telah saya buat, dapat disimpulkan bahwa proses perpindahan data dari SQLite ke PostgreSQL pada aplikasi Sistem Manajemen Laundry Miya Laundry telah dirancang secara sistematis, aman, dan siap digunakan untuk kebutuhan deployment.

Script ini berfungsi sebagai jembatan antara database lokal yang digunakan pada tahap pengembangan dengan database PostgreSQL yang lebih andal untuk lingkungan produksi. Pada bagian awal, saya memanfaatkan environment variable untuk menyimpan informasi koneksi database PostgreSQL, sehingga konfigurasi tetap aman dan fleksibel tanpa perlu mengubah kode program.

Penggunaan Flask application context memungkinkan proses migrasi tetap terintegrasi dengan struktur aplikasi utama, terutama dalam pembuatan tabel menggunakan SQLAlchemy. Dengan cara ini, struktur tabel yang dibuat di PostgreSQL sepenuhnya konsisten dengan model yang digunakan pada aplikasi. Setiap tabel utama dalam sistem, mulai dari tabel roles, users, laundry_services, hingga service_orders, dimigrasikan secara bertahap dan terkontrol. Proses insert data dilakukan menggunakan metode batch agar lebih efisien, serta dilengkapi dengan mekanisme penanganan konflik untuk mencegah duplikasi data. Selain itu, script ini juga dilengkapi dengan validasi konfigurasi, logging proses migrasi, serta penanganan error yang membantu dalam proses debugging. Dengan adanya script migrasi ini, aplikasi Miya Laundry menjadi lebih siap untuk digunakan pada lingkungan produksi karena data dapat dipindahkan dengan aman tanpa kehilangan struktur maupun isi database.

Secara keseluruhan, implementasi migrasi database ini menunjukkan bahwa sistem telah dirancang dengan mempertimbangkan aspek skalabilitas, keamanan, dan kesiapan deployment, sehingga mendukung aplikasi untuk berkembang ke tahap penggunaan nyata.

12. Prepare devloyment

Berdasarkan pembuatan dan pengujian script persiapan deployment ini, dapat disimpulkan bahwa aplikasi Sistem Manajemen Laundry Miya Laundry telah dilengkapi dengan mekanisme otomatis untuk memastikan kesiapan sistem sebelum dipublikasikan ke platform Railway.

Script ini saya rancang untuk membantu proses deployment agar lebih terstruktur dan meminimalkan kesalahan konfigurasi. Pada tahap awal, script memastikan keberadaan file .env yang berfungsi sebagai penyimpan konfigurasi sensitif, sehingga keamanan aplikasi tetap terjaga. Selanjutnya, pengecekan environment variable DATABASE_URL dilakukan untuk memastikan aplikasi telah terhubung dengan database postgresql yang digunakan pada lingkungan produksi. Proses instalasi dependency dilakukan secara otomatis menggunakan file requirements.txt, sehingga seluruh library yang dibutuhkan aplikasi dapat terpasang dengan benar tanpa harus dilakukan secara manual. Setelah itu, script menjalankan pengujian koneksi database untuk memastikan bahwa aplikasi benar-benar siap digunakan dan tidak mengalami kegagalan saat dijalankan di server Railway. Selain itu, script ini juga mendukung proses migrasi database secara otomatis apabila aplikasi menggunakan PostgreSQL.

Hal ini memungkinkan data dan struktur database tetap konsisten antara lingkungan pengembangan dan produksi. Di bagian akhir, script memberikan panduan langkah-langkah lanjutan yang harus dilakukan, seperti menghubungkan repository GitHub ke Railway dan mengatur environment variable yang diperlukan.

Dengan adanya script persiapan deployment ini, proses deploy aplikasi menjadi lebih aman, efisien, dan terkontrol. Script ini menunjukkan bahwa aplikasi Miya Laundry tidak hanya berfungsi dengan baik secara lokal, tetapi juga telah dipersiapkan secara matang untuk digunakan pada lingkungan produksi berbasis cloud.

Berdasarkan hasil perancangan dan implementasi Sistem Manajemen Laundry Professional berbasis Flask, dapat disimpulkan bahwa aplikasi ini telah dibangun secara terstruktur, modular, dan siap untuk digunakan baik pada tahap pengembangan maupun deployment ke lingkungan production. Setiap file dan konfigurasi dalam project memiliki peran yang jelas dan saling terintegrasi untuk mendukung fungsionalitas aplikasi secara keseluruhan.

13. Procfile

File Procfile berfungsi sebagai instruksi kepada platform deployment (seperti Railway atau Heroku) untuk menjalankan aplikasi menggunakan web server production. Baris web: gunicorn run:app menunjukkan bahwa aplikasi dijalankan menggunakan **Gunicorn**, yang lebih stabil dan efisien dibandingkan server bawaan Flask, sehingga cocok untuk lingkungan production.

14. railway.json

File railway.json digunakan untuk mengatur proses build dan deployment aplikasi di Railway. Konfigurasi ini menentukan:

- Metode build menggunakan Nixpacks
- Perintah menjalankan aplikasi
- Health check untuk memastikan aplikasi berjalan normal
- Kebijakan restart jika aplikasi mengalami kegagalan
  Dengan adanya file ini, proses deployment menjadi lebih terkontrol dan otomatis.

15. README.md / README.txt

File README berfungsi sebagai dokumentasi utama proyek. Di dalamnya berisi:

- Deskripsi aplikasi
- Fitur utama
- Teknologi yang digunakan
- Struktur folder
- Panduan instalasi dan deployment
- Diagram relasi database
  Dokumentasi ini memudahkan pengguna atau pengembang lain untuk memahami cara kerja dan penggunaan aplikasi.

16. requirements.txt

File requirements.txt berisi daftar seluruh dependency Python yang dibutuhkan oleh aplikasi. Library seperti Flask, Flask-SQLAlchemy, Flask-Login, serta database driver (PyMySQL dan psycopg2) disertakan agar aplikasi:

- Konsisten dijalankan di berbagai environment
- Mudah dipindahkan dari SQLite ke MySQL atau PostgreSQL
  Keberadaan MySQL di file ini bukan kesalahan, melainkan sebagai persiapan deployment ke database production.

17. run.py

File run.py berfungsi sebagai **entry point** aplikasi. File ini bertugas:

- Membuat instance aplikasi Flask melalui create_app()
- Menjalankan aplikasi pada host dan port yang sesuai
- Mengambil port dari environment variable saat deployment
  File ini menjadi penghubung antara konfigurasi aplikasi dan proses eksekusi.

18. config.py

File config.py berisi konfigurasi utama aplikasi, seperti:

- Secret key untuk keamanan session
- Konfigurasi database berbasis environment variable
- Pengaturan keamanan cookie
- Durasi session login dengan pemisahan konfigurasi ini, aplikasi menjadi lebih aman, fleksibel, dan mudah disesuaikan antara development dan production.

19. File .env (Environment Variables)

File .env digunakan untuk menyimpan konfigurasi sensitif seperti SECRET_KEY, DATABASE_URL, dan mode aplikasi. File ini tidak disertakan ke repository publik demi keamanan, namun sangat penting dalam proses deployment.

20. migrate_db.py

File ini digunakan untuk melakukan migrasi data dari database SQLite ke PostgreSQL. Script ini:

- Membaca data dari SQLite
- Membuat struktur tabel di PostgreSQL
- Memindahkan data secara otomatis
  File ini sangat membantu saat aplikasi berpindah dari tahap development ke production.

21. prepare_deployment.py

File ini berfungsi untuk mempersiapkan aplikasi sebelum deployment, meliputi:

- Pengecekan file .env
- Instalasi dependency
- Pengujian koneksi database
- Menjalankan migrasi database jika diperlukan
  Dengan adanya script ini, risiko error saat deployment dapat diminimalkan.

22. test_db.py

File test_db.py digunakan untuk memastikan koneksi database berjalan dengan baik sebelum aplikasi di-deploy. Script ini melakukan:

- Tes koneksi database
- Perhitungan jumlah data pada setiap tabel
- Pengujian relasi antar tabel
  File ini berperan sebagai alat validasi kesiapan database.

3.4 Penjelasan alur sistem laundry

A. Set up user/ costumer

1. Registrasi user/costumer

Customer baru yang mau pakai sistem harus register terlebih dahulu. Mereka mengisi form registrasi dengan data:

- username (unique identifier)
- nama lengkap
- email
- nomor telepon
- password (akan di-hash oleh sistem) dan konfirmasi password

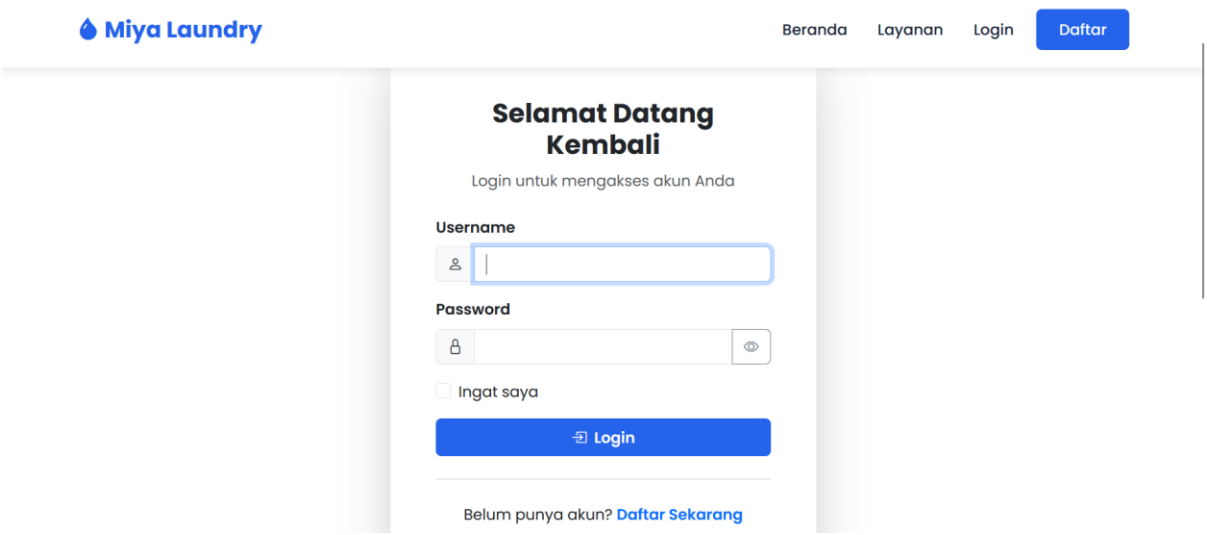
Jika sudah memiliki akun maka langsung log in

2. Login

User yang sudah register bisa login dengan username dan password. Sistem akan:

- verify apakah username exist di database
- compare hash password yang diinput dengan hash yang tersimpan
- kalau match: create session dan redirect ke dashboard sesuai role
- kalau tidak match: tampilkan error message "username atau password salah"

Setelah login, user punya session yang bertahan sampai mereka logout atau session expire.



