UJIAN AKHIR SEMESTER MATA KULIAH PEMROGRAMAN WEB

CAMPUS INTERN (WEBSITE INFORMASI LOWONGAN MAGANG BERBASIS PHP NATIVE)

Kelas : 2024 A

Kelompok : 5 (Lima)

Nama Anggota :

Zaryan Nugraha Islah (24091397010)

Nazzala Risky Nafi'a (24091397009)

Randy Pradana Bintang O. (24091397018)

Video Presentasi :

https://youtu.be/JP1d8zmxCOA

Sumber Kode Program :

https://github.com/Zaryan19/CampusIntern_ProjectUASPemWeb.git

PROGRAM STUDI D4 MANAJEMEN INFORMATIKA

FAKULTAS VOKASI

UNIVERSITAS NEGERI SURABAYA

TAHUN 2025

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Di era digital saat ini, mahasiswa dan fresh graduate semakin membutuhkan akses mudah terhadap peluang magang (internship) untuk mengembangkan keterampilan, pengalaman kerja, dan jaringan profesional. Kampus sering kali menjadi pusat penghubung antara mahasiswa dengan dunia industri melalui program internship. Namun, proses pencarian dan pendaftaran magang sering kali masih manual, terfragmentasi, dan kurang efisien. Banyak mahasiswa kesulitan menemukan daftar perusahaan terkenal (PT) yang menawarkan program magang, terutama yang sesuai dengan bidang studi mereka. Proyek "Campus Intern" hadir sebagai solusi berbasis web yang menyediakan platform terintegrasi untuk daftar perusahaan terkenal di Indonesia dan internasional. Web ini memungkinkan pengguna (mahasiswa) untuk browsing daftar PT, melihat detail program magang, dan langsung mendaftar melalui sistem online. Latar belakang proyek ini didasari oleh tren peningkatan kebutuhan tenaga kerja muda yang terampil, di mana data dari Kementerian Pendidikan dan Kebudayaan menunjukkan bahwa lebih dari 70% mahasiswa membutuhkan pengalaman magang untuk meningkatkan employability. Selain itu, pandemi COVID-19 telah mempercepat adopsi teknologi digital dalam proses rekrutmen, sehingga platform seperti ini menjadi semakin relevan untuk menghubungkan kampus dengan industri.

1.2 Dasar Permasalahan

Proses pendaftaran magang di kampus sering menghadapi beberapa tantangan utama:

1. Kurangnya Informasi Terpusat: Mahasiswa harus mencari informasi magang secara manual melalui situs perusahaan, email kampus, atau media sosial, yang memakan waktu dan sering kali tidak lengkap. Daftar PT terkenal seperti Google, Unilever, atau PT Pertamina sering tersebar dan sulit diakses dalam satu tempat.

2. Proses Pendaftaran yang Rumit: Banyak perusahaan mengharuskan pengiriman CV, surat lamaran, dan dokumen lain melalui email atau portal terpisah, yang rentan terhadap kesalahan dan keterlambatan. Selain itu, tidak ada sistem tracking status pendaftaran yang real-time.
3. Kurangnya Personalisasi: Pengguna sering tidak mendapatkan rekomendasi magang berdasarkan profil mereka (seperti jurusan, IPK, atau minat), sehingga banyak peluang yang terlewat.
4. Aksesibilitas Rendah: Mahasiswa di daerah terpencil atau kampus kecil sering kesulitan mengakses program magang dari PT besar karena keterbatasan informasi dan jaringan.
5. Kurangnya Integrasi dengan Kampus: Tidak ada kolaborasi langsung antara sistem kampus dengan perusahaan, sehingga verifikasi data mahasiswa menjadi lambat dan tidak efisien.

Permasalahan ini menyebabkan tingkat partisipasi magang rendah, di mana survei internal kampus menunjukkan hanya 40% mahasiswa yang berhasil magang sebelum lulus, yang berdampak pada kesiapan kerja mereka di pasar tenaga kerja yang kompetitif.

1.3 Tujuan Project

Proyek Campus Intern bertujuan untuk mengatasi permasalahan di atas melalui pengembangan web platform yang inovatif. Tujuan utama proyek ini meliputi:

1. **Menyediakan Daftar PT Terkenal yang Terintegrasi**: Membuat database lengkap daftar perusahaan terkenal beserta program magang mereka, yang dapat diakses secara gratis dan mudah oleh pengguna.
2. **Memfasilitasi Proses Pendaftaran Online**: Mengembangkan fitur pendaftaran magang langsung melalui web, termasuk upload dokumen, tracking status, dan notifikasi real-time untuk meningkatkan efisiensi.

3. **Memberikan Rekomendasi Personalisasi**: Menggunakan algoritma sederhana (seperti matching berdasarkan profil pengguna) untuk merekomendasikan program magang yang sesuai, sehingga meningkatkan relevansi dan partisipasi.
4. **Meningkatkan Kolaborasi Kampus-Industri**: Integrasi dengan sistem kampus untuk verifikasi data dan kerjasama dengan PT, guna memastikan proses yang aman dan terpercaya.
5. **Meningkatkan Aksesibilitas dan Inklusivitas**: Membuat platform yang mobile-friendly dan mendukung bahasa Indonesia/Inggris, sehingga dapat diakses oleh mahasiswa dari berbagai latar belakang dan wilayah.

Dengan mencapai tujuan ini, proyek diharapkan dapat meningkatkan tingkat magang mahasiswa hingga 60% dalam satu tahun pertama peluncuran, serta menjadi model untuk platform serupa di institusi pendidikan lainnya.

BAB II

ANALISIS SISTEM

2.1 Daftar Pengguna

Sistem informasi lowongan magang ini memiliki tiga jenis pengguna utama, yaitu Admin, Perusahaan, dan user. Setiap pengguna memiliki hak akses dan fungsi yang berbeda sesuai dengan perannya masing-masing.

A. Admin

Administrator merupakan pengguna dengan hak akses tertinggi dalam sistem.

Tugas dan hak akses Administrator meliputi:

1. Mengelola seluruh data pengguna (perusahaan dan user)
2. Melihat seluruh lowongan magang yang tersedia di sistem
3. Mengaktifkan dan memblokir akun Perusahaan maupun user
4. Menjaga keamanan dan kelancaran sistem

Administrator berperan penting dalam memastikan bahwa sistem berjalan dengan baik serta lowongan magang yang tersedia sesuai dengan ketentuan yang berlaku.

B. Perusahaan

Perusahaan merupakan pengguna yang berperan sebagai penyedia lowongan magang.

Tugas dan hak akses Perusahaan meliputi:

1. Melakukan login sebagai akun perusahaan
2. Mengelola profil perusahaan
3. Menambahkan lowongan magang
4. Mengubah dan menghapus lowongan magang

Apabila perusahaan diblokir oleh Administrator, maka seluruh lowongan magang yang dimiliki perusahaan tersebut tidak dapat diakses oleh user.

C. User

User merupakan pengguna yang berperan sebagai pencari lowongan magang.

Tugas dan hak akses user meliputi:

1. Melakukan registrasi dan login akun
2. Melihat daftar lowongan magang yang aktif
3. Melihat detail informasi lowongan magang
4. Melamar lowongan magang yang tersedia

User hanya dapat mengakses lowongan magang dari perusahaan yang berstatus aktif.

2.1 Daftar Fitur

Website campus intern memiliki berbagai fitur untuk masing masing role yang berbeda. Fitur fitur yang tersedia dirancang untuk memenuhi kebutuhan serta tujuan dari website campus intern. Fitur yang ada dalam website ini antara lain :

A. Admin

Admin merupakan role tertinggi dari website karena memiliki wewenang untuk mengatur website seperti melakukan sistem banned akun, ini dilakukan guna menciptakan lingkungan website yang aman dan tertib. Untuk role admin menyediakan fitur,

1. Login

   Admin mampu melakukan login dengan akun yang sudah dibuat oleh super admin.
2. Manajemen akun

   Melalui dashboard admin dapat melihat jumlah semua user seperti akun pengguna dan akun perusahaan.
3. Banned akun

   Admin dapat melakukan banned pada akun tertentu yang merupakan pelanggar pada website ini. Banned dapat diberikan kepada seluruh akun seperti pengguna atau perusahaan. Ketika akun telah dibanned maka user tidak bisa melakukan login pada website. Admin juga dapat melakukan active akun kembali setelah di banned.

4. Melihat daftar lowongan

   Admin dapat melihat seluruh daftar lowongan yang telah dibuat oleh perusahaan. Masing masing lowongan dapat dilihat secara detail dari informasi hingga status.

a. User

User sebagai pendaftar magang yang dapat mengakses dan mendaftar lowongan magang dari perusahaan. Mempunyai fitur yakni

1. Login dan register

   Mampu untuk melakukan register ketika user belum memiliki akun untuk pertama kali. Setelah memiliki akun, user dapat melakukan login untuk mendaftar lowongan magang.
2. Tampilan daftar lowongan

   User dapat melihat semua daftar lowongan magang yang tersedia, setiap lowongan magang terdiri dari informasi lebih detail beserta terdapat tombol ‘daftar’ untuk mendaftar pada lowongan magang terkait.
3. Daftar magang

   User dapat mendaftarkan diri melalui tombol daftar pada setiap lowongan magang yang tersedia. Akan diarahkan pada menu pendaftaran magang yang berisi berbagai macam pertanyaan yang bisa dijawab dimulai dari data diri, riwayat pendidikan, hingga melampirkan portofolio atau CV. Setelah semua kolom terisi, user bisa mengirim kannya dan akan menampilkan notifikasi berhasil. Hasil diterima atau tidaknya akan dikirim melalui email masing masing user.

b. Perusahaan

Perusahaan merupakan role yang menyediakan lowongan magang yang memiliki fitur berbeda dari role lain, yakni :

1. Login atau register

   Perusahaan dapat melakukan register untuk pertama kalinya, memilih role ‘company’ yang menandakan bahwa akun tersebut merupakan akun company. Setelah melakukan register maka dapat melakukan login memakai akun yang telah terdaftar.

2. Dashboard company

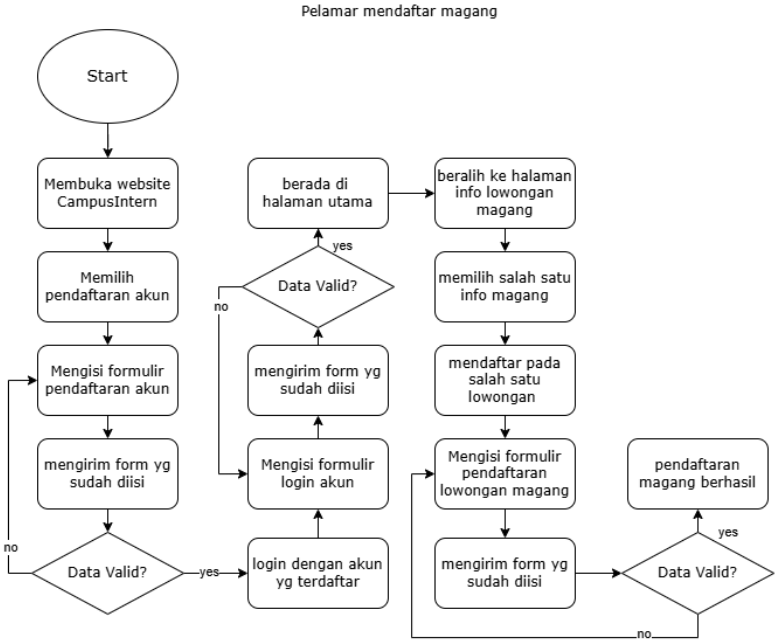
   Dalam dashboard role perusahaan dapat melakukan menambahkan lowongan magang, mengedit, serta menghapus lowongan yang ada. Fitur menambahkan lowongan dilakukan dengan cara mengisi kolom field yang kosong seperti mengisi deksripsi, jurusan, ketentuan serta batas waktu pendaftaran. Setelah lowongan berhasil dibuat, lowongan tersebut dapat di edit atau dihapus.