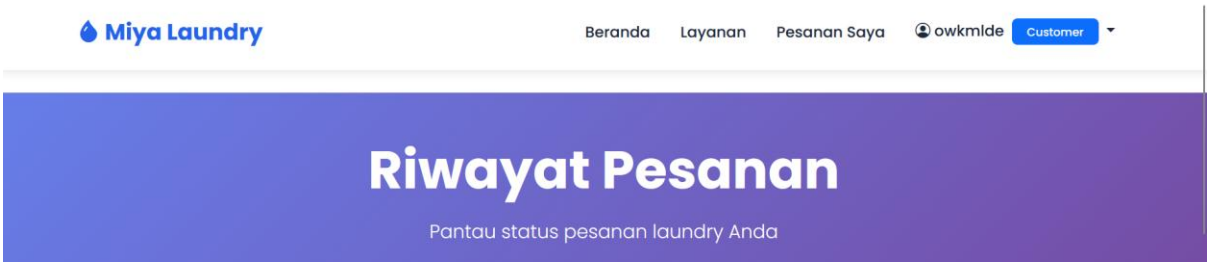
Gambar.fitur untuk user/costumer login

Jika costumer belum daftar atau salah memasukan autentikasi

3. Dashboard user (customer/operator)

Setelah login sebagai costumer, mereka akan melihat dashboard sederhana yang ditampilkan yaitu

- welcome message dengan nama mereka
- jumlah total pesanan/transaksi yang mereka punya
- menu untuk membuat pesanan baru
- list pesanan mereka dengan status (pending, diproses, selesai)



Dashboard/akses yang dimiliki costumer

4. Membuat pesanan baru

a. clik klik "buat pesanan baru" yang ada di menu layanan dan akan melihat form:

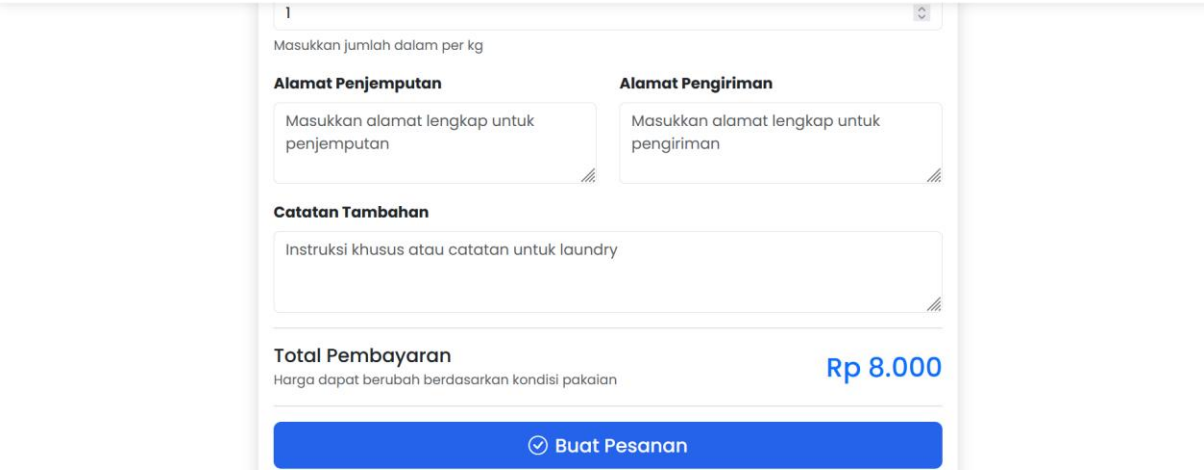
- input quantity (berapa kg atau berapa item)
- input tanggal pesanan
- alamat penjemputan dan pengiriman
- textarea untuk catatan tambahan (optional)

b. sistem akan otomatis calculate total harga berdasarkan layanan yang dipilih dan quantity. Misalnya pilih "cuci setrika (3kg)" quantity 2, total harga = rp 18.000 x 2 = rp 36.000.

c. setelah user submit, pesanan tersimpan ke database dengan:

- user_id: dari user yang login
- service_id: dari layanan yang dipilih
- quantity, total_price, alamat penjemputan, alamat pengiriman, catatan tambahan
- status: default "pending"

user langsung melihat notifikasi "pesanan berhasil dibuat!" dan diredirect ke list pesanan mereka.



Fitur untuk membuat pesanan

5. Mengelola pesanan costumer

- view detail: klik pesanan untuk lihat full detail
- edit pesanan: kalau status masih "pending", bisa edit quantity atau ganti layanan. Sistem recalculate total harga.
- cancel/delete pesanan: hapus pesanan yang tidak jadi.

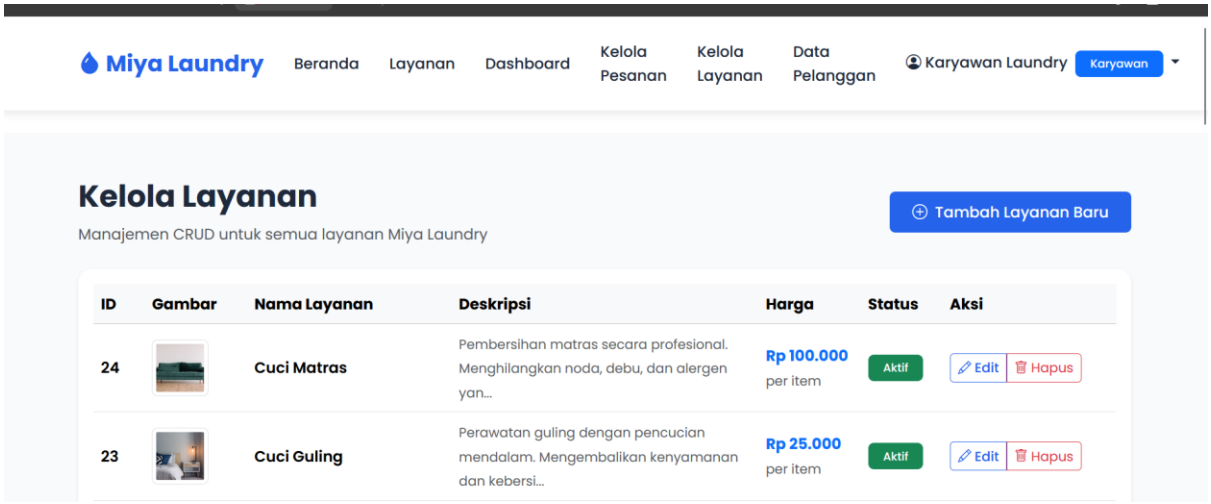
User customer tidak bisa melihat atau edit pesanan user lain. Sistem enforce ini di level database query dan authorization check.

B. set up karyawan/admin

1. Setup awal (admin/karyawan)

Ketika sistem pertama kali dijalankan, admin perlu setup data master terlebih dahulu. Admin login ke sistem dan mengakses menu management layanan. Di sini admin bisa menambahkan berbagai jenis layanan laundry yang tersedia, misalnya:

- cuci kering (3kg): rp 15.000
- cuci setrika (3kg): rp 18.000
- dry cleaning (per item): rp 25.000
- express service (tambahan): rp 10.000



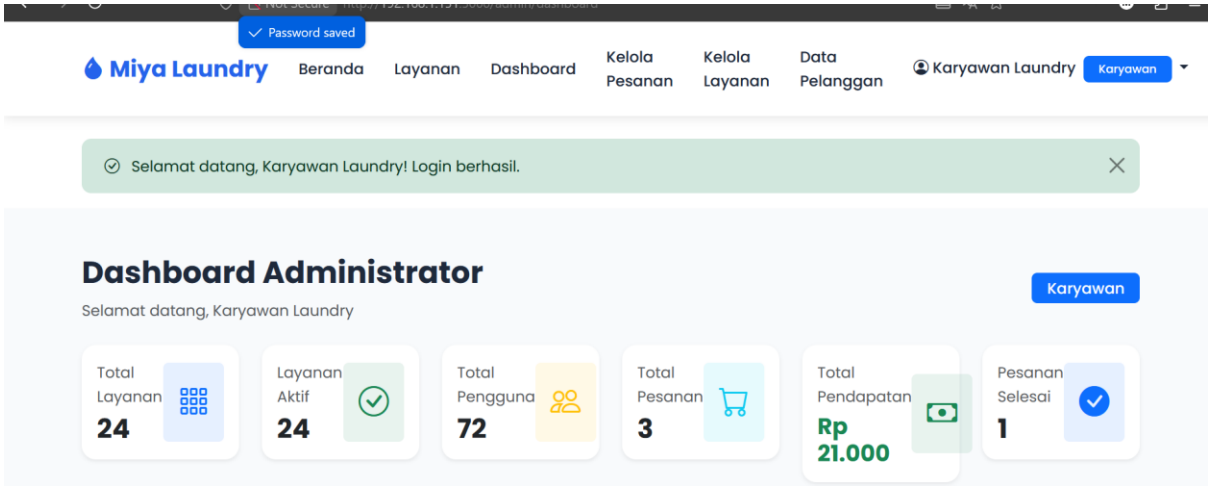
Fitur kelola layanan

Setiap layanan punya detail seperti nama layanan, deskripsi, harga per unit, satuan (kg atau item) dan estimasi waktu serta url gambar. Admin bisa sewaktu-waktu edit harga atau deskripsi kalau ada perubahan, atau menghapus layanan yang sudah tidak ditawarkan lagi.