BAB III

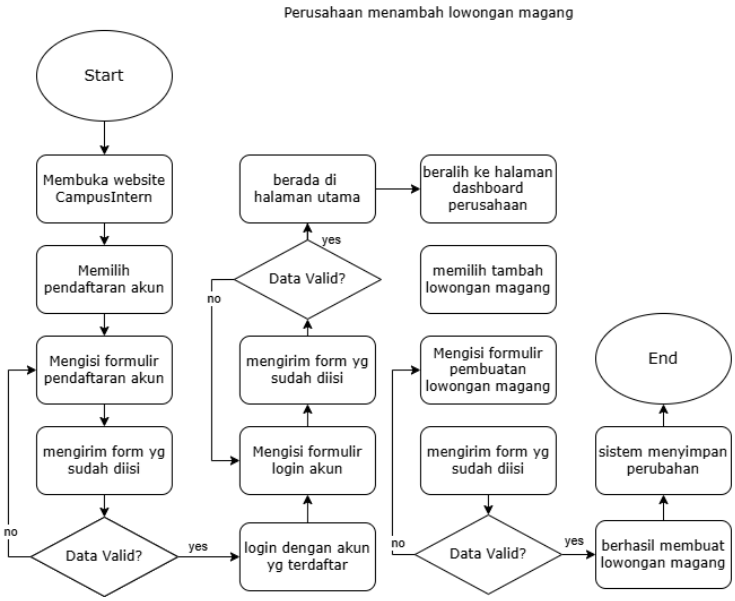
PERANCANGAN SISTEM

3.1 Flow Chart

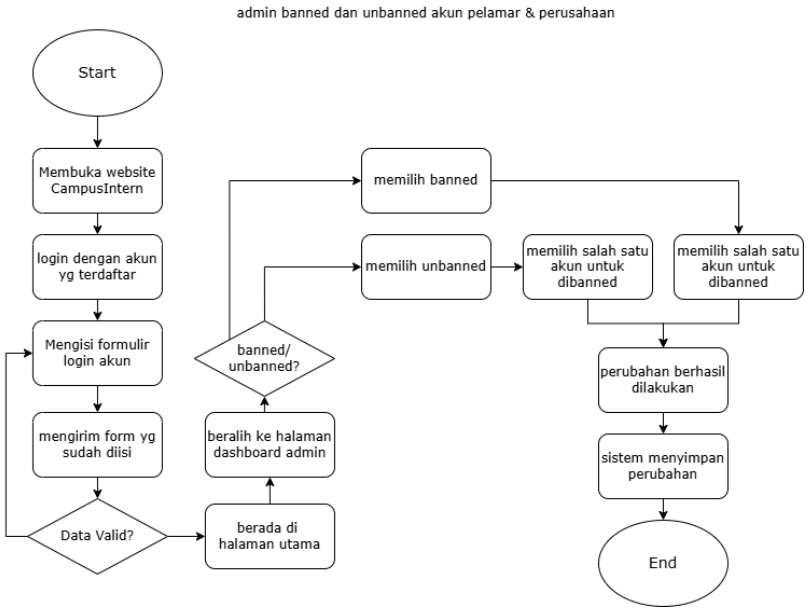
3.1.1 Pelamar Mendaftar Lowongan Magang



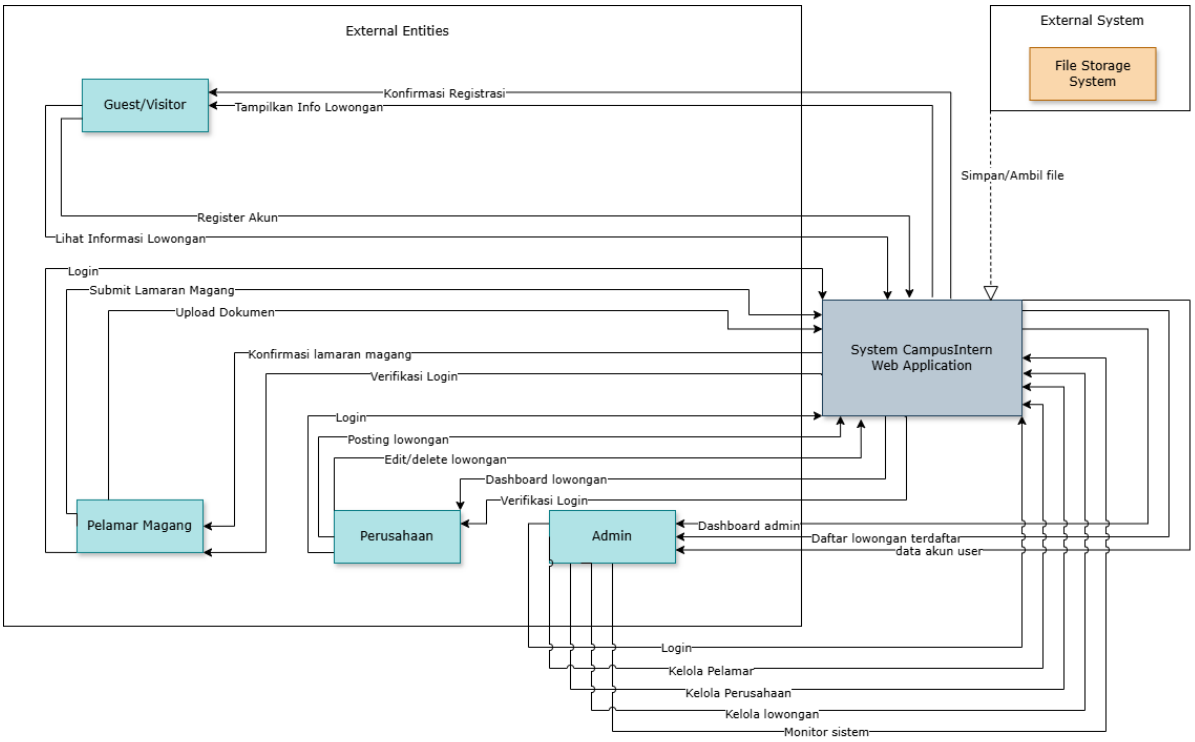
3.1.2 Perusahaan Menambah Lowongan Magang



3.1.3 Admin Mengelola Akun Pelamar & Perusahaan (Banned/Unbanned)



3.2 Context Diagram



1. Identifikasi Entitas Eksternal

A. Guest / Visitor

Guest merupakan pengguna yang belum memiliki akun.

Interaksi utama:

- Melihat informasi lowongan magang
- Melakukan registrasi akun

Respon sistem:

- Menampilkan informasi lowongan yang tersedia
- Memberikan konfirmasi registrasi

Peran Guest dibatasi untuk menjaga keamanan sistem dan mendorong autentikasi sebelum akses lanjutan.

B. Pelamar Magang (Applicant)

Applicant adalah pengguna yang telah terdaftar dan login.

Interaksi utama:

- Login ke sistem
- Mengirim lamaran magang
- Mengunggah dokumen pendukung (CV, surat lamaran, portofolio)

Respon sistem:

- Verifikasi login
- Konfirmasi keberhasilan pengajuan lamaran

Pada tahap ini, sistem mulai mengelola data personal dan dokumen, sehingga autentikasi menjadi syarat utama.

C. Perusahaan

Perusahaan berperan sebagai penyedia lowongan magang.

Interaksi utama:

- Login ke sistem
- Membuat (posting) lowongan magang
- Mengedit atau menghapus lowongan

Respon sistem:

- Menyediakan dashboard untuk mengelola lowongan

Pemisahan peran perusahaan memastikan bahwa hanya pihak terkait yang dapat memodifikasi data lowongan.

D. Admin

Admin memiliki hak akses tertinggi dalam sistem.

Interaksi utama:

- Login ke sistem
- Mengelola pengguna
- Mengelola data perusahaan
- Mengelola lowongan magang
- Memantau sistem secara keseluruhan

Respon sistem:

- Menyediakan dashboard admin
- Menyajikan data akun user
- Menyediakan data lowongan terdaftar

Peran admin penting untuk menjaga integritas data, stabilitas sistem, dan pengawasan operasional.

2. Sistem Utama

Sistem Informasi Manajemen Magang (Web Application) berfungsi sebagai pusat pengolahan data dan interaksi.

Tanggung jawab sistem meliputi:

- autentikasi dan otorisasi pengguna,
- pengelolaan data lowongan dan lamaran,
- penyimpanan dokumen,
- penyajian informasi dan laporan.

Diagram menegaskan batas yang jelas antara sistem dan pihak luar.

3. Sistem Eksternal

File Storage System

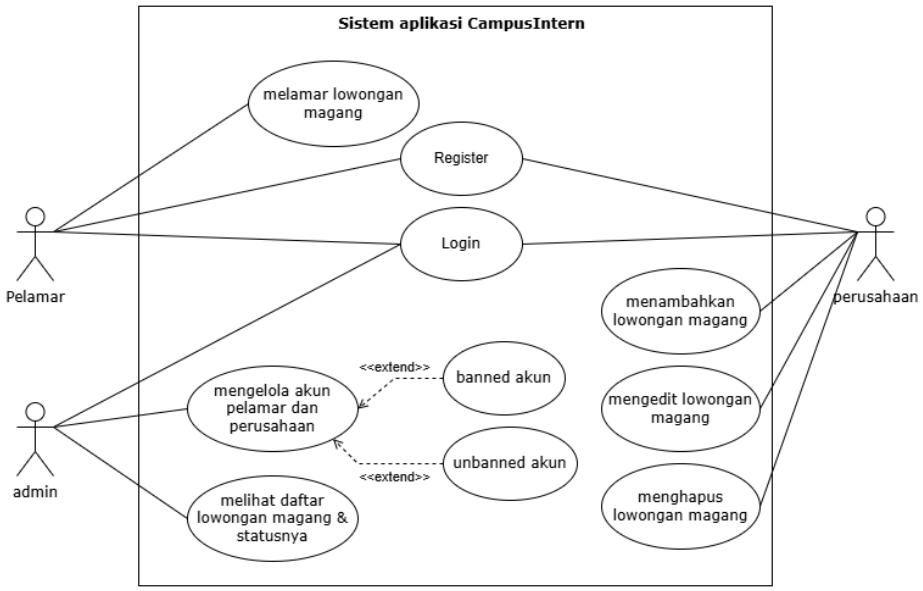
File Storage System merupakan sistem pendukung eksternal.

Fungsi utama:

- Menyimpan dan mengambil dokumen lamaran
- Menangani file upload dan retrieval

Interaksi bersifat tidak langsung (dotted line), menandakan ketergantungan fungsional tanpa integrasi logika bisnis utama.

3.3 Use Case Diagram



3.4 Architecture Diagram

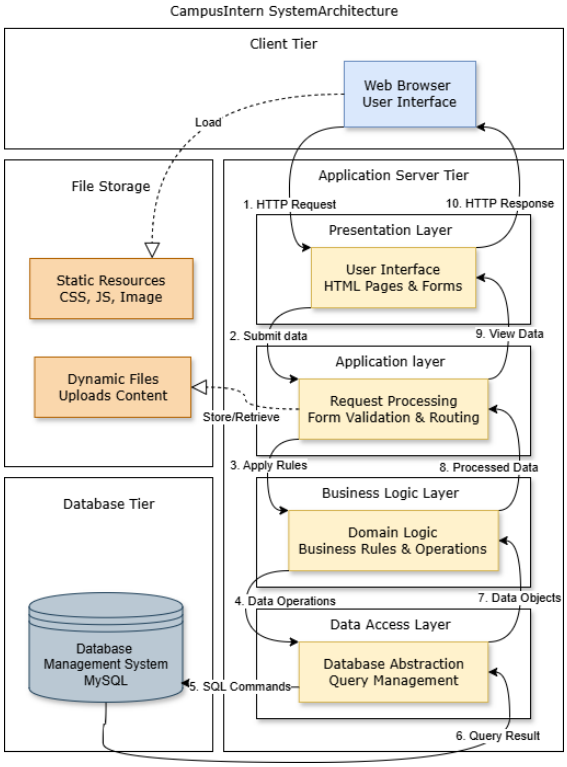
1. Analisis Arsitektur Sistem

Diagram ini merepresentasikan arsitektur aplikasi web berbasis multi-tier (berlapis) yang memisahkan sistem ke dalam beberapa lapisan tanggung jawab. Tujuan utama pendekatan ini adalah:

- meningkatkan maintainability (kemudahan perawatan),
- memperjelas pemisahan tugas (separation of concerns),
- serta meningkatkan skalabilitas dan keamanan sistem.

Arsitektur dibagi menjadi Client Tier, Application Server Tier, Database Tier, dan File Storage.

3. Penjelasan Tiap Tier dan Layer



2.1 Client Tier

Client Tier direpresentasikan oleh *Web Browser*.

- Berfungsi sebagai antarmuka interaksi pengguna.
- Mengirimkan HTTP Request ke server dan menerima HTTP Response.
- Tidak memiliki logika bisnis; hanya menampilkan data dan mengirim input pengguna.

Peran client dibatasi agar tidak terjadi ketergantungan logika pada sisi pengguna, yang dapat berisiko terhadap keamanan.

2.2 Application Server Tier

Tier ini merupakan inti sistem dan dibagi menjadi beberapa layer untuk menjaga modularitas.

a. Presentation Layer

- Mengelola User Interface berupa halaman HTML dan form.
- Bertugas menampilkan data yang telah diproses oleh sistem.
- Menjadi penghubung langsung antara pengguna dan logika aplikasi.

Layer ini tidak melakukan pemrosesan data atau aturan bisnis.

b. Application Layer

- Bertanggung jawab terhadap pemrosesan request, validasi form, dan routing.
- Mengontrol alur kerja aplikasi (workflow).
- Menentukan logika aplikasi mana yang dijalankan berdasarkan permintaan pengguna.

Layer ini berperan sebagai orchestrator, bukan pengambil keputusan bisnis.

c. Business Logic Layer

- Mengandung aturan bisnis utama dan operasi domain.
- Menentukan bagaimana data diproses sesuai kebutuhan sistem.
- Menjaga agar aturan bisnis tetap konsisten meskipun antarmuka atau database berubah.

Pemisahan ini penting agar perubahan kebijakan bisnis tidak berdampak langsung pada tampilan atau database.

d. Data Access Layer

- Menyediakan abstraksi terhadap database.
- Mengelola query dan komunikasi dengan DBMS.
- Mencegah layer lain berinteraksi langsung dengan database.

Layer ini meningkatkan keamanan dan memudahkan migrasi database di masa depan.

2.3 Database Tier

Database Tier menggunakan Database Management System (MySQL).

- Menyimpan data secara persisten.
- Menjalankan perintah SQL dari Data Access Layer.
- Tidak berinteraksi langsung dengan client atau UI.

Pemutusan akses langsung dari client ke database merupakan praktik standar untuk mencegah kebocoran data.

2.4 File Storage

File Storage dipisahkan menjadi dua jenis:

1. Static Resources
    - Berisi file CSS, JavaScript, dan gambar.
    - Diakses langsung oleh browser untuk meningkatkan performa.
2. Dynamic Files
    - Berisi file hasil upload atau konten yang dihasilkan sistem.
    - Diakses melalui logika aplikasi untuk menjaga kontrol dan keamanan.

3. Alur Request dan Response

3.1 Alur Request

1. Browser mengirim HTTP Request ke Presentation Layer.
2. Data dari form diteruskan ke Application Layer.
3. Business Logic Layer menerapkan aturan bisnis.
4. Data Access Layer menyiapkan query.

5. Database mengeksekusi perintah SQL.

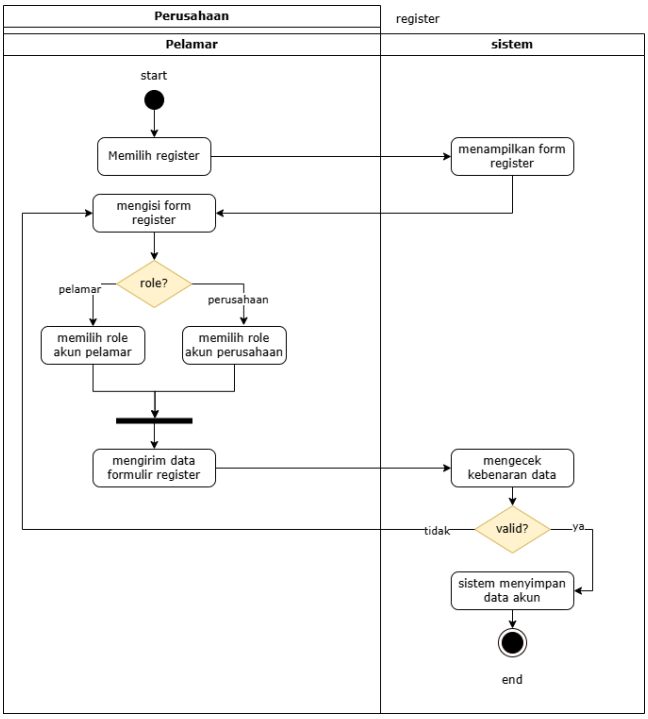
3.2 Alur Response

6. Database mengembalikan hasil query.
7. Data diteruskan ke Business Logic Layer.
8. Data diproses sesuai kebutuhan aplikasi.
9. Application Layer menyiapkan data untuk tampilan.
10. Presentation Layer mengirim HTTP Response ke Browser.

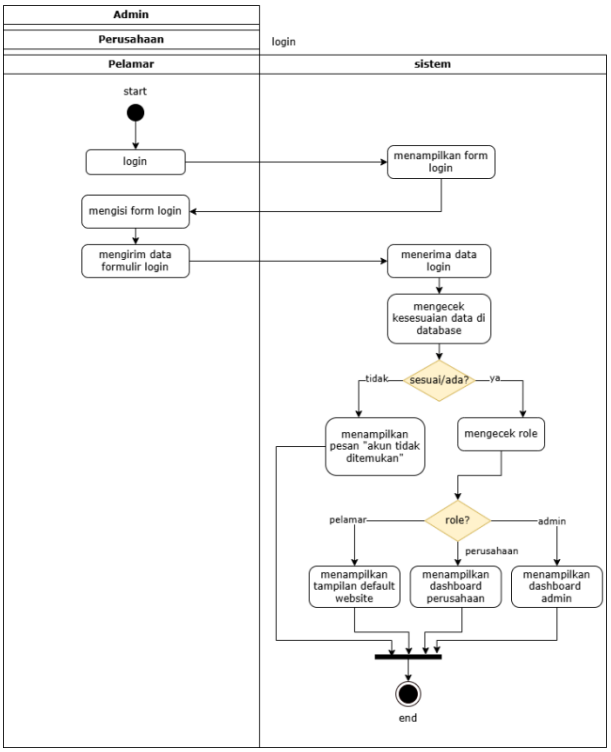
Alur ini memastikan setiap layer hanya menjalankan tanggung jawabnya masing-masing.

3.5 Activity Diagram

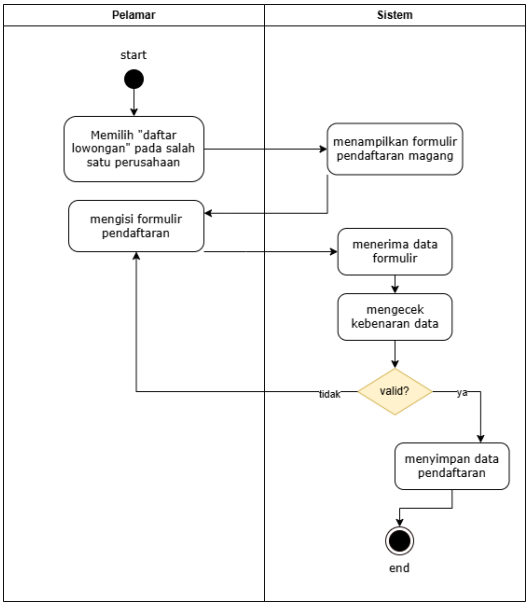
3.5.1 Register – Pelamar, Perusahaan



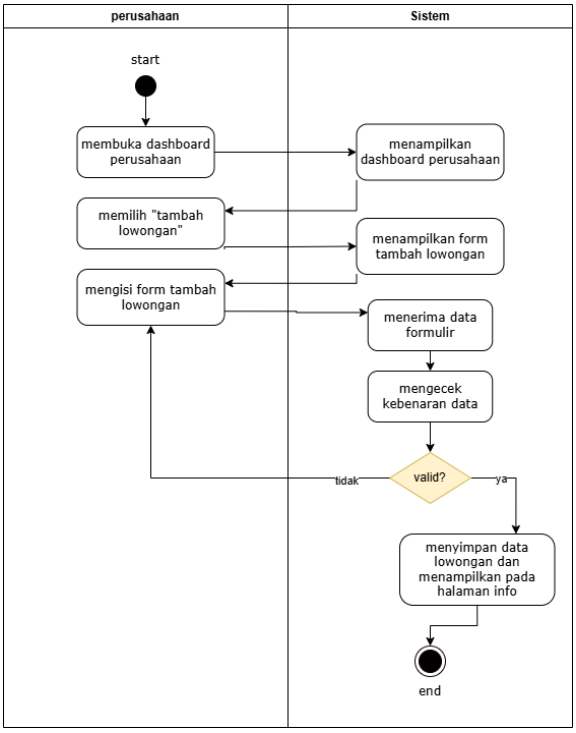
3.5.2 Login – Pelamar, Perusahaan, Admin



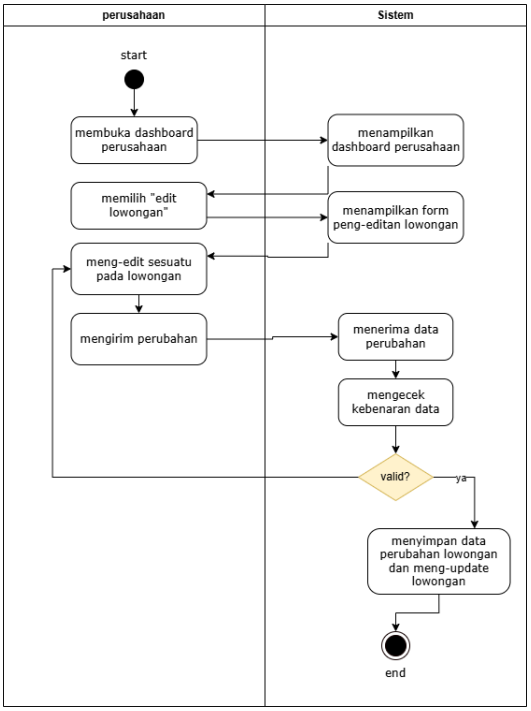
3.5.3 Melamar Lowongan Magang – Pelamar



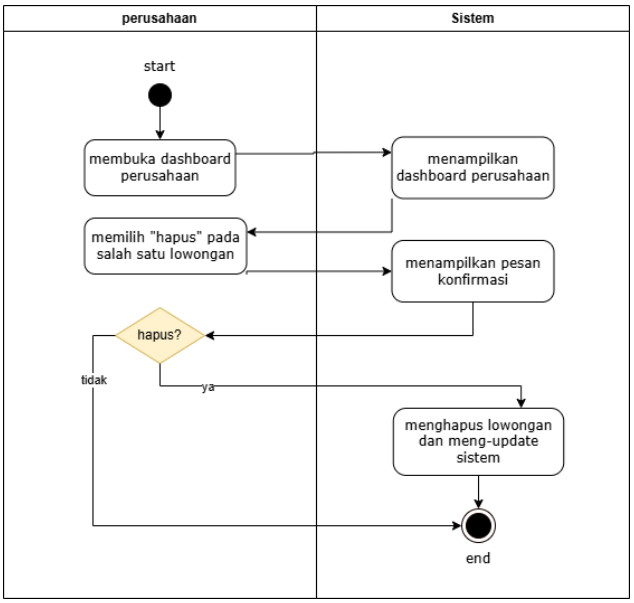
3.5.4 Menambah Lowongan Magang – Perusahaan



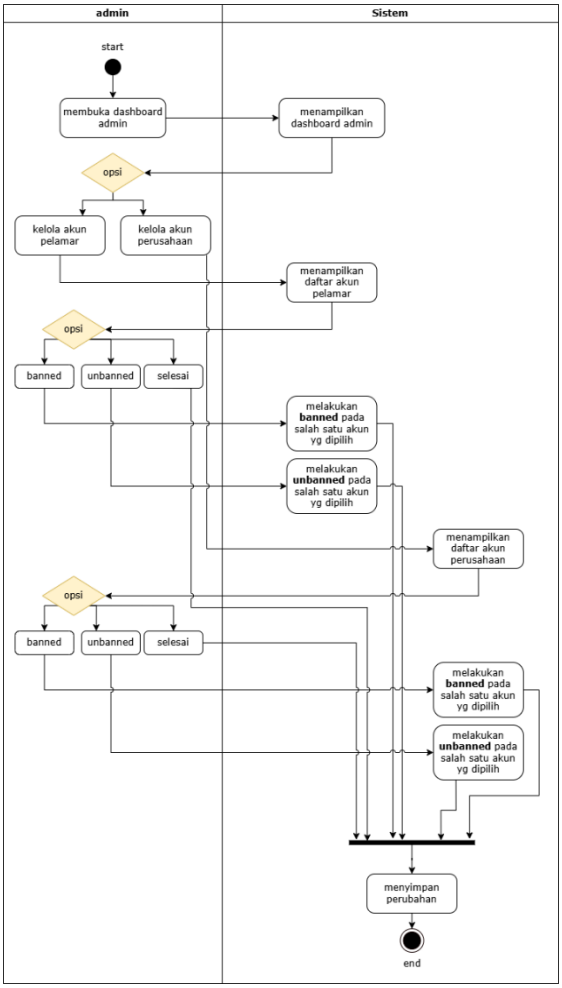
3.5.5 Mengedit Lowongan Magang – Perusahaan



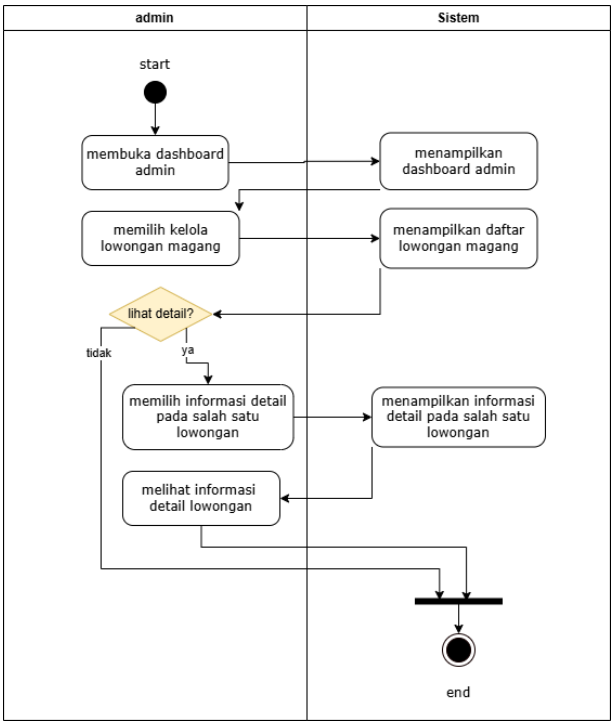
3.5.6 Menghapus Lowongan Magang – Perusahaan



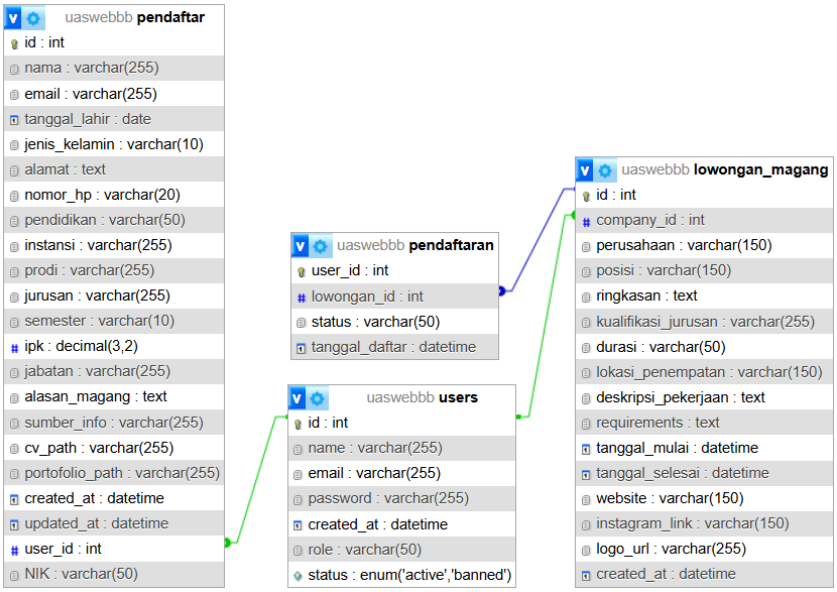
3.5.7 Mengelola Akun Pelamar dan Perusahaan – Admin



3.5.8 Melihat Daftar Lowongan Magang & Statusnya – Admin



3.6 Struktur Database



1. Analisis Masalah Utama

Diagram ini memodelkan alur pendaftaran magang, mulai dari:

- User (akun)
- Data detail pendaftar
- Lowongan magang
- Relasi pendaftaran antara user dan lowongan

Tujuan utamanya: memastikan satu user hanya bisa daftar satu lowongan, dan satu lowongan bisa menerima banyak pendaftar, dengan status yang bisa dilacak.