2. Dashboard admin/karyawan

Kalau yang login adalah admin, mereka melihat dashboard yang lebih lengkap:

- total semua pesanan di sistem
- total layanan yang tersedia
- total registered users
- recent orders dari semua customer
- menu management untuk layanan dan pesanan



Fitur dashboard admin/karyawan

3. Management layanan (admin only)

Admin bisa full crud layanan:

- create: tambah layanan baru dengan detail lengkap
- read: view list semua layanan
- update: edit harga, deskripsi, satuan layanan existing
- delete: hapus layanan yang tidak ditawarkan lagi (dengan confirm)

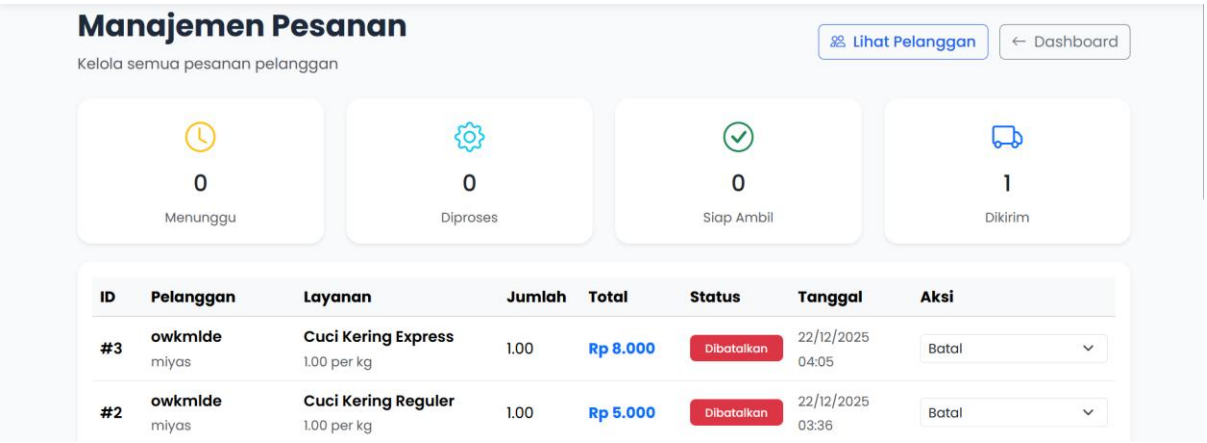
Setiap operasi ada feedback notification ke admin.

9. View semua pesanan (admin/karyawan)

Admin punya view ke semua pesanan dari semua customer. Di sini admin bisa:

- lihat detail setiap pesanan
- filter pesanan by status atau customer
- update status pesanan (misal dari "pending" ke "diproses", dari "diproses" ke "selesai")
- delete pesanan jika diperlukan

Ini membantu admin untuk track dan manage workflow laundry.

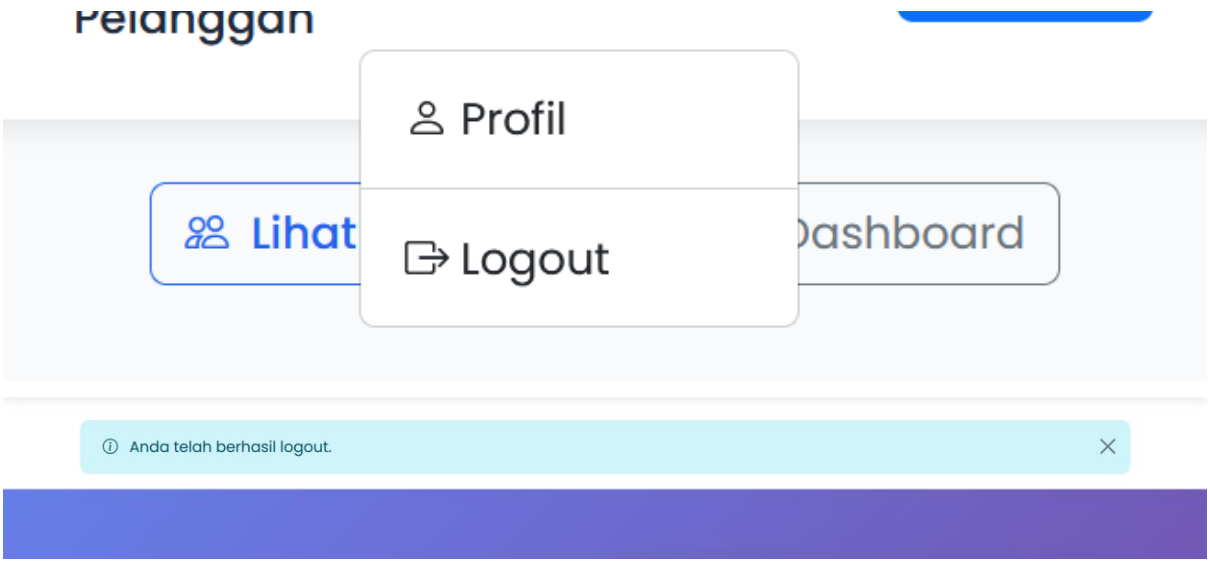


Fitur manajemen pesanan

10. Logout

User atau admin bisa logout kapan saja. Sistem akan:

- destroy session mereka
- redirect ke halaman login
- tampilkan message "anda telah logout"



Fitur logout

Setelah logout, mereka tidak bisa akses halaman protected lagi sampai login ulang.

Security features yang berjalan di background:

- setiap request ke halaman protected, sistem check apakah user sudah login (`@login_required`)
- setiap request ke halaman admin, sistem check apakah user punya role admin (`@admin_required`)
- kalau unauthorized, automatic redirect ke halaman appropriate dengan message

- sql injection prevention via parameterized queries
- xss prevention via jinja2 auto-escaping
- password selalu di-hash, tidak pernah plain text

Sistem ini end-to-end cover user journey dari register sampai complete transaction, dengan clear separation antara apa yang bisa dilakukan costumer vs karyawan.