2. Entitas & Fungsi Tiap Tabel

A. users — Akun Sistem

Peran: Identitas login dan otorisasi

Atribut penting:

- id → primary key
- email, password → autentikasi
- role → membedakan (misalnya: admin, pelamar, perusahaan)
- status → active / banned

Makna desain:

- Tabel ini tidak menyimpan data detail pendaftaran.
- Fokus pada akses sistem, bukan profil lengkap

B. pendaftar — Profil Lengkap Pelamar

Peran: Menyimpan data personal & akademik

Isi utama:

- Identitas: nama, email, NIK, tanggal_lahir
- Akademik: pendidikan, prodi, jurusan, semester, ipk
- Kontak & dokumen: nomor_hp, cv_path, portofolio_path
- Motivasi: alasan_magang, sumber_info
- Relasi: user_id

Relasi penting:

- user_id → FK ke users.id

Makna desain:

- 1 user → 1 data pendaftar
- Memisahkan akun dan profil (best practice)

C. lowongan_magang — Informasi Lowongan

Peran: Menyimpan detail peluang magang

Atribut inti:

- Perusahaan: company_id, perusahaan, website, instagram_link
- Posisi: posisi, deskripsi_pekerjaan
- Kualifikasi: kualifikasi_jurusan, requirements
- Teknis: durasi, lokasi_penempatan
- Waktu: tanggal_mulai, tanggal_selesai
- Branding: logo_url

Makna desain:

- Satu lowongan berdiri sendiri
- Bisa dihubungkan ke banyak pelamar

D. pendaftaran — Tabel Relasi (Junction Table)

Peran krusial: Penghubung user ↔ lowongan

Atribut:

- user_id → FK ke users
- lowongan_id → FK ke lowongan_magang
- status → (misal: pending, diterima, ditolak)
- tanggal_daftar

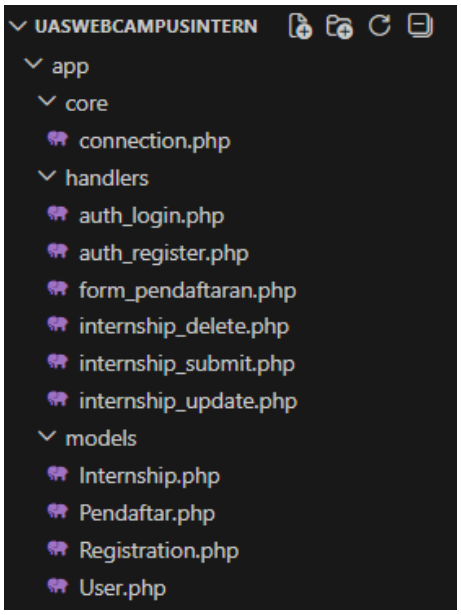
Makna desain:

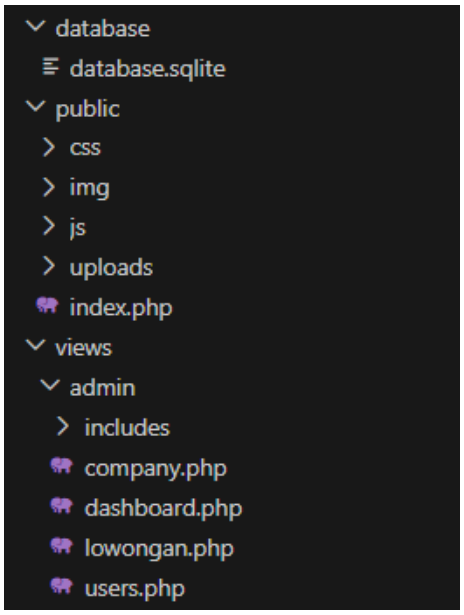
- Mewakili many-to-many relationship
- Menyimpan state proses seleksi

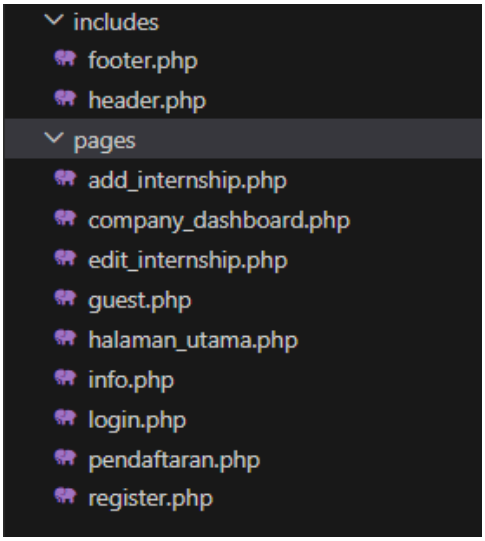
Tanpa tabel ini, sistem tidak bisa melacak pendaftaran dengan benar.

BAB IV
IMPLEMENTASI SISTEM DAN STRUKTUR PROGRAM

4.1 Struktur Program







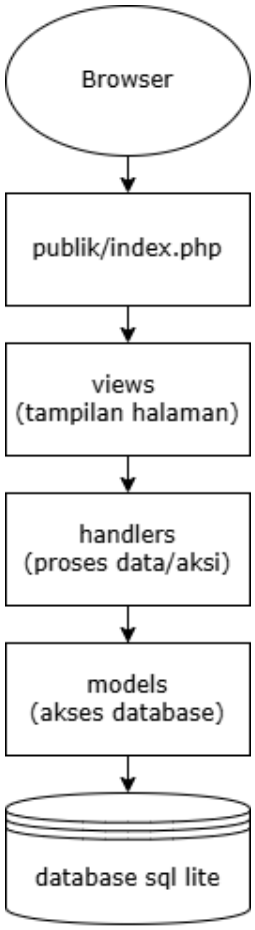
1. Analisis Masalah

Struktur Website campus intern ini adalah aplikasi web PHP native dengan pola semi-MVC:

- Model → *models/*
- Controller / Logic → *handlers/*
- View → *views/*
- Entry point → *public/index.php*

Tujuannya: memisahkan logika, data, dan tampilan.

2. Gambaran Besar Alur Sistem



Artinya:

- User tidak langsung mengakses file handler
- Semua data diproses terpusat dan terkontrol

3. Penjelasan Tiap Folder & File

- app/

Folder ini berisi logika backend utama, ini adalah isi logika aplikasi, tempat seluruh proses bisnis berjalan.

- core/

Fungsi: Konfigurasi dasar sistem, berisi konfigurasi dasar yang digunakan oleh seluruh aplikasi.

- connection.php
  → Mengatur koneksi ke database SQLite
  → Dipanggil oleh model atau handler

Arah dependensi: handlers & models → connection.php

- handlers/ - Controller atau logic layer

Berisi file pemroses aksi user (POST/GET).

Contoh fungsi:

- Validasi input
- Insert / update / delete database
- Redirect halaman

Isi penting:

- auth_login.php → proses login
- auth_register.php → proses registrasi user
- form_pendaftaran.php → simpan data pendaftar
- internship_submit.php → tambah lowongan
- internship_update.php → edit lowongan
- internship_delete.php → hapus lowongan

Arah alur: views (form) → handlers → models → database

- models/ - data access layer

berisi class representasi tabel database, merepresentasikan struktur tabel database dalam bentuk class PHP.

File	Representasi
User.php	Tabel user
Pendaftar.php	Tabel Pendaftar
Internship.php	Tabel Lowongan magang
Registration.php	Tabel Pendaftaran

Fungsi utama:

- Query database
- CRUD data
- Menjaga agar SQL tidak tercampur dengan HTML

- Database/
    - Database.sqllite

- Database utama aplikasi

Dipakai oleh connection.php

- Public/ - asset public

Folder ini boleh diakses langsung oleh browser**.**

- css/style.css → styling tampilan
- js/ → interaksi client-side
- img/ → gambar statis
- uploads/ → file CV & portofolio

- Public/index.php – entry point

Fungsi:

- Routing sederhana
- Menentukan halaman yang ditampilkan
- Include file view yang sesuai

Semua akses halaman diarahkan ke file ini.

- Views/ - tampilan (UI layer)

Berisi file HTML + PHP ringan.

- includes/

- header.php
- footer.php

Digunakan ulang di semua halaman (DRY principle).

- admin/

Halaman khusus admin:

- dashboard.php
- users.php
- lowongan.php
- company.php

Biasanya hanya bisa diakses jika role = admin.

- pages/

Halaman utama sistem:

- halaman_utama.php → landing page
- login.php, register.php
- pendaftaran.php → form daftar magang
- add_internship.php, edit_internship.php
- company_dashboard.php
- guest.php, info.php

Halaman ini:

- Menampilkan data dari models
- Mengirim form ke handlers

4.2 Penjelasan Source Code

4.2.1 Entry Point & Routing

Public/index.php

```
<?php
```

```php
/**
 * 1. KONFIGURASI SISTEM & KONEKSI DATABASE
 */
session_start();

// Definisi path absolut untuk memudahkan pemanggilan file dari berbagai level folder
define('ROOT_PATH', __DIR__ . '/../');

require_once ROOT_PATH . 'app/core/connection.php';
require_once ROOT_PATH . 'app/models/User.php';
require_once ROOT_PATH . 'app/models/Internship.php';


/**
 * 2. FUNGSI HELPER
 */
function url_for($page) {
    return 'index.php?page=' . urlencode($page);
}


/**
 * 3. MANAJEMEN SESI & OTENTIKASI
 */
$is_logged_in = isset($_SESSION['user_id']);
$user_name    = $_SESSION['user_name'] ?? 'Tamu';
$user_role    = $_SESSION['user_role'] ?? 'guest';


/**
 * 4. LOGIKA ROUTING & KONTROL AKSES
```

```php
 */
$page   = $_GET['page'] ?? 'halaman_utama';
$action = $_GET['action'] ?? null;

// Menangani permintaan logout
if ($action === 'logout') {
    session_unset();
    session_destroy();
    header('Location: ' . url_for('halaman_utama'));
    exit();
}

/**
 * Switch case ini berfungsi sebagai "Router" utama aplikasi
 */
switch ($page) {
    // Menampilkan halaman beranda
    case 'halaman_utama':
        $view_file = 'views/pages/halaman_utama.php';
        break;

    // Menampilkan daftar lowongan magang
    case 'info':
        $view_file = 'views/pages/info.php';
        break;

    // Menampilkan form login (Dicek: jika sudah login, lempar ke beranda)
    case 'login':
        if ($is_logged_in) { header('Location: ' . url_for('halaman_utama')); exit(); }
        $view_file = 'views/pages/login.php';
        break;
```

```
// Memproses data login yang dikirim via POST
case 'login_submit':
    require_once ROOT_PATH . 'app/handlers/auth_login.php';
    exit();

// Menampilkan form registrasi akun baru
case 'register':
    if ($is_logged_in) { header('Location: ' . url_for('halaman_utama')); exit(); }
    $view_file = 'views/pages/register.php';
    break;

// Memproses data registrasi akun baru
case 'register_submit':
    require_once ROOT_PATH . 'app/handlers/auth_register.php';
    exit();

// Panel utama untuk pengguna dengan role perusahaan
case 'company_dashboard':
    if ($user_role !== 'company') { header('Location: ' . url_for('halaman_utama')); exit(); }
    $view_file = 'views/pages/company_dashboard.php';
    break;

// Panel utama untuk administrator (statistik dan rangkuman)
case 'admin_dashboard':
    if ($user_role !== 'admin') { header('Location: ' . url_for('halaman_utama')); exit(); }
    $view_file = 'views/admin/dashboard.php';
    break;

// Manajemen data pengguna oleh admin
case 'admin_users':
```

```
    $view_file = 'views/admin/users.php';
    break;

// Manajemen data perusahaan oleh admin
case 'admin_company':
    $view_file = 'views/admin/company.php';
    break;

// Manajemen seluruh lowongan magang oleh admin
case 'admin_lowongan':
    $view_file = 'views/admin/lowongan.php';
    break;

// Form bagi perusahaan untuk membuat lowongan baru
case 'add_internship':
    if ($user_role !== 'company') { header('Location: ' . url_for('halaman_utama')); exit(); }
    $view_file = 'views/pages/add_internship.php';
    break;

// Memproses data lowongan baru (Insert ke DB)
case 'submit_internship':
    require_once ROOT_PATH . 'app/handlers/internship_submit.php';
    exit();

// Form bagi perusahaan untuk mengedit lowongan yang ada
case 'edit_internship':
    if ($user_role !== 'company') { header('Location: ' . url_for('halaman_utama')); exit(); }
    $view_file = 'views/pages/edit_internship.php';
    break;

// Memproses update data lowongan (Update ke DB)
```

```
    case 'internship_update':
        require_once ROOT_PATH . 'app/handlers/internship_update.php';
        exit();

    // Memproses penghapusan lowongan oleh perusahaan
    case 'delete_internship':
        if ($user_role !== 'company') { header('Location: ' . url_for('halaman_utama')); exit(); }
        require_once ROOT_PATH . 'app/handlers/internship_delete.php';
        exit();

    // Form pendaftaran magang bagi mahasiswa
    case 'pendaftaran':
        $view_file = 'views/pages/pendaftaran.php';
        break;

    // Memproses pengiriman lamaran magang (Insert ke tabel pendaftaran)
    case 'submit_pendaftaran':
        require_once ROOT_PATH . 'app/handlers/form_pendaftaran.php';
        exit();

    // Halaman jika parameter 'page' tidak ditemukan dalam daftar di atas
    default:
        http_response_code(404);
        $view_file = 'views/pages/404.php';
        break;
}


/**
 * 5. RENDERING VIEW (LAYOUT ENGINE)
 */
$full_path = ROOT_PATH . $view_file;
```

```
if (file_exists($full_path)) {
    // Memisahkan Header/Footer khusus Admin agar tampilan Dashboard Admin berbeda
    if ($user_role === 'admin' && str_starts_with($view_file, 'views/admin')) {
        require_once ROOT_PATH . 'views/admin/includes/header.php';
        require_once $full_path;
        require_once ROOT_PATH . 'views/admin/includes/footer.php';
    } else {
        require_once ROOT_PATH . 'views/includes/header.php';
        require_once $full_path;
        require_once ROOT_PATH . 'views/includes/footer.php';
    }
} else {
    http_response_code(404);
    echo "<h1>404 Halaman Tidak Ditemukan</h1>";
}
```

```
session_start();

// Definisi path absolut untuk memudahkan pemanggilan file dari berbagai level folder
define('ROOT_PATH', __DIR__ . '/../');
```