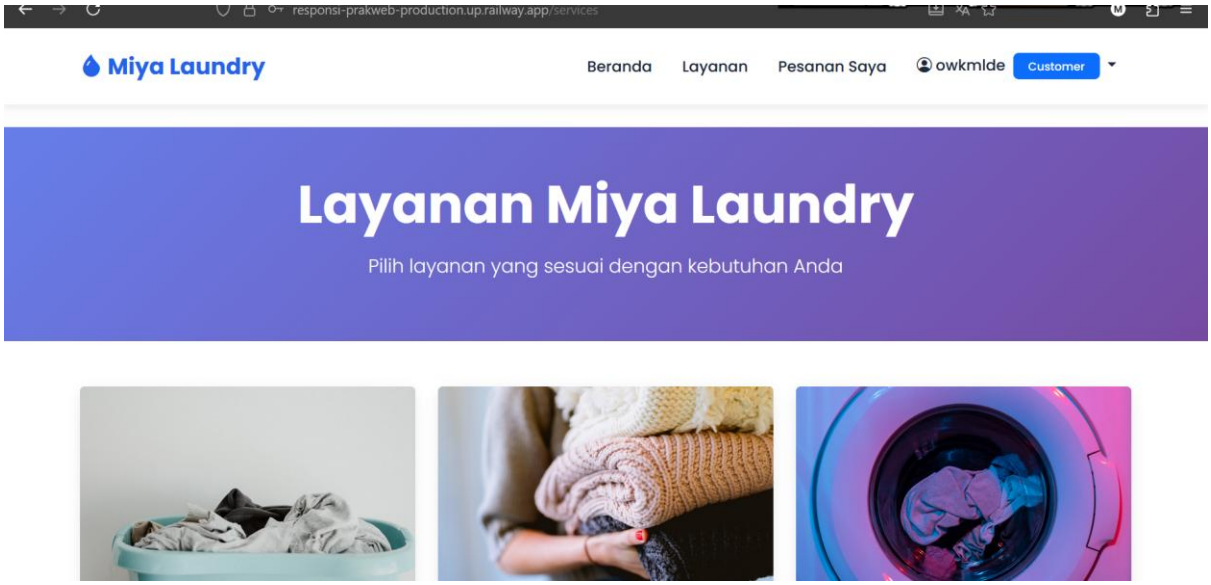
BAB IV

HASIL DAN OUTPUT YANG DIHARAPKAN

1. Tampilan view

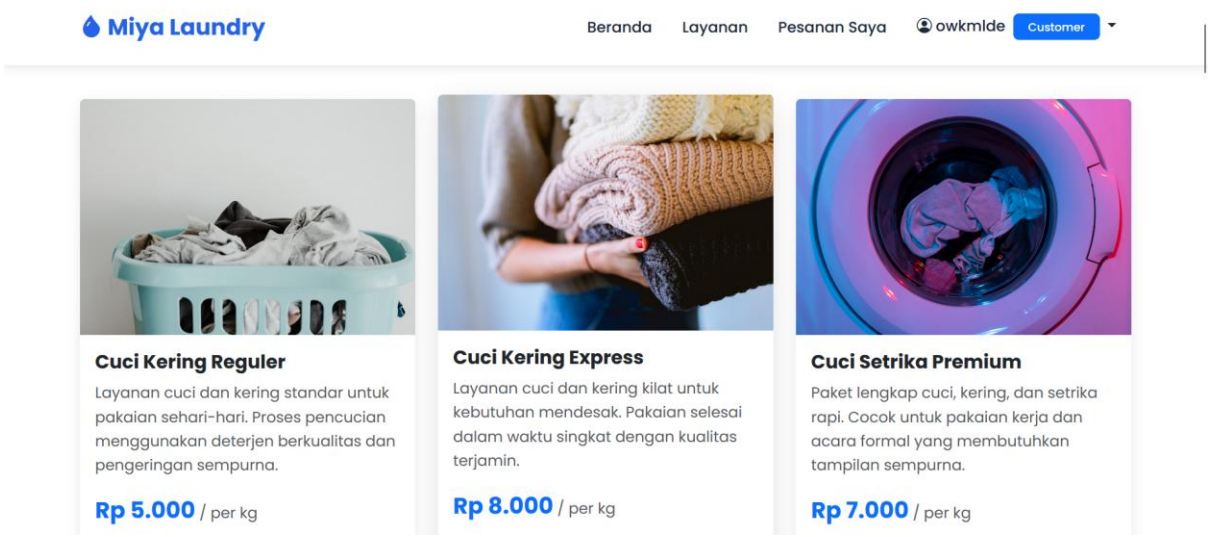
A fitur - fitur user

Dashboard



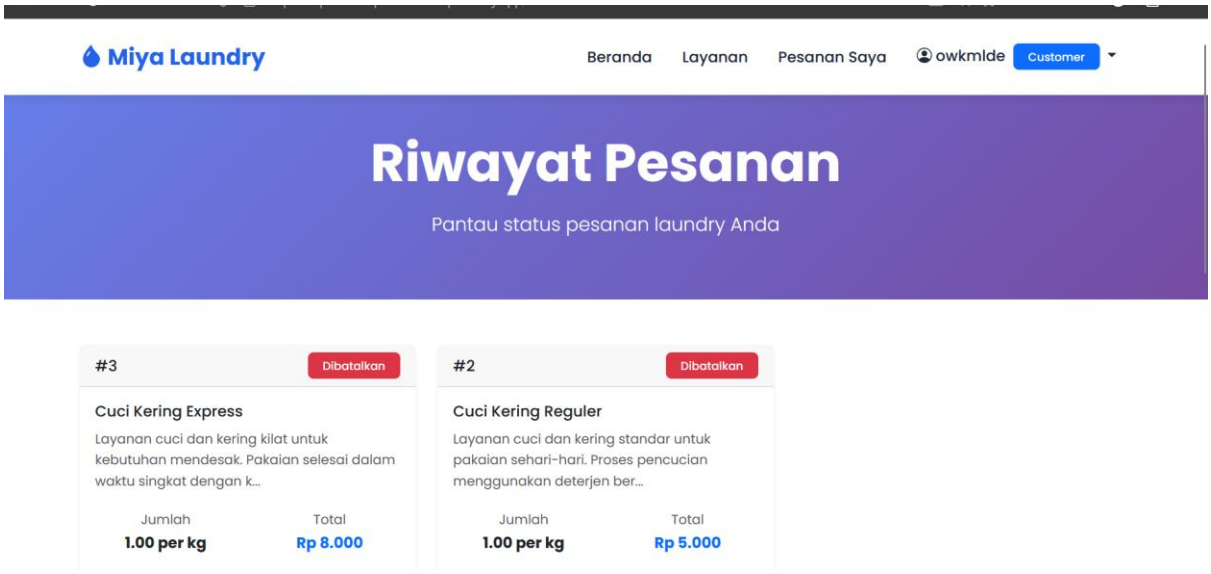
Halaman dashboard customer

Layanan



Halaman layanan

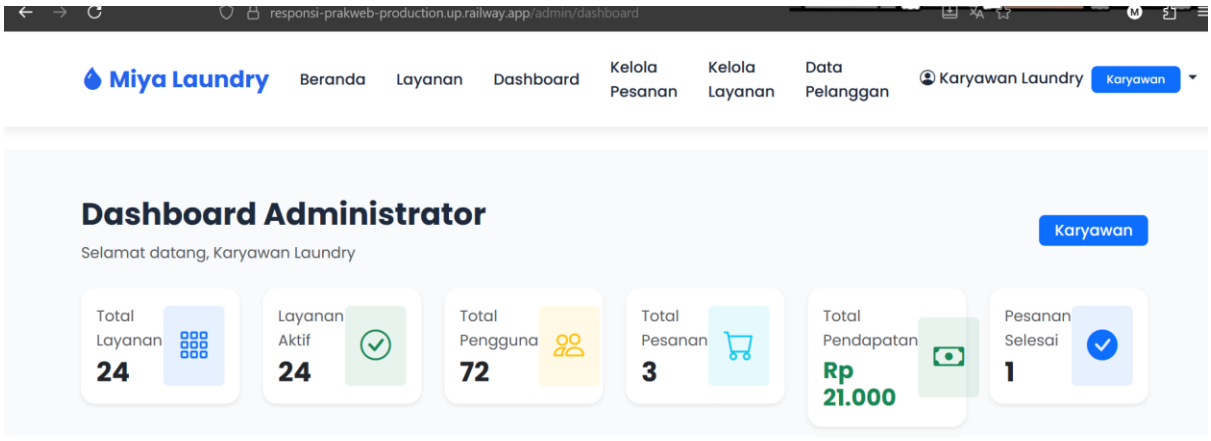
Riwayat pesanan



Halaman pesanan saya / riwayat pesanan