Kode ini digunakan untuk memulai session PHP serta menentukan path absolut proyek agar pemanggilan file dari berbagai folder menjadi konsisten dan tidak bergantung pada lokasi file saat ini.

```
require_once ROOT_PATH . 'app/core/connection.php';
require_once ROOT_PATH . 'app/models/User.php';
require_once ROOT_PATH . 'app/models/Internship.php';
```

Kode ini untuk memuat koneksi database dan class model utama agar dapat digunakan di seluruh alur aplikasi, khususnya pada proses autentikasi dan pengelolaan data lowongan magang.

Bagian ini berfungsi sebagai tahap awal (bootstrap) aplikasi:

- Mengaktifkan session untuk login & role
- Menentukan path absolut agar file dapat dipanggil dari mana pun
- Memuat koneksi database dan model utama

```
function url_for($page) {
    return 'index.php?page=' . urlencode($page);
}
```

Digunakan untuk membentuk URL halaman secara konsisten dan aman, serta mempermudah proses redirect tanpa menulis URL secara manual berulang kali.

Fungsi ini:

- Menyeragamkan pembuatan URL
- Menghindari penulisan URL manual berulang
- Membantu keamanan parameter URL

```
$is_logged_in = isset($_SESSION['user_id']);
$user_name    = $_SESSION['user_name'] ?? 'Tamu';
$user_role    = $_SESSION['user_role'] ?? 'guest';
```

Bagian ini:

- Menentukan apakah user sudah login
- Mengambil role user (admin / company / guest)
- Menjadi dasar kontrol akses halaman

```
$page   = $_GET['page'] ?? 'halaman_utama';
$action = $_GET['action'] ?? null;
```

Digunakan untuk menentukan halaman yang akan ditampilkan serta aksi khusus yang akan dijalankan berdasarkan parameter URL.

```
if ($action === 'logout') {
    session_unset();
    session_destroy();
    header('Location: ' . url_for('halaman_utama'));
    exit();
}
```

Kode ini berfungsi untuk menghapus seluruh data session pengguna dan mengembalikan pengguna ke halaman utama setelah logout.

```
switch ($page) {
    // Menampilkan halaman beranda
    case 'halaman_utama':
        $view_file = 'views/pages/halaman_utama.php';
        break;
```

Digunakan sebagai pengatur rute utama aplikasi yang menentukan file tampilan mana yang akan ditampilkan berdasarkan parameter page**.**

```
    case 'login_submit':
        require_once ROOT_PATH . 'app/handlers/auth_login.php';
        exit();
```

Kode ini menjalankan file handler untuk memproses data yang dikirim melalui form tanpa menampilkan halaman baru.

```
    if ($user_role === 'admin' && str_starts_with($view_file, 'views/admin')) {
```

Digunakan untuk memastikan hanya pengguna dengan peran tertentu yang dapat mengakses halaman tertentu dalam sistem.

```
    } else {
        require_once ROOT_PATH . 'views/includes/header.php';
```

```
        require_once $full_path;
        require_once ROOT_PATH . 'views/includes/footer.php';
    }
```

Kode ini bertanggung jawab menampilkan halaman web secara lengkap dengan menyatukan header, konten utama, dan footer.

```
} else {
    http_response_code(404);
    echo "<h1>404 Halaman Tidak Ditemukan</h1>";
}
```

Digunakan untuk menampilkan halaman kesalahan ketika halaman yang diminta tidak tersedia di dalam sistem.

4.2.2 Koneksi Database

Core/connection.php

```
<?php

/**
 * 1. KONFIGURASI KREDENSIAL DATABASE
 * Mengatur parameter koneksi ke database MySQL.
 */
$host    = '127.0.0.1';
$db      = 'uaswebbb';
$user    = 'root';
$pass    = 'ZARY666AN';
$charset = 'utf8mb4';

/**
 * 2. KONSTRUKSI DSN & OPSI PDO
 * DSN (Data Source Name) mendefinisikan driver, host, dan nama database.
```

```php
 */
$dsn = "mysql:host=$host;dbname=$db;charset=$charset";

$options = [
    // Mengaktifkan mode error Exception untuk penanganan error yang lebih baik
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,

    // Mengatur hasil query default sebagai array asosiatif
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,

    // Keamanan: Menonaktifkan emulasi prepared statements untuk mencegah SQL Injection
    PDO::ATTR_EMULATE_PREPARES   => false,
];

/**
 * 3. INISIALISASI KONEKSI (DATABASE CONNECTION)
 * Membuat instance PDO global yang akan digunakan di seluruh modul aplikasi.
 */
try {
    $pdo = new PDO($dsn, $user, $pass, $options);
} catch (\PDOException $e) {
    // Menghentikan skrip dan menampilkan pesan jika koneksi database gagal
    die("Koneksi database gagal: " . $e->getMessage());
}
/**
 * 1. KONFIGURASI KREDENSIAL DATABASE
 * Mengatur parameter koneksi ke database MySQL.
 */
$host    = '127.0.0.1';
$db      = 'uaswebbb';
$user    = 'root';
```

```
$pass    = 'ZARY666AN';
$charset = 'utf8mb4';
```

Kode ini digunakan untuk menyimpan parameter utama koneksi database, meliputi alamat server, nama database, username, password, dan karakter set yang digunakan oleh sistem.

```
 * 2. KONSTRUKSI DSN & OPSI PDO
 * DSN (Data Source Name) mendefinisikan driver, host, dan nama database.
 */
$dsn = "mysql:host=$host;dbname=$db;charset=$charset";
```

DSN berfungsi sebagai string identitas koneksi yang memberi tahu PDO jenis database yang digunakan, lokasi server, serta database yang akan diakses.

```
$options = [
    // Mengaktifkan mode error Exception untuk penanganan error yang lebih baik
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,

    // Mengatur hasil query default sebagai array asosiatif
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,

    // Keamanan: Menonaktifkan emulasi prepared statements untuk mencegah SQL Injection
    PDO::ATTR_EMULATE_PREPARES   => false,
];
```

Kode ini mengatur perilaku koneksi PDO, seperti cara penanganan error, format hasil query, serta pengamanan query menggunakan prepared statement asli untuk mencegah SQL Injection.

```
 * 3. INISIALISASI KONEKSI (DATABASE CONNECTION)
 * Membuat instance PDO global yang akan digunakan di seluruh modul aplikasi.
 */
try {
    $pdo = new PDO($dsn, $user, $pass, $options);
```

Kode ini membuat objek koneksi database menggunakan PDO yang nantinya digunakan oleh seluruh model dan handler dalam aplikasi.

```
} catch (\PDOException $e) {
    // Menghentikan skrip dan menampilkan pesan jika koneksi database gagal
    die("Koneksi database gagal: " . $e->getMessage());
}
```

Digunakan untuk menangani kegagalan koneksi database dengan menghentikan eksekusi program dan menampilkan pesan error agar kesalahan dapat segera diketahui.

4.2.3 Implementasi Model

- Models/User.php

```
<?php
/**
 * User.php
 * Model ini menangani semua interaksi database untuk tabel 'users'.
 * Model ini sudah dimodifikasi untuk menangani kolom 'role'.
 */

// Pastikan koneksi database ($pdo) sudah tersedia di scope global
global $pdo;

class User {

    /**
     * Mencari pengguna berdasarkan alamat email.
     * Digunakan selama proses login dan register (untuk cek duplikasi).
     * @param string $email
     * @return array|false Data pengguna (termasuk role) atau false jika tidak ditemukan.
     */
    public static function findByEmail($email) {
        global $pdo; // Akses koneksi database global
```

```
        // Kolom 'role' sudah disertakan dalam SELECT
        $sql = "SELECT id, name, email, password, role, status, created_at FROM users WHERE email = ?";
        $stmt = $pdo->prepare($sql);
        $stmt->execute([$email]);
        return $stmt->fetch();
    }

    /**
     * Membuat pengguna baru.
     * Digunakan selama proses registrasi.
     * @param array $data Array asosiatif berisi 'name', 'email', 'password', dan 'role'.
     * @return bool True jika berhasil, false jika gagal.
     */
    public static function create(array $data) {
        global $pdo;

        if (!isset($data['password'])) {
            return false;
        }

        // Pastikan role diset, default ke 'user'
        $role = $data['role'] ?? 'user';
        $hashed_password = password_hash($data['password'], PASSWORD_DEFAULT);

        // SQL diubah: Menambahkan kolom 'role'
        $sql = "INSERT INTO users (name, email, password, role, created_at) VALUES (?, ?, ?, ?, NOW())";
        $stmt = $pdo->prepare($sql);

        try {
```

```
            return $stmt->execute([
                $data['name'],
                $data['email'],
                $hashed_password,
                $role // <-- BINDING ROLE
            ]);
        } catch (\PDOException $e) {
            // Dalam kasus error database (misal: kolom role belum ada), ini akan mengembalikan false
            return false;
        }
    }

    /**
     * Memverifikasi kredensial pengguna.
     * @param string $email
     * @param string $password
     * @return array|false Data pengguna (termasuk role) jika kredensial valid, false jika gagal.
     */
    public static function verifyCredentials($email, $password) {
        // findByEmail sekarang akan mengembalikan data termasuk 'role'
        $user = self::findByEmail($email);

        if ($user) {
            // Membandingkan password yang di-hash dengan password yang dimasukkan
            if (password_verify($password, $user['password'])) {
                // Berhasil: Data user lengkap dikembalikan, siap disimpan ke sesi
                return $user;
            }
        }

        // Gagal
```

```
        return false;
    }
}
```

```
class User {
```

Kode ini mendefinisikan class User sebagai **model** yang merepresentasikan tabel users di database. Seluruh operasi terkait data pengguna dipusatkan dalam class ini.

```
global $pdo;
```

Digunakan untuk mengakses objek koneksi database (PDO) yang telah diinisialisasi pada file connection.php, sehingga model dapat menjalankan query tanpa membuat koneksi baru.

```
    public static function findByEmail($email) {
        global $pdo; // Akses koneksi database global

        // Kolom 'role' sudah disertakan dalam SELECT
        $sql = "SELECT id, name, email, password, role, status, created_at FROM users WHERE email = ?";
        $stmt = $pdo->prepare($sql);
        $stmt->execute([$email]);
        return $stmt->fetch();
    }
```

Kode ini digunakan untuk mengambil satu data pengguna berdasarkan email, yang dimanfaatkan pada proses login dan pengecekan data ganda saat registrasi.

```
        $stmt = $pdo->prepare($sql);
        $stmt->execute([$email]);
```

Digunakan untuk menjalankan query SQL secara aman dengan parameter binding guna mencegah serangan SQL Injection.

```
    public static function create(array $data) {
```

```
    $sql = "INSERT INTO users (name, email, password, role, created_at) VALUES (?, ?, ?, ?, NOW())";
```

Kode ini berfungsi untuk menyimpan data pengguna baru ke dalam tabel users saat proses registrasi akun.

```
    $role = $data['role'] ?? 'user';
```

Digunakan untuk menentukan peran pengguna, dengan nilai default user apabila role tidak dikirimkan dari form.

```
    $hashed_password = password_hash($data['password'], PASSWORD_DEFAULT);
```

Kode ini mengamankan password pengguna dengan metode hashing standar PHP sebelum disimpan ke database, sehingga password tidak tersimpan dalam bentuk teks asli.

```
    try {
        return $stmt->execute([
            $data['name'],
            $data['email'],
            $hashed_password,
            $role // <-- BINDING ROLE
        ]);
```

Digunakan untuk menangani kemungkinan kegagalan query database dan mencegah aplikasi berhenti secara tidak terkontrol.

```
public static function verifyCredentials($email, $password) {
    // findByEmail sekarang akan mengembalikan data termasuk 'role'
    $user = self::findByEmail($email);

    if ($user) {
        // Membandingkan password yang di-hash dengan password yang dimasukkan
        if (password_verify($password, $user['password'])) {
            // Berhasil: Data user lengkap dikembalikan, siap disimpan ke sesi
            return $user;
        }
```

```
    }
```

Kode ini digunakan untuk memverifikasi kecocokan email dan password pengguna saat login dengan membandingkan password input dengan hash yang tersimpan di database.

- models/Internship.php

```php
<?php
/**
 * Internship.php
 * Model untuk mengelola data lowongan magang (tabel lowongan_magang).
 */

class Internship {

    // Pastikan koneksi $pdo tersedia secara global (dari connection.php)
    private static function getPdo() {
        global $pdo;
        if (!isset($pdo)) {
            // Ini akan memastikan error ditangkap jika PDO belum terinisialisasi
            throw new Exception("Koneksi database (PDO) tidak tersedia.");
        }
        return $pdo;
    }

    /**
     * Mengambil semua lowongan magang, diurutkan berdasarkan tanggal terbaru.
     * @return array Array lowongan magang
     */

//  perusahaan banned → lowongan HILANG
// user tidak bisa lihat & daftar
```

```php
// admin tetap bisa manage
    public static function getAllPostings() {
        $pdo = self::getPdo();

        $sql = "
            SELECT
                lm.*
            FROM lowongan_magang lm
            JOIN users u ON lm.company_id = u.id
            WHERE u.role = 'company'
            AND u.status = 'active'
            ORDER BY lm.created_at DESC
        ";

        $stmt = $pdo->prepare($sql);
        $stmt->execute();
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    }


    // ADMIN ONLY
    public static function getAllPostingsForAdmin() {
        $pdo = self::getPdo();

        $sql = "
            SELECT
                lm.*,
                u.status AS company_status
            FROM lowongan_magang lm
            JOIN users u ON lm.company_id = u.id
            WHERE u.role = 'company'
            ORDER BY lm.created_at DESC
```

```
        ";

        $stmt = $pdo->prepare($sql);
        $stmt->execute();
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
    }


    /**
     * Mengambil satu lowongan berdasarkan ID.
     * @param int $id ID Lowongan
     * @return array|false Data lowongan atau false jika tidak ditemukan
     */
    public static function getPostingById($id) {
        $pdo = self::getPdo();
        $stmt = $pdo->prepare("SELECT * FROM lowongan_magang WHERE id = :id");
        $stmt->execute([':id' => $id]);
        return $stmt->fetch(PDO::FETCH_ASSOC);
    }

    /**
     * Mengambil lowongan magang berdasarkan ID Perusahaan.
     * @param int $companyId ID pengguna perusahaan yang sedang login
     * @return array Array lowongan magang milik perusahaan tersebut
     */
    public static function getPostingsByCompanyId($companyId) {
        $pdo = self::getPdo();
        $stmt = $pdo->prepare("SELECT * FROM lowongan_magang WHERE company_id =
:company_id ORDER BY created_at DESC");
        $stmt->execute([':company_id' => $companyId]);
        return $stmt->fetchAll(PDO::FETCH_ASSOC);
```

```
}

/**
 * Menyimpan lowongan magang baru.
 * @param array $data Data lowongan dari form
 * @return bool True jika berhasil, False jika gagal
 */
public static function createPosting($data) {
    $pdo = self::getPdo();
    $sql = "INSERT INTO lowongan_magang (
                company_id, perusahaan, posisi, ringkasan, kualifikasi_jurusan,
                durasi, lokasi_penempatan, deskripsi_pekerjaan, requirements,
                tanggal_mulai, tanggal_selesai, website, instagram_link, logo_url
            ) VALUES (
                :company_id, :perusahaan, :posisi, :ringkasan, :kualifikasi_jurusan,
                :durasi, :lokasi_penempatan, :deskripsi_pekerjaan, :requirements,
                :tanggal_mulai, :tanggal_selesai, :website, :instagram_link, :logo_url
            )";

    $stmt = $pdo->prepare($sql);

    return $stmt->execute([
        ':company_id'          => $data['company_id'],
        ':perusahaan'          => $data['perusahaan'],
        ':posisi'              => $data['posisi'],
        ':ringkasan'           => $data['ringkasan'],
        ':kualifikasi_jurusan' => $data['kualifikasi_jurusan'],
        ':durasi'              => $data['durasi'],
        ':lokasi_penempatan'   => $data['lokasi_penempatan'],
        ':deskripsi_pekerjaan' => $data['deskripsi_pekerjaan'],
        ':requirements'        => $data['requirements'],
```

```
        ':tanggal_mulai'       => $data['tanggal_mulai'],
        ':tanggal_selesai'     => $data['tanggal_selesai'],
        ':website'             => $data['website'],
        ':instagram_link'      => $data['instagram_link'],
        ':logo_url'            => $data['logo_url'],
    ]);
}

/**
 * Mengubah data lowongan magang yang sudah ada.
 * @param int $id ID Lowongan yang akan diubah
 * @param array $data Data lowongan baru dari form
 * @return bool True jika berhasil, False jika gagal
 */
public static function updatePosting($id, $data) {
    $pdo = self::getPdo();

    // Perusahaan dan company_id tidak perlu di-update kecuali jika ada migrasi akun
    $sql = "UPDATE lowongan_magang SET
                posisi = :posisi,
                ringkasan = :ringkasan,
                kualifikasi_jurusan = :kualifikasi_jurusan,
                durasi = :durasi,
                lokasi_penempatan = :lokasi_penempatan,
                deskripsi_pekerjaan = :deskripsi_pekerjaan,
                requirements = :requirements,
                tanggal_mulai = :tanggal_mulai,
                tanggal_selesai = :tanggal_selesai,
                website = :website,
                instagram_link = :instagram_link,
                logo_url = :logo_url
```

```
            WHERE id = :id";

    $stmt = $pdo->prepare($sql);

    return $stmt->execute([
        ':id'                 => $id,
        ':posisi'             => $data['posisi'],
        ':ringkasan'          => $data['ringkasan'],
        ':kualifikasi_jurusan' => $data['kualifikasi_jurusan'],
        ':durasi'             => $data['durasi'],
        ':lokasi_penempatan'  => $data['lokasi_penempatan'],
        ':deskripsi_pekerjaan' => $data['deskripsi_pekerjaan'],
        ':requirements'       => $data['requirements'],
        ':tanggal_mulai'      => $data['tanggal_mulai'],
        ':tanggal_selesai'    => $data['tanggal_selesai'],
        ':website'            => $data['website'] ?? null,
        ':instagram_link'     => $data['instagram_link'] ?? null,
        ':logo_url'           => $data['logo_url'] ?? 'default/logo.png', // Pastikan logo_url di handle
di handler
    ]);
  }

  /**
   * Menghapus lowongan berdasarkan ID.
   * @param int $id ID Lowongan
   * @return bool True jika berhasil, False jika gagal
   */
  public static function deletePosting($id) {
    $pdo = self::getPdo();
    $stmt = $pdo->prepare("DELETE FROM lowongan_magang WHERE id = :id");
    return $stmt->execute([':id' => $id]);
```

```
    }
}
class Internship {
```

Mendefinisikan class Internship sebagai model yang bertanggung jawab mengelola seluruh operasi database pada tabel lowongan_magang.

```
    private static function getPdo() {
        global $pdo;
        if (!isset($pdo)) {
            // Ini akan memastikan error ditangkap jika PDO belum terinisialisasi
            throw new Exception("Koneksi database (PDO) tidak tersedia.");
        }
        return $pdo;
    }
```

Menjamin bahwa koneksi database tersedia sebelum query dijalankan serta mencegah error tersembunyi akibat PDO yang belum terinisialisasi.

```
    public static function getAllPostings() {
        $pdo = self::getPdo();

        $sql = "
            SELECT
                lm.*
            FROM lowongan_magang lm
            JOIN users u ON lm.company_id = u.id
            WHERE u.role = 'company'
            AND u.status = 'active'
            ORDER BY lm.created_at DESC
        ";

        $stmt = $pdo->prepare($sql);
        $stmt->execute();
```

```
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
  }
```

Mengambil seluruh lowongan magang dari perusahaan yang masih aktif, sehingga lowongan dari perusahaan yang diblokir tidak ditampilkan ke pengguna umum.

```
  public static function getAllPostingsForAdmin() {
    $pdo = self::getPdo();
```

Memberikan akses penuh kepada admin untuk melihat semua lowongan, termasuk milik perusahaan yang berstatus tidak aktif.

```
  public static function getPostingById($id) {
    $pdo = self::getPdo();
    $stmt = $pdo->prepare("SELECT * FROM lowongan_magang WHERE id = :id");
```

Digunakan untuk menampilkan detail satu lowongan magang, biasanya pada halaman detail atau form edit.

```
  public static function getPostingsByCompanyId($companyId) {
      $stmt = $pdo->prepare("SELECT * FROM lowongan_magang WHERE company_id =
:company_id ORDER BY created_at DESC");
```

Mengambil seluruh lowongan yang dimiliki oleh satu akun perusahaan yang sedang login.

```
  public static function createPosting($data) {
    $pdo = self::getPdo();
    $sql = "INSERT INTO lowongan_magang (
```

Menyimpan data lowongan magang baru yang dikirim melalui form perusahaan ke dalam database.

```
    return $stmt->execute([
      ':company_id'        => $data['company_id'],
      ':perusahaan'        => $data['perusahaan'],
```

Mengikat data input ke query SQL secara aman untuk mencegah SQL Injection.

```
  public static function updatePosting($id, $data) {
    $pdo = self::getPdo();
```

```
    // Perusahaan dan company_id tidak perlu di-update kecuali jika ada migrasi akun
    $sql = "UPDATE lowongan_magang SET
```

Mengubah data lowongan magang yang sudah ada tanpa mengubah kepemilikan perusahaan.

```
        ':website'          => $data['website'] ?? null,
        ':instagram_link'   => $data['instagram_link'] ?? null,
        ':logo_url'         => $data['logo_url'] ?? 'default/logo.png',
```

Menangani data opsional agar aplikasi tetap stabil meskipun beberapa field tidak diisi.

```
  public static function deletePosting($id) {
    $pdo = self::getPdo();
    $stmt = $pdo->prepare("DELETE FROM lowongan_magang WHERE id = :id");
```

Menghapus data lowongan magang berdasarkan ID yang dipilih oleh perusahaan.

- Models/Registration.php

```
<?php
/**
 * Registration.php
 * Model untuk mengelola data Log Aplikasi / Pendaftaran Magang (tabel 'pendaftaran').
 */

class Registration {

  private static function getPdo() {
    global $pdo;
    if (!isset($pdo)) {
      throw new Exception("Koneksi database (PDO) tidak tersedia.");
    }
    return $pdo;
  }
```

```
    /**
     * Memeriksa apakah pengguna sudah terdaftar di lowongan tertentu.
     */
    public static function hasRegistered($userId, $lowonganId) {
        $pdo = self::getPdo();
        $stmt = $pdo->prepare("SELECT COUNT(*) FROM pendaftaran WHERE user_id = :user_id AND lowongan_id = :lowongan_id");
        $stmt->execute([
            ':user_id' => $userId,
            ':lowongan_id' => $lowonganId
        ]);
        return $stmt->fetchColumn() > 0;
    }

    /**
     * Mencatat log pendaftaran baru ke tabel 'pendaftaran'.
     * @param int $userId ID pengguna yang mendaftar
     * @param int $lowonganId ID lowongan yang didaftar
     * @return bool True jika berhasil, False jika gagal atau sudah terdaftar.
     */
    public static function createRegistration($userId, $lowonganId) {
        $pdo = self::getPdo();

        // Cek duplikasi sebelum INSERT
        if (self::hasRegistered($userId, $lowonganId)) {
            return false; // User sudah pernah mendaftar lowongan ini
        }

        // Asumsi: Tabel Anda bernama 'pendaftaran' dan memiliki kolom user_id, lowongan_id, status, tanggal_daftar
```

```
        $sql = "INSERT INTO pendaftaran (user_id, lowongan_id, status, tanggal_daftar)
                VALUES (:user_id, :lowongan_id, 'Pending', NOW())";

        $stmt = $pdo->prepare($sql);

        try {
            return $stmt->execute([
                ':user_id' => $userId,
                ':lowongan_id' => $lowonganId
            ]);
        } catch (\PDOException $e) {
            // Log error atau kembalikan false
            return false;
        }
    }
}
```

```
class Registration {
```

Mendefinisikan model Registration yang merepresentasikan aktivitas pendaftaran magang (relasi antara user dan lowongan).

```
    private static function getPdo() {
        global $pdo;
        if (!isset($pdo)) {
            throw new Exception("Koneksi database (PDO) tidak tersedia.");
        }
        return $pdo;
    }
```

Menjamin seluruh operasi database berjalan dengan koneksi yang valid dan mencegah eksekusi query tanpa PDO aktif.

```
    public static function hasRegistered($userId, $lowonganId) {
```

```
        $pdo = self::getPdo();
        $stmt = $pdo->prepare("SELECT COUNT(*) FROM pendaftaran WHERE user_id = :user_id
AND lowongan_id = :lowongan_id");
```

Memastikan satu pengguna tidak dapat mendaftar ke lowongan yang sama lebih dari satu kali.

```
        $stmt->execute([
            ':user_id' => $userId,
            ':lowongan_id' => $lowonganId
```

Mengikat nilai input secara aman ke query SQL untuk mencegah SQL Injection.

```
        if (self::hasRegistered($userId, $lowonganId)) {
            return false; // User sudah pernah mendaftar lowongan ini
        }
```

Menerapkan aturan bisnis bahwa satu user hanya boleh memiliki satu pendaftaran per lowongan.

```
        $sql = "INSERT INTO pendaftaran (user_id, lowongan_id, status, tanggal_daftar)
                VALUES (:user_id, :lowongan_id, 'Pending', NOW())";
```

Mencatat pendaftaran baru ke sistem dengan status awal *Pending* sebagai bagian dari proses seleksi.

```
        try {
            return $stmt->execute([
                ':user_id' => $userId,
                ':lowongan_id' => $lowonganId
            ]);
```

Menjaga stabilitas aplikasi dengan mencegah crash ketika terjadi kegagalan query database.