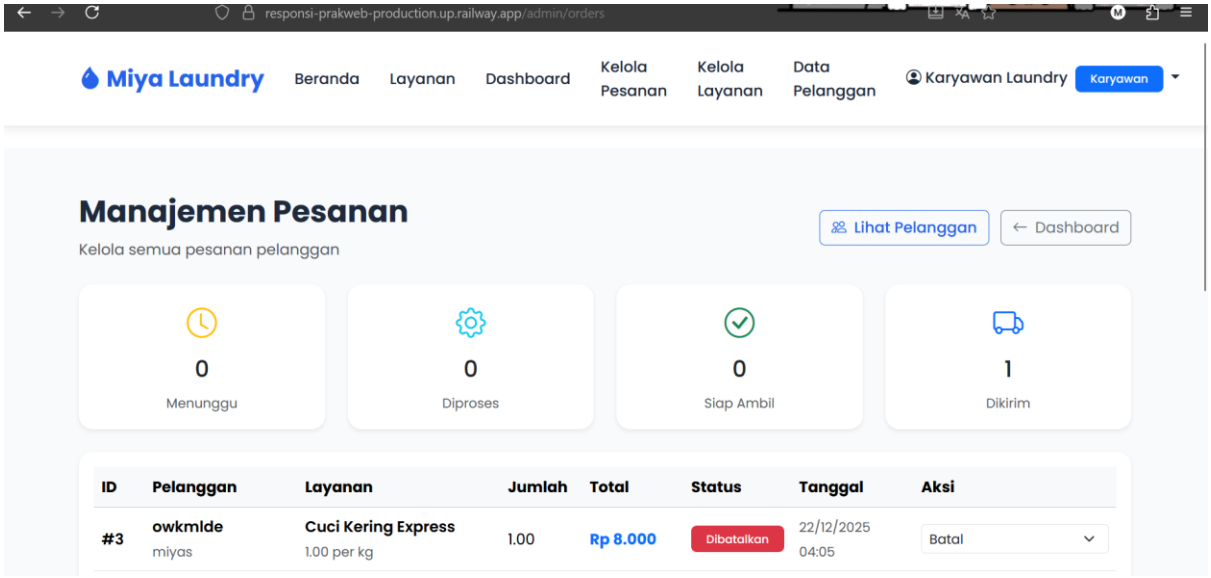
B fitur-fitur karyawan

Dashboard administrator



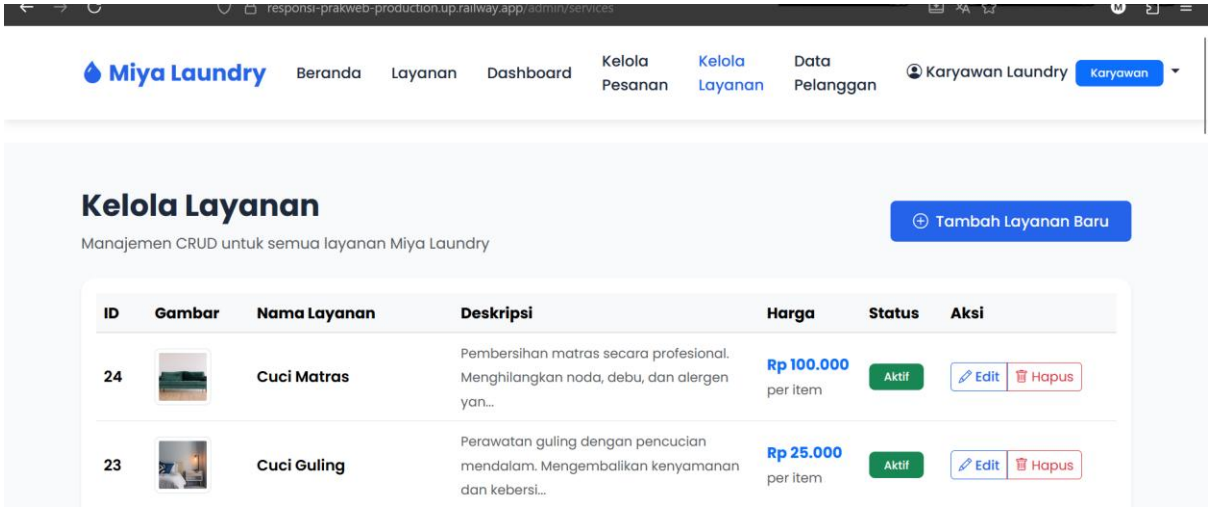
Halaman dashboard administrator

Kelola pesanan



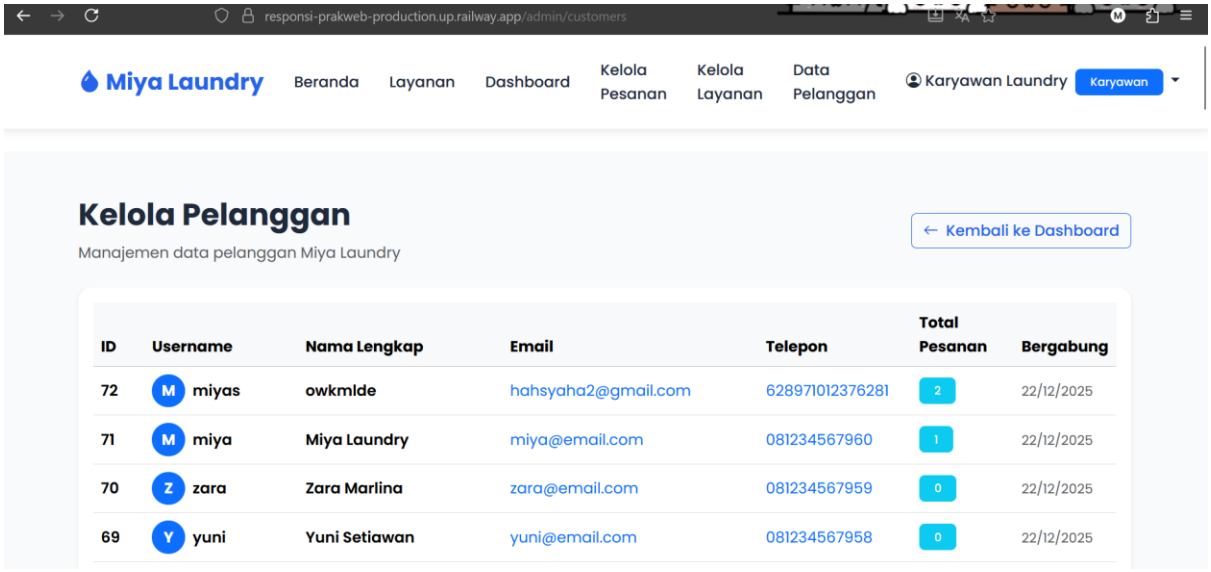
Halaman kelola pesann khusus karyawan/admiistrator

Kelola layanan



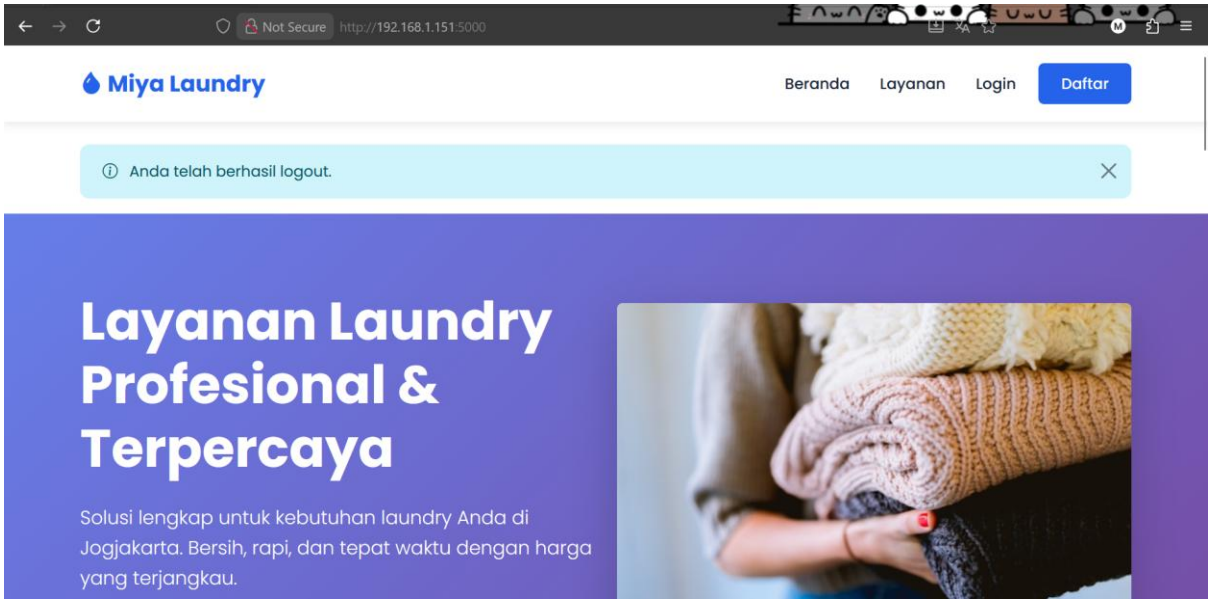
Halaman kelola layanan khusus karyawan/admiistrator

Kelola user / kelola pelanggan

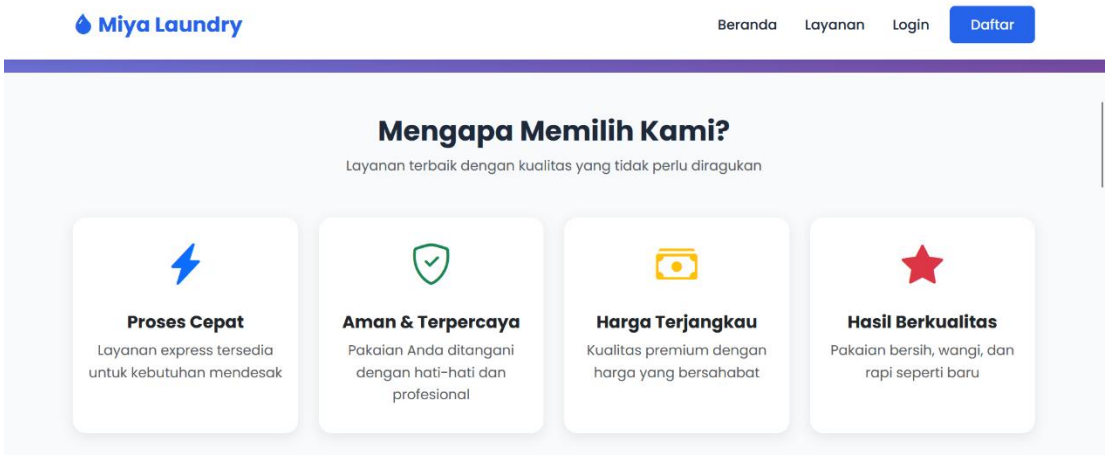


Halaman kelola user pelanggan

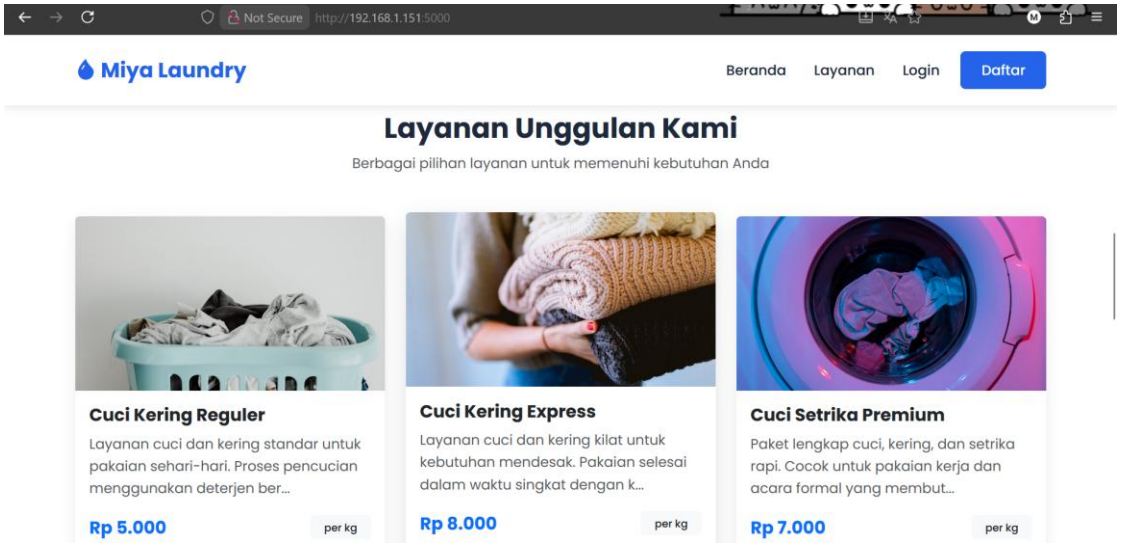
Beranda



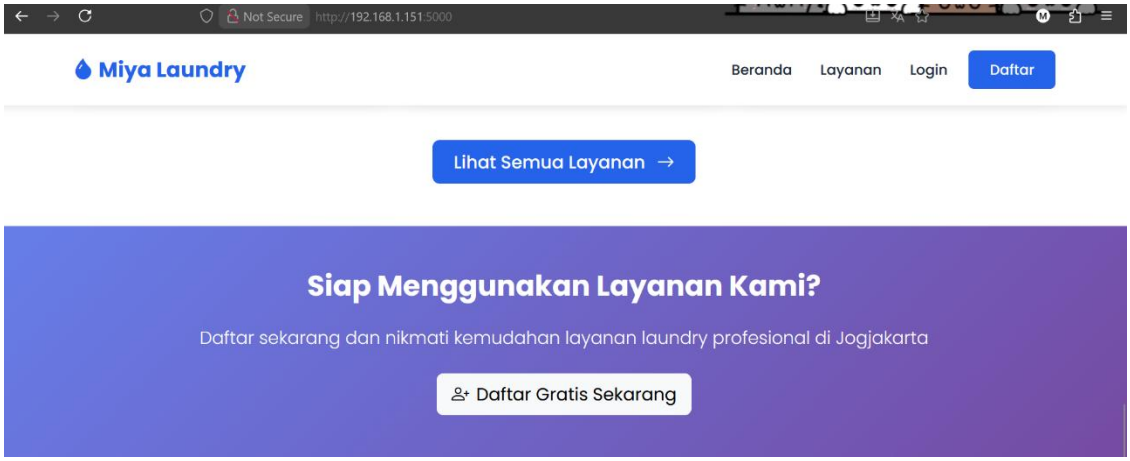
Beranda tampilan web sebelum autentkasi



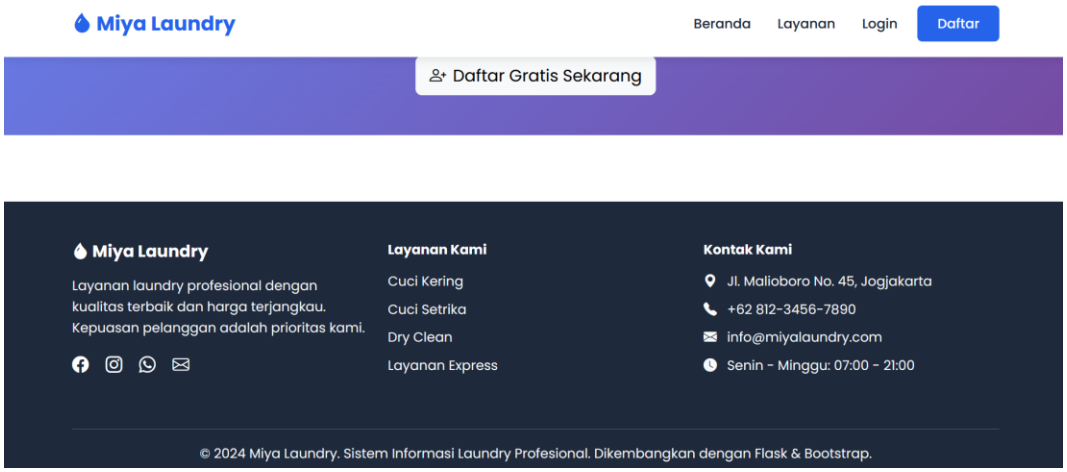
session 2



session 3

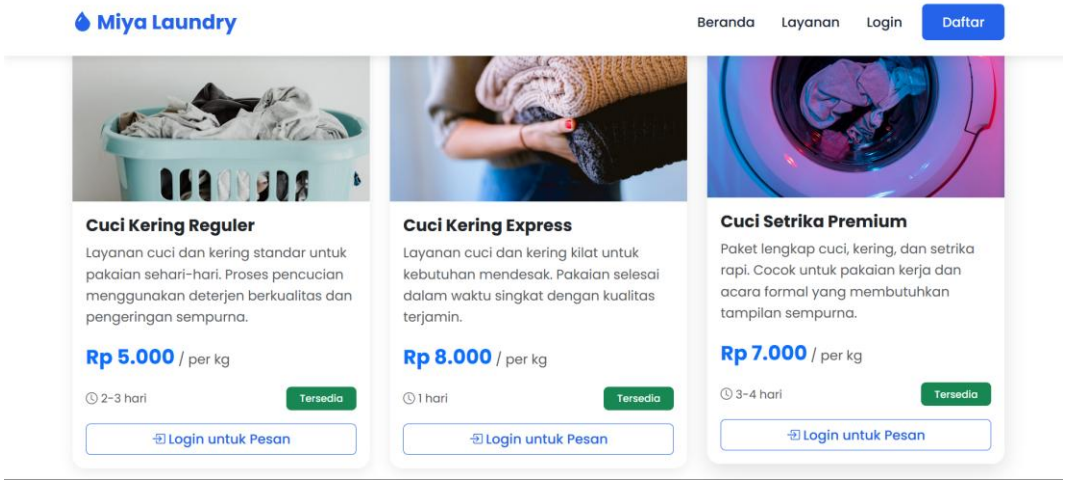


session 4



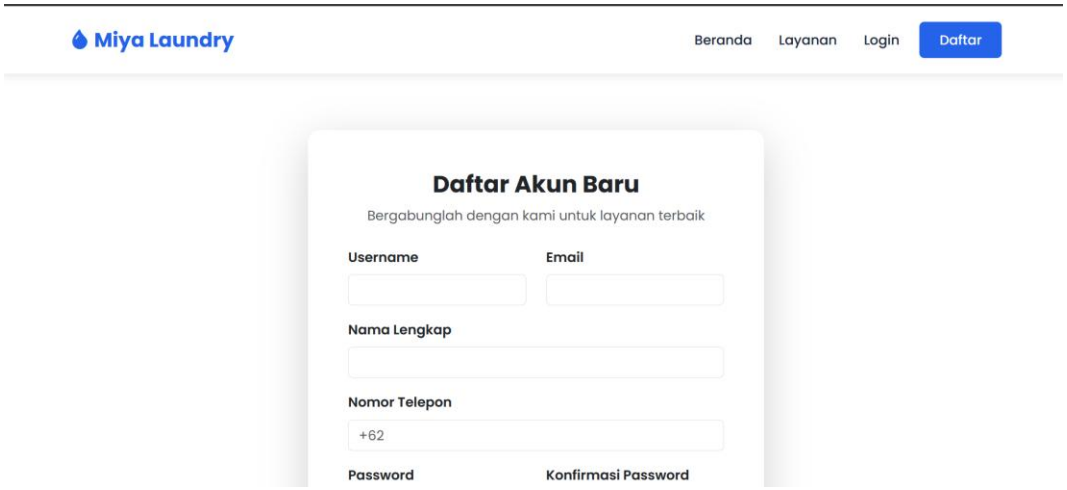
session 5

Layanan

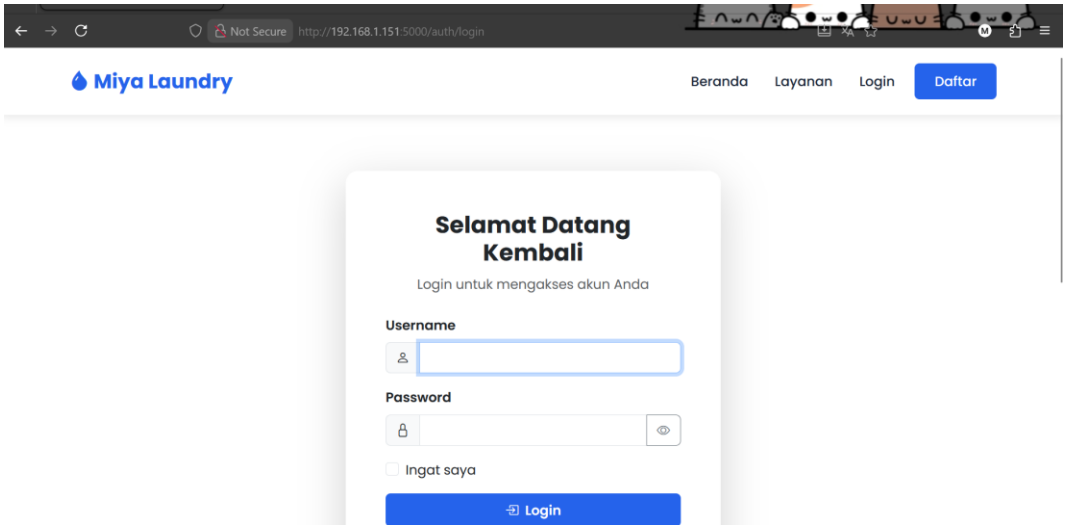


2. Proses autentikasi

Daftar akun baru

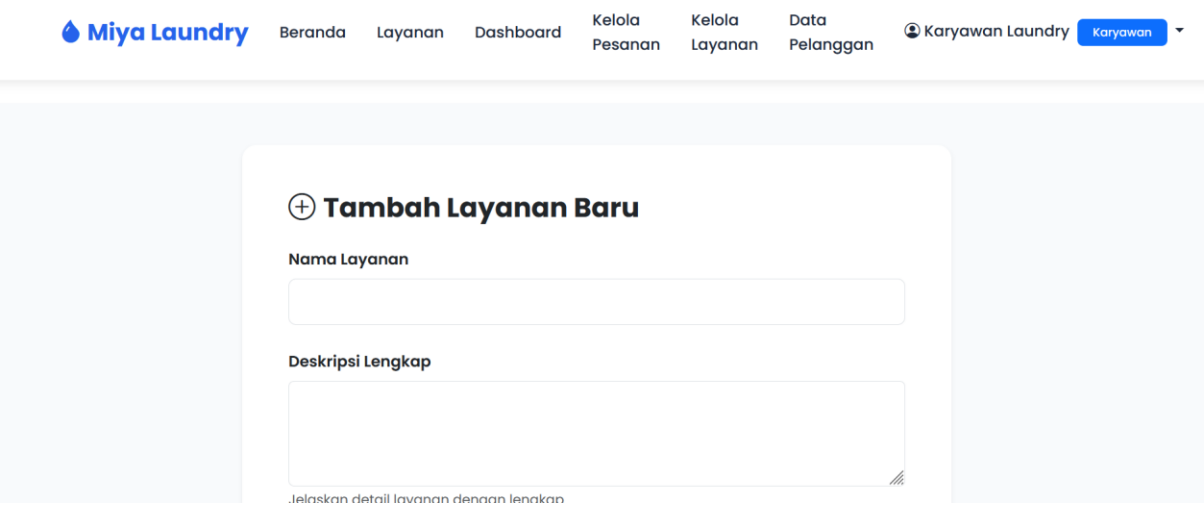


Login