4.2.4 Proses Autentikasi & Proses Pendaftaran Magang

- handlers/auth_login.php

```
<?php

/**
```

```php
 * auth_login.php
 * Handler utama untuk memproses autentikasi pengguna.
 * File ini dipanggil melalui Front Controller (index.php) saat formulir login disubmit.
 */

// Proteksi akses: Memastikan skrip hanya diproses melalui metode POST
if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    header('Location: ' . url_for('login'));
    exit();
}

/**
 * 1. LOAD DEPENDENSI
 * Menggunakan model User untuk melakukan verifikasi data ke database.
 */
require_once ROOT_PATH . 'app/models/User.php';

/**
 * 2. AMBIL DAN SANITASI INPUT
 * Membersihkan data input untuk mencegah celah keamanan dasar.
 */
$email    = filter_var(trim($_POST['email'] ?? ''), FILTER_SANITIZE_EMAIL);
$password = $_POST['password'] ?? '';
$remember = isset($_POST['remember']); // Pengecekan opsi "Ingat Saya"

/**
 * 3. VALIDASI FORM & VERIFIKASI KREDENSIAL
 */
if (empty($email) || empty($password)) {
    $_SESSION['auth_error'] = "Email dan password wajib diisi.";
    header('Location: ' . url_for('login'));
```

```
    exit();
}

// Memanggil method di model User untuk mencocokkan email dan password hash
$user = User::verifyCredentials($email, $password);

if ($user) {

    /**
     * 4. VALIDASI STATUS AKUN (BANNED CHECK)
     * Memastikan pengguna yang valid secara kredensial tidak sedang dalam masa blokir.
     */
    if ($user['status'] === 'banned') {
        $_SESSION['auth_error'] = "Akun Anda telah dibanned. Silakan hubungi admin.";
        $_SESSION['old_input'] = ['email' => $email];
        header('Location: ' . url_for('login'));
        exit();
    }

    /**
     * 5. LOGIN BERHASIL: INISIALISASI SESI
     * Menyimpan informasi identitas dan peran (role) pengguna ke dalam session global.
     */
    $_SESSION['user_id']   = $user['id'];
    $_SESSION['user_name'] = $user['name'];
    $_SESSION['user_role'] = $user['role'];

    // (Opsional) Implementasi cookie untuk session jangka panjang
    if ($remember) {
        // Logika Persistent Login/Remember Me
    }
```

```php
    /**
     * 6. PENGALIHAN HALAMAN BERDASARKAN PERAN (ROLE-BASED REDIRECT)
     * Mengarahkan pengguna ke dashboard yang sesuai dengan hak aksesnya.
     */
    switch ($user['role']) {
        case 'admin':
            header('Location: ' . url_for('admin_dashboard'));
            break;

        case 'company':
            header('Location: ' . url_for('halaman_utama'));
            break;

        case 'user':
        default:
            header('Location: ' . url_for('halaman_utama'));
            break;
    }

    exit();

} else {

    /**
     * 7. LOGIN GAGAL
     * Mengembalikan pengguna ke halaman login dengan pesan kesalahan.
     */
    $_SESSION['auth_error'] = "Kredensial tidak valid. Silakan coba lagi.";

    // Mempertahankan input email agar pengguna tidak perlu mengetik ulang
```

```
    $_SESSION['old_input'] = ['email' => $email];

    header('Location: ' . url_for('login'));
    exit();
}
```

```
// Proteksi akses: Memastikan skrip hanya diproses melalui metode POST
if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    header('Location: ' . url_for('login'));
    exit();
}
```

Mencegah file handler diakses langsung melalui URL dan memastikan proses login hanya dijalankan melalui submit form.

```
require_once ROOT_PATH . 'app/models/User.php';
```

Menghubungkan handler dengan model User agar proses autentikasi dapat berinteraksi langsung dengan database.

```
$email    = filter_var(trim($_POST['email'] ?? ''), FILTER_SANITIZE_EMAIL);
$password = $_POST['password'] ?? '';
```

Membersihkan dan menormalisasi data input pengguna sebelum diproses untuk mengurangi risiko input berbahaya.

```
if (empty($email) || empty($password)) {
    $_SESSION['auth_error'] = "Email dan password wajib diisi.";
    header('Location: ' . url_for('login'));
    exit();
}
```

Menjamin bahwa data penting tersedia sebelum proses autentikasi dilanjutkan.

```
$user = User::verifyCredentials($email, $password);
```

Mencocokkan email dan password pengguna dengan data yang tersimpan di database melalui model.

```
if ($user) {
    if ($user['status'] === 'banned') {
```

Mencegah pengguna yang diblokir tetap dapat mengakses sistem meskipun kredensialnya valid.

```
    $_SESSION['user_id']   = $user['id'];
    $_SESSION['user_name'] = $user['name'];
    $_SESSION['user_role'] = $user['role'];
```

Menyimpan identitas dan peran pengguna sebagai state global selama sesi login berlangsung.

```
    switch ($user['role']) {
        case 'admin':
            header('Location: ' . url_for('admin_dashboard'));
            break;

        case 'company':
            header('Location: ' . url_for('halaman_utama'));
            break;

        case 'user':
        default:
            header('Location: ' . url_for('halaman_utama'));
            break;
    }
```

Mengatur hak akses dan halaman tujuan berdasarkan peran pengguna dalam sistem.

```
    $_SESSION['auth_error'] = "Kredensial tidak valid. Silakan coba lagi.";

    // Mempertahankan input email agar pengguna tidak perlu mengetik ulang
    $_SESSION['old_input'] = ['email' => $email];
```

Memberikan umpan balik kepada pengguna dan menjaga pengalaman pengguna dengan mempertahankan input sebelumnya.

- handlers/auth_register.php

```php
<?php

/**
 * auth_register.php
 * Handler utama untuk memproses registrasi pengguna baru.
 * File ini menangani validasi input, pembuatan akun, dan inisialisasi profil awal.
 */

// Proteksi akses: Memastikan skrip hanya diproses melalui metode POST
if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    header('Location: ' . url_for('register'));
    exit();
}

/**
 * 1. LOAD DEPENDENSI
 * Memuat Model User untuk interaksi data akun dan autentikasi.
 */
require_once ROOT_PATH . 'app/models/User.php';

/**
 * 2. AMBIL DAN SANITASI INPUT
 * Membersihkan data input untuk keamanan dan konsistensi data.
 */
$name                 = trim($_POST['name'] ?? '');
$email                = filter_var(trim($_POST['email'] ?? ''), FILTER_SANITIZE_EMAIL);
$password             = $_POST['password'] ?? '';
$password_confirmation = $_POST['password_confirmation'] ?? '';
$role                 = $_POST['role'] ?? 'user'; // Menangani pemilihan peran (User/Company)
```

```php
$errors = [];

/**
 * 3. VALIDASI INPUT FORM
 * Melakukan pengecekan kepatuhan data terhadap aturan bisnis sistem.
 */
if (empty($name) || empty($email) || empty($password) || empty($password_confirmation)) {
    $errors[] = "Semua bidang wajib diisi.";
}

if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    $errors[] = "Format email tidak valid.";
}

if (strlen($password) < 8) {
    $errors[] = "Password minimal 8 karakter.";
}

if ($password !== $password_confirmation) {
    $errors[] = "Konfirmasi password tidak cocok.";
}

// Validasi Role: Mencegah manipulasi nilai peran dari sisi klien
if (!in_array($role, ['user', 'company'])) {
    $errors[] = "Pilihan peran tidak valid.";
}

/**
 * 4. VALIDASI DUPLIKASI DAN PROSES PENYIMPANAN
 */
if (empty($errors)) {
```

```
    // Memastikan email belum digunakan oleh pengguna lain
    if (User::findByEmail($email)) {
        $errors[] = "Email ini sudah terdaftar. Silakan login.";
    }
}

if (empty($errors)) {

    // Menyiapkan dataset untuk dikirim ke layer Model
    $data = [
        'name'     => $name,
        'email'    => $email,
        'password' => $password,
        'role'     => $role
    ];

    // Eksekusi pembuatan akun melalui Model User
    $success = User::create($data);

    if ($success) {

        /**
         * 5. REGISTRASI BERHASIL: INISIALISASI SESI & PROFIL
         * Melakukan login otomatis setelah akun berhasil dibuat.
         */
        $user = User::findByEmail($email);

        if ($user) {

            // Logika spesifik untuk peran 'user' (Mahasiswa/Pendaftar)
            if ($user['role'] === 'user') {
```

```
            require_once ROOT_PATH . 'app/models/Pendaftar.php';

            // Membuat entri dasar pada tabel pendaftar untuk melengkapi profil nantinya
            $initial_profile_data = [
                'user_id' => $user['id'],
                'nama'    => $user['name'],
                'email'   => $user['email']
            ];

            $profile_created = Pendaftar::createInitialProfile($initial_profile_data);

            if (!$profile_created) {
                // Penanganan kegagalan pembuatan profil tanpa menghentikan proses login
              $_SESSION['profile_error'] = "Peringatan: Gagal membuat entri profil awal. Lengkapi
data di Dashboard Anda.";
            }
        }

        // Inisialisasi data session pengguna
        $_SESSION['user_id']        = $user['id'];
        $_SESSION['user_name']      = $user['name'];
        $_SESSION['user_role']      = $user['role'];
        $_SESSION['success_message'] = "Pendaftaran berhasil! Anda sudah login.";

        header('Location: ' . url_for('halaman_utama'));
        exit();
    }

} else {
    $errors[] = "Gagal menyimpan pengguna ke database. Coba lagi.";
}
```

```
}

/**
 * 6. PENANGANAN ERROR
 * Mengembalikan user ke form registrasi jika terjadi kesalahan validasi.
 */
if (!empty($errors)) {
    $_SESSION['auth_error'] = implode('<br>', $errors);

    // Mekanisme "Sticky Form" agar input sebelumnya tidak hilang
    $_SESSION['old_input'] = [
        'name'  => $name,
        'email' => $email,
        'role'  => $role
    ];

    header('Location: ' . url_for('register'));
    exit();
}
```

```
if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    header('Location: ' . url_for('register'));
    exit();
}
```

Menjamin proses registrasi hanya dapat dijalankan melalui submit form, bukan akses langsung lewat URL.

```
require_once ROOT_PATH . 'app/models/User.php';
```

Menghubungkan handler dengan layer Model agar proses registrasi dapat berinteraksi dengan database pengguna.

```
$name                 = trim($_POST['name'] ?? '');
$email                = filter_var(trim($_POST['email'] ?? ''), FILTER_SANITIZE_EMAIL);
$password             = $_POST['password'] ?? '';
$password_confirmation = $_POST['password_confirmation'] ?? '';
$role                 = $_POST['role'] ?? 'user';
```

Menormalisasi data input dan membersihkan karakter berbahaya sebelum masuk ke logika bisnis.

```
if (strlen($password) < 8) {
    $errors[] = "Password minimal 8 karakter.";
}
```

Menjaga konsistensi data dan memastikan input memenuhi kebijakan sistem (keamanan & kualitas data).

```
if (!in_array($role, ['user', 'company'])) {
    $errors[] = "Pilihan peran tidak valid.";
}
```

Mencegah manipulasi role dari sisi klien (misalnya user mencoba mendaftar sebagai admin).

```
    if (User::findByEmail($email)) {
        $errors[] = "Email ini sudah terdaftar. Silakan login.";
    }
```

Menjamin bahwa satu email hanya digunakan oleh satu akun pengguna.

```
    $data = [
        'name'     => $name,
        'email'    => $email,
        'password' => $password,
        'role'     => $role
    ];
```

Mengemas data input agar terstruktur sebelum dikirim ke layer Model.

```
    $success = User::create($data);
```

Menyimpan akun baru ke database melalui Model, termasuk hashing password.

```
        if ($user) {
```

```
        if ($user['role'] === 'user') {
```

Membuat entri profil awal otomatis untuk pengguna tertentu tanpa mengganggu alur registrasi utama.

```
            $_SESSION['user_id']        = $user['id'];
            $_SESSION['user_name']      = $user['name'];
```

Mengaktifkan status login otomatis setelah registrasi berhasil.

```
            header('Location: ' . url_for('halaman_utama'));
```

Mengalihkan pengguna ke halaman utama sebagai tanda registrasi sukses.

```
    $_SESSION['auth_error'] = implode('<br>', $errors);

    // Mekanisme "Sticky Form" agar input sebelumnya tidak hilang
    $_SESSION['old_input'] = [
        'name'  => $name,
        'email' => $email,
        'role'  => $role
    ];
```

Menampilkan pesan kesalahan dan mempertahankan input lama agar UX tetap baik.