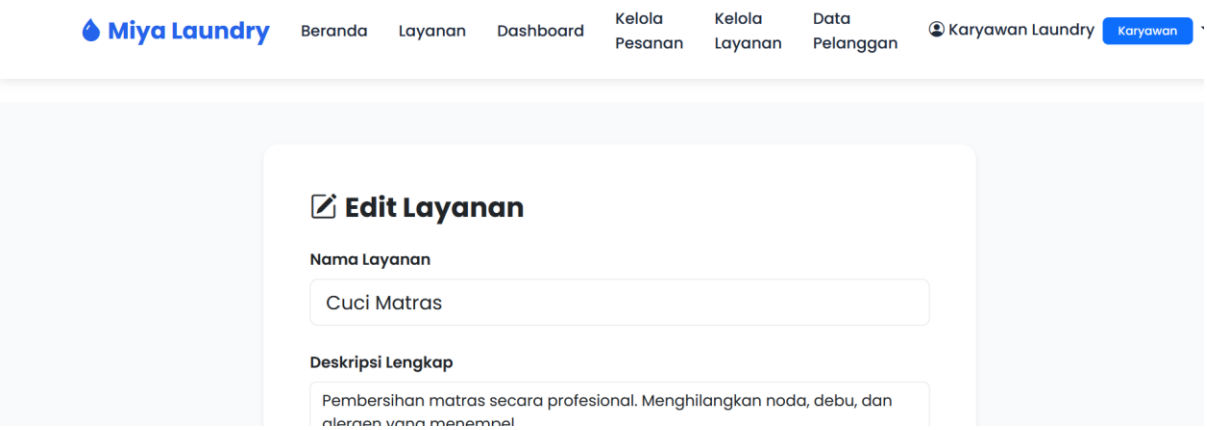
3. Fitur-fitur CRUD

Create menambah layanan baru dengan detail lengkap



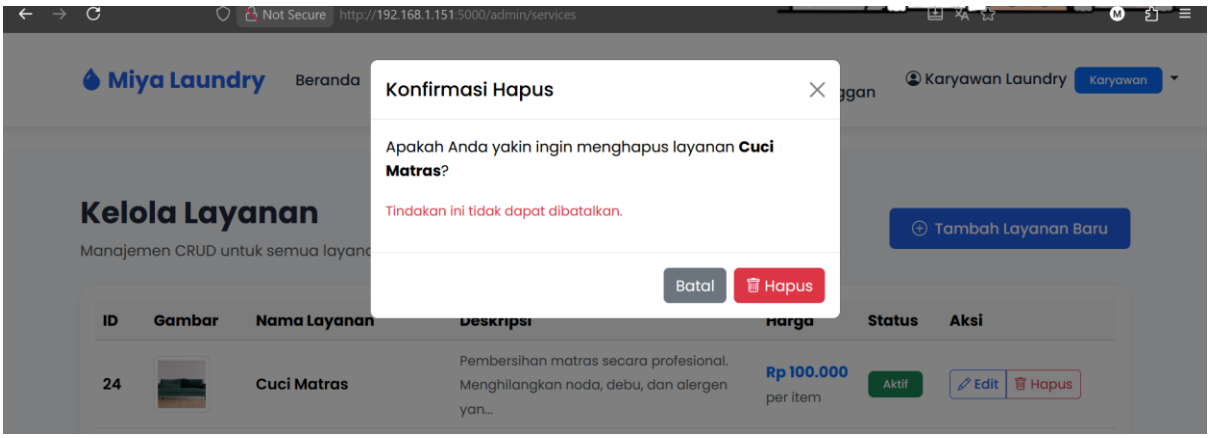
Contoh create

Update mengedit harga, deskripsi, satuan layanan existing



Fitur update

Delete mengahapus data yang tidak diperlukan misal menghapus layanan, user dll

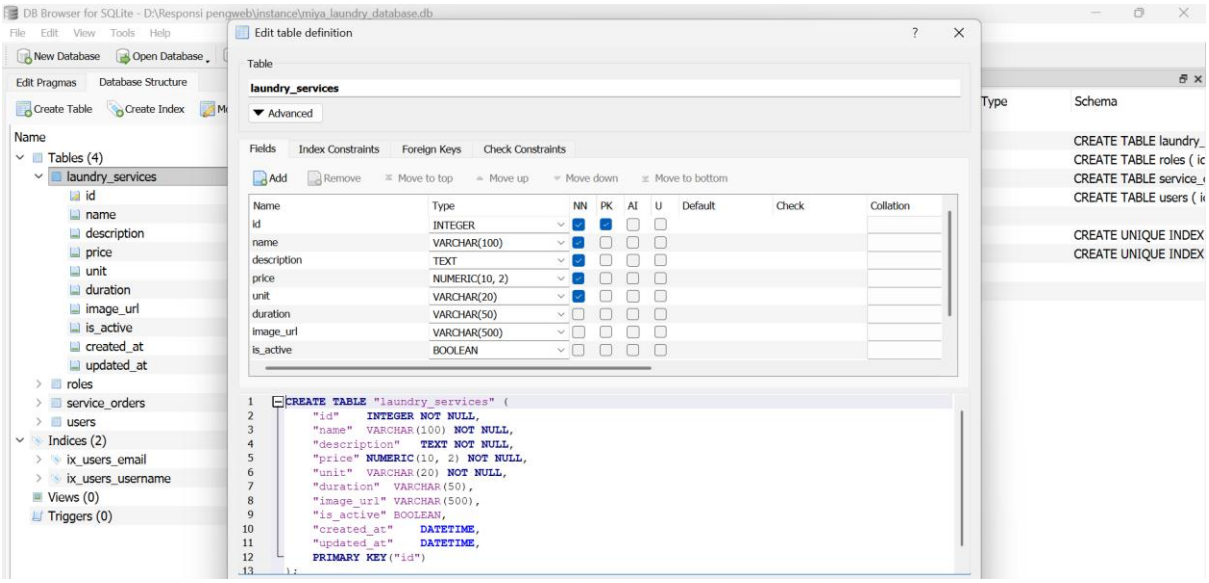


fitur delete

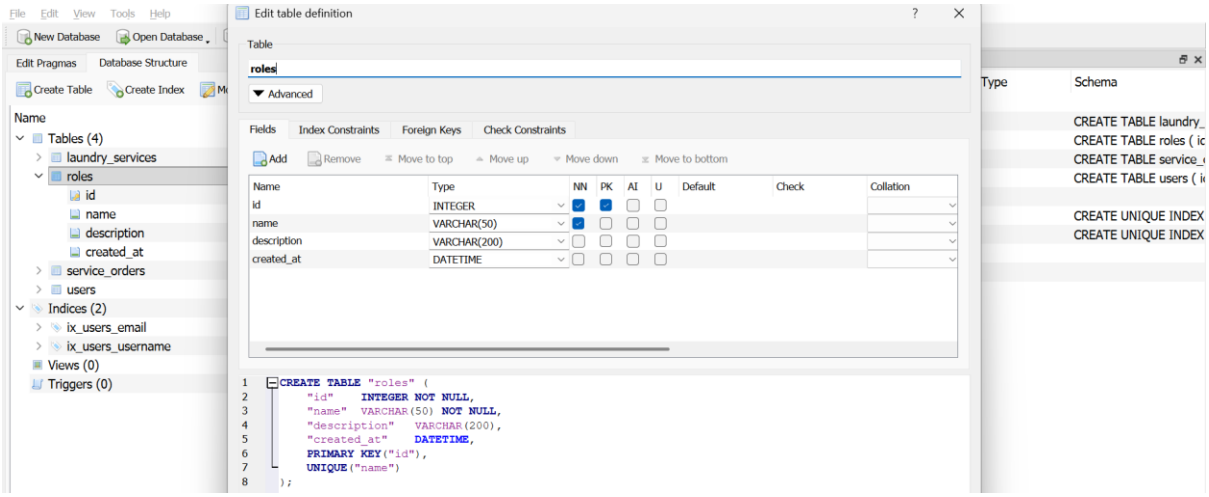
4. Database rasional

Database yang telah dibuat. disini saya menggunakan database sqllite karena ia berbasis file sehingga memudahkan untuk mendevloy secara gratis

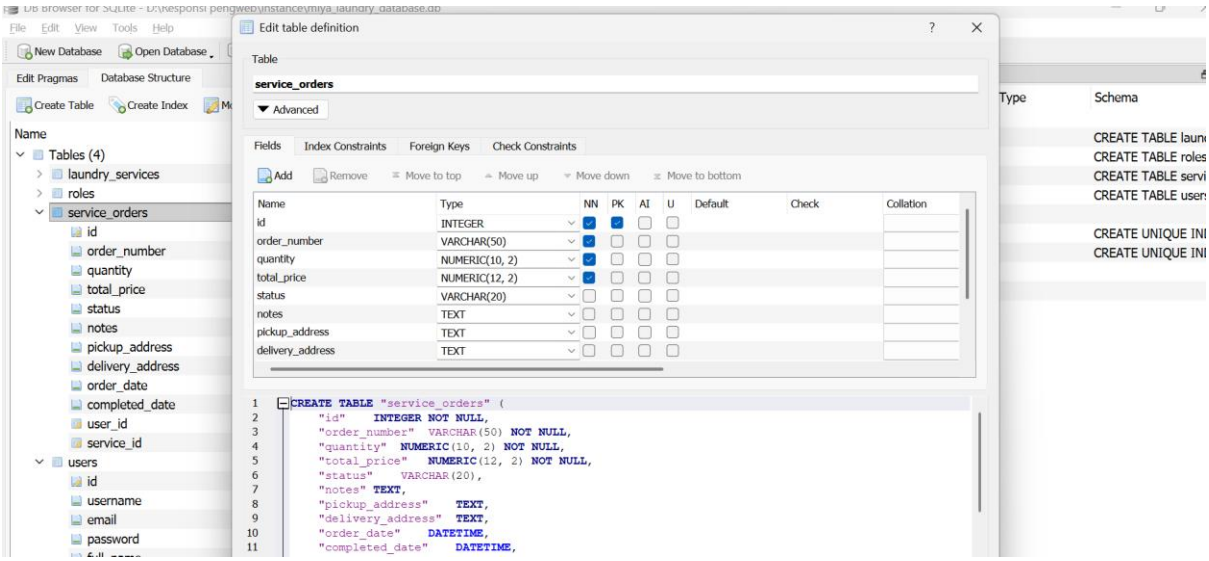
A laundry_service



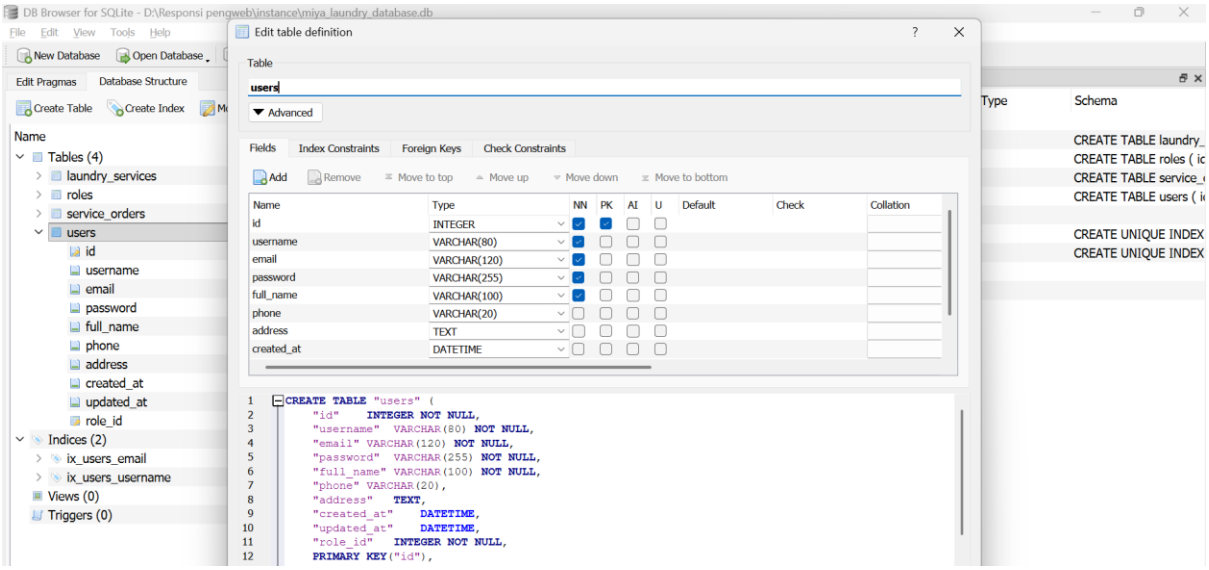
B roles



C service_orders



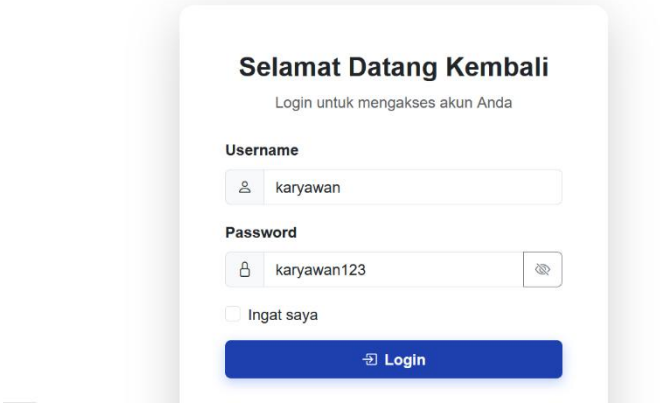
D user

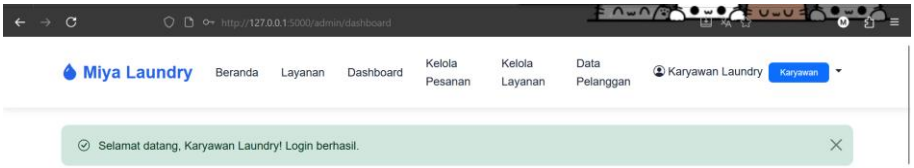


5. Otorisasi

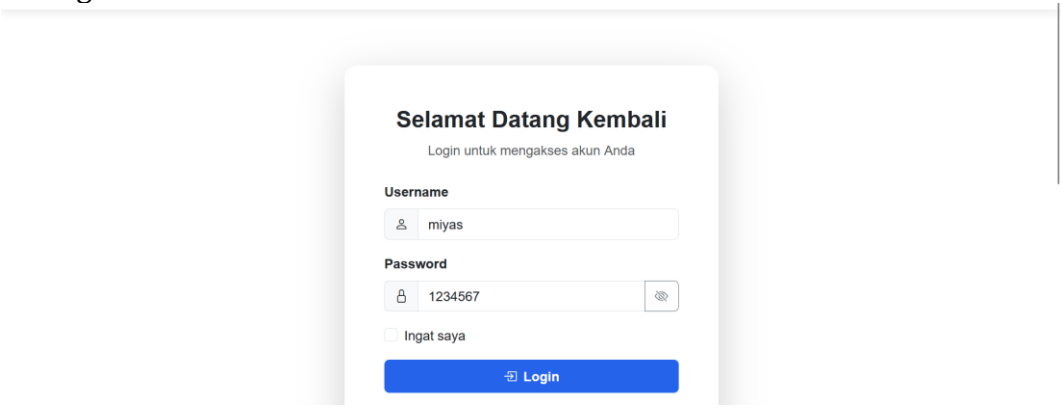
Terdapat 2 role yaitu karyawan dan costumer

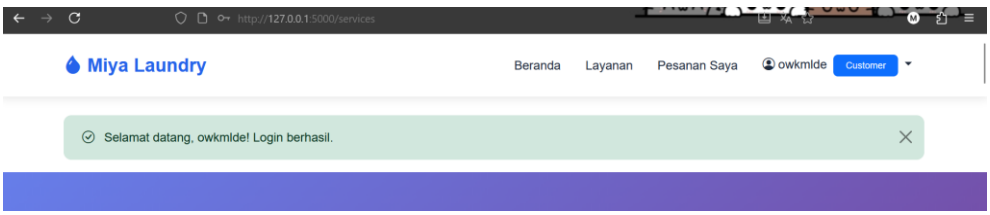
Log in sebagai karyawan





Login sebagai costumer





BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Praktikum Sistem Manajemen Laundry ini berhasil mencapai tujuan pembelajaran yang telah ditetapkan. Berikut adalah pencapaian utama:

1. Virtual Environment: Berhasil mengimplementasikan isolasi dependency menggunakan venv, memastikan tidak ada konflik dengan package sistem.

2. Database Relasional: Merancang 4 tabel (roles, users, laundry_services, service_orders) dengan relasi one-to-many yang tepat menggunakan foreign key.

3. Operasi CRUD Mengimplementasikan Create, Read, Update, Delete untuk layanan dan pesanan dengan validasi input yang proper.

4. Autentikasi: Membangun sistem login/logout dengan password hashing menggunakan Werkzeug, session management dengan Flask-Login.

5. Otorisasi: Menerapkan Role-Based Access Control (RBAC) dengan 2 role (customer dan karyawan/admin) dengan pembatasan akses yang jelas.

6. Responsive Design: Membangun interface menggunakan Bootstrap 5 yang responsif di berbagai ukuran layar.

Hambatan yang Dihadapi:

- Konfigurasi foreign key di SQLite yang tidak auto-increment
- Deployment ke Railway memerlukan migrasi dari SQLite ke PostgreSQL
- Debugging session management untuk multi-role access

Pembelajaran Penting:
Praktikum ini memberikan pemahaman mendalam tentang full-stack web development, dari perancangan database hingga deployment. Pengalaman ini sangat berharga untuk pengembangan aplikasi web production-ready di masa depan.

5.2 Saran

1. Pengembangan fitur payment gateway
2. Implementasi notification system (email/SMS)
3. Dashboard analytics untuk admin
4. Mobile application untuk customer

DAFTAR PUSTAKA

Flask Documentation. (2024). Flask Web Development.
https://flask.palletsprojects.com/

Grinberg, M. (2024). Flask Web Development: Developing Web Applications with Python. O'Reilly Media.

Bootstrap Documentation. (2024). Bootstrap 5.3.
https://getbootstrap.com/docs/5.3/

SQLAlchemy Documentation. (2024). SQLAlchemy ORM Tutorial.
https://docs.sqlalchemy.org/