- handlers/form_pendaftaran.php

```
<?php

/**
 * form_pendaftaran.php
 * Controller/Handler untuk memproses data formulir pendaftaran magang.
 * Memiliki fungsi ganda: Sinkronisasi profil pendaftar dan pencatatan log aplikasi.
 */

// Validasi metode akses (hanya mengizinkan metode POST)
if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
```

```
    header('Location: ' . url_for('pendaftaran'));
    exit();
}

/**
 * 0. KONTROL AKSES & DEKLARASI DEPENDENSI
 */
global $pdo, $is_logged_in, $user_role;

// Verifikasi otentikasi dan hak akses role 'user'
if (!$is_logged_in || $user_role !== 'user') {
    $_SESSION['auth_error'] = "Anda harus login sebagai Pendaftar.";
    header('Location: ' . url_for('login'));
    exit();
}

require_once ROOT_PATH . 'app/models/Pendaftar.php';
require_once ROOT_PATH . 'app/models/Registration.php';

$user_id     = $_SESSION['user_id'];
$lowongan_id = filter_var($_POST['lowongan_id'] ?? null, FILTER_VALIDATE_INT);
$errors      = [];

/**
 * 1. SANITASI DATA & VALIDASI INPUT TEKS
 */

// Sanitasi Data Diri
$nama          = trim($_POST['nama'] ?? '');
$nik           = trim($_POST['NIK'] ?? '');
$jenis_kelamin = $_POST['jenis_kelamin'] ?? '';
```

```php
$tanggal_lahir = $_POST['tanggal_lahir'] ?? '';
$alamat       = trim($_POST['alamat'] ?? '');
$nomor_hp     = trim($_POST['nomor_hp'] ?? '');
$email        = filter_var(trim($_POST['email'] ?? ''), FILTER_SANITIZE_EMAIL);

// Sanitasi Data Pendidikan
$pendidikan = $_POST['pendidikan'] ?? '';
$instansi   = trim($_POST['instansi'] ?? '');
$prodi      = trim($_POST['prodi'] ?? '');
$jurusan    = trim($_POST['jurusan'] ?? '');
$semester   = trim($_POST['semester'] ?? '');
$ipk        = trim($_POST['ipk'] ?? null);

// Sanitasi Informasi Magang
$jabatan       = trim($_POST['jabatan'] ?? '');
$alasan_magang = trim($_POST['alasan'] ?? '');
$sumber_info   = $_POST['sumber'] ?? '';

// Validasi Aturan Bisnis (Mandatory Fields)
if (empty($nama) || empty($email) || empty($jabatan) || empty($nik)) {
    $errors[] = "Semua bidang wajib harus diisi (termasuk NIK).";
}
if (!preg_match('/^\d{16}$/', $nik)) {
    $errors[] = "NIK harus terdiri dari 16 digit angka.";
}
if (!$lowongan_id) {
    $errors[] = "ID Lowongan tidak ditemukan.";
}
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    $errors[] = "Format email tidak valid.";
}
```

```php
/**
 * 2. MANAJEMEN UNGGAH BERKAS (CV & PORTOFOLIO)
 */
$cv_path       = null;
$portfolio_path = null;
$upload_dir    = ROOT_PATH . 'public/uploads/';

if (empty($errors)) {
    // Memastikan direktori unggahan tersedia
    if (!is_dir($upload_dir)) {
        mkdir($upload_dir, 0777, true);
    }

    // Pemrosesan unggah berkas CV (Wajib)
    if (isset($_FILES['cv']) && $_FILES['cv']['error'] === UPLOAD_ERR_OK) {
        $cv_ext  = pathinfo($_FILES['cv']['name'], PATHINFO_EXTENSION);
        $cv_name = "CV_" . $user_id . "_" . time() . "." . $cv_ext;
        if (move_uploaded_file($_FILES['cv']['tmp_name'], $upload_dir . $cv_name)) {
            $cv_path = 'uploads/' . $cv_name;
        } else {
            $errors[] = "Gagal memindahkan file CV.";
        }
    } else {
        $errors[] = "File CV wajib diunggah.";
    }

    // Pemrosesan unggah berkas Portofolio (Opsional)
    if (isset($_FILES['portofolio']) && $_FILES['portofolio']['error'] === UPLOAD_ERR_OK) {
        $port_ext  = pathinfo($_FILES['portofolio']['name'], PATHINFO_EXTENSION);
        $port_name = "PORT_" . $user_id . "_" . time() . "." . $port_ext;
```

```
        if (move_uploaded_file($_FILES['portofolio']['tmp_name'], $upload_dir . $port_name)) {
            $portfolio_path = 'uploads/' . $port_name;
        }
    }
}

/**
 * 3. PERSISTENSI DATA (DATABASE TRANSACTION)
 */
if (empty($errors)) {
    // Mapping data ke array asosiatif untuk layer Model
    $pendaftar_data = [
        'user_id'        => $user_id,
        'NIK'            => $nik,
        'nama'           => $nama,
        'jenis_kelamin'  => $jenis_kelamin,
        'tanggal_lahir'  => $tanggal_lahir,
        'alamat'         => $alamat,
        'nomor_hp'       => $nomor_hp,
        'email'          => $email,
        'pendidikan'     => $pendidikan,
        'instansi'       => $instansi,
        'prodi'          => $prodi,
        'jurusan'        => $jurusan,
        'semester'       => $semester,
        'ipk'            => $ipk,
        'jabatan'        => $jabatan,
        'alasan_magang'  => $alasan_magang,
        'sumber_info'    => $sumber_info,
        'cv_path'        => $cv_path,
        'portofolio_path' => $portfolio_path
```

```
    ];

    try {
        /**
         * Menggunakan Database Transaction untuk menjamin integritas data (Atomicity).
         * Jika salah satu proses gagal, seluruh perubahan akan dibatalkan (Rollback).
         */
        $pdo->beginTransaction();

        // Operasi 1: Sinkronisasi profil lengkap pendaftar
        $profile_success = Pendaftar::updateFullProfile($pendaftar_data);

        // Operasi 2: Pencatatan aplikasi magang (Relasi User-Lowongan)
        $regis_success = Registration::createRegistration($user_id, $lowongan_id);

        if ($profile_success && $regis_success) {
            $pdo->commit();
            $_SESSION['success_message'] = "Pendaftaran magang berhasil!";
            header('Location: ' . url_for('halaman_utama'));
            exit();
        } else {
            throw new Exception("Gagal menyimpan data ke database.");
        }
    } catch (Exception $e) {
        // Pembatalan transaksi jika terjadi eksepsi
        $pdo->rollBack();
        $errors[] = "Detail Error: " . $e->getMessage();
    }
}

/**
```

```
 * 4. ERROR HANDLING & REDIRECT
 */
if (!empty($errors)) {
    // Menyimpan state error dan data lama (sticky form) ke dalam session
    $_SESSION['form_errors'] = $errors;
    $_SESSION['form_data']   = $_POST;
    header('Location: ' . url_for('pendaftaran') . '&lowongan_id=' . $lowongan_id);
    exit();
}
```

```
if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
    header('Location: ' . url_for('pendaftaran'));
    exit();
}
```

Mencegah handler pendaftaran diakses langsung melalui URL dan memastikan data hanya diproses dari form resmi.

```
if (!$is_logged_in || $user_role !== 'user') {
    $_SESSION['auth_error'] = "Anda harus login sebagai Pendaftar.";
```

Menjamin hanya pengguna dengan role **pendaftar (user)** yang dapat melakukan pendaftaran magang.

```
require_once ROOT_PATH . 'app/models/Pendaftar.php';
require_once ROOT_PATH . 'app/models/Registration.php';
```

Menghubungkan controller dengan model yang menangani data profil pendaftar dan log pendaftaran magang.

```
$nama          = trim($_POST['nama'] ?? '');
$nik           = trim($_POST['NIK'] ?? '');
```

Membersihkan input pengguna agar konsisten, aman, dan siap divalidasi lebih lanjut.

```
if (!preg_match('/^\d{16}$/', $nik)) {
    $errors[] = "NIK harus terdiri dari 16 digit angka.";
```

Menjamin data yang masuk sesuai dengan aturan administratif dan kebutuhan sistem.

```
    if (move_uploaded_file($_FILES['cv']['tmp_name'], $upload_dir . $cv_name)) {
```

Mengelola unggahan dokumen pendukung (CV dan portofolio) secara aman dan terstruktur.

```
if (empty($errors)) {
  // Mapping data ke array asosiatif untuk layer Model
  $pendaftar_data = [
    'user_id'        => $user_id,
    'NIK'            => $nik,
```

Mengemas seluruh data pendaftaran agar terstruktur sebelum dikirim ke layer Model.

```
    $pdo->beginTransaction();
```

Menjamin **konsistensi data** dengan memastikan semua operasi berhasil atau dibatalkan seluruhnya.

```
    $profile_success = Pendaftar::updateFullProfile($pendaftar_data);
```

Memperbarui profil lengkap pendaftar agar selalu sinkron dengan data terbaru dari formulir.

```
    $regis_success = Registration::createRegistration($user_id, $lowongan_id);
```

Mencatat hubungan antara pengguna dan lowongan magang yang didaftarkan.

```
      $pdo->commit();
    $pdo->rollBack();
```

Menentukan apakah perubahan data disimpan permanen atau dibatalkan saat terjadi error.

```
  $_SESSION['form_errors'] = $errors;
  $_SESSION['form_data']   = $_POST;
```

Menampilkan pesan error dan mempertahankan data input agar pengguna tidak perlu mengisi ulang.

- handlers/internship_submit.php

```
<?php
```

```php
/**
 * internship_submit.php
 * Handler yang bertanggung jawab untuk memproses form penambahan lowongan magang.
 */

// Pastikan hanya bisa diakses via POST dan hanya oleh Perusahaan
if ($_SERVER['REQUEST_METHOD'] !== 'POST' || ($user_role ?? 'guest') !== 'company') {
    header('Location: ' . url_for('halaman_utama'));
    exit();
}

// 1. Ambil data dari POST
$data = $_POST;
$company_id = $_SESSION['user_id'];
$company_name = $_SESSION['user_name']; // Menggunakan nama perusahaan yang login

// 2. Validasi Input Kritis
$required_fields = ['posisi', 'deskripsi_pekerjaan', 'tanggal_mulai', 'tanggal_selesai'];
$errors = [];

foreach ($required_fields as $field) {
    if (empty($data[$field])) {
        $errors[] = "Kolom '{$field}' wajib diisi.";
    }
}

if (!empty($errors)) {
    $_SESSION['internship_message'] = ['type' => 'error', 'text' => implode('<br>', $errors)];
    $_SESSION['old_input'] = $data;
    header('Location: ' . url_for('add_internship'));
    exit();
```

```
}

// 3. Penanganan Upload File Logo
$logo_url = null;
if (isset($_FILES['logo_file']) && $_FILES['logo_file']['error'] === UPLOAD_ERR_OK) {
    $upload_dir = ROOT_PATH . 'public/uploads/logos/';
    if (!is_dir($upload_dir)) {
        mkdir($upload_dir, 0777, true);
    }

    $file_info = pathinfo($_FILES['logo_file']['name']);
    $file_ext = strtolower($file_info['extension']);
    $allowed_ext = ['jpg', 'jpeg', 'png', 'gif'];

    if (in_array($file_ext, $allowed_ext)) {
        // Buat nama file unik: companyID_timestamp.ext
        $file_name = $company_id . '_' . time() . '.' . $file_ext;
        $target_path = $upload_dir . $file_name;

        if (move_uploaded_file($_FILES['logo_file']['tmp_name'], $target_path)) {
            // Path yang disimpan di database adalah path relatif dari public/
            $logo_url = 'uploads/logos/' . $file_name;
        } else {
            $errors[] = "Gagal memindahkan file logo.";
        }
    } else {
        $errors[] = "Jenis file logo tidak didukung.";
    }
}

if (!empty($errors)) {
```

```
    $_SESSION['internship_message'] = ['type' => 'error', 'text' => implode('<br>', $errors)];
    $_SESSION['old_input'] = $data;
    header('Location: ' . url_for('add_internship'));
    exit();
}

// 4. Persiapan Data Akhir untuk Model
$final_data = [
    'company_id'            => $company_id,
    'perusahaan'            => $company_name, // Mengambil dari sesi
    'posisi'                => $data['posisi'],
    'ringkasan'             => $data['ringkasan'] ?? null,
    'kualifikasi_jurusan'   => $data['kualifikasi_jurusan'] ?? null,
    'durasi'                => $data['durasi'] ?? null,
    'lokasi_penempatan'     => $data['lokasi_penempatan'] ?? null,
    'deskripsi_pekerjaan'   => $data['deskripsi_pekerjaan'],
    'requirements'          => $data['requirements'] ?? null,
    'tanggal_mulai'         => $data['tanggal_mulai'] . ' 00:00:00', // Tambah waktu
    'tanggal_selesai'       => $data['tanggal_selesai'] . ' 23:59:59', // Tambah waktu
    'website'               => $data['website'] ?? null,
    'instagram_link'        => $data['instagram_link'] ?? null,
    'logo_url'              => $logo_url ?? 'img/default_logo.png', // Default jika tidak ada upload
];


// 5. Simpan ke Database
$success = Internship::createPosting($final_data);

if ($success) {
    $_SESSION['internship_message'] = ['type' => 'success', 'text' => "Lowongan magang
'{$final_data['posisi']}' berhasil diposting!"];
    // Redirect ke halaman informasi (info.php) atau dashboard perusahaan
```

```
    header('Location: ' . url_for('info'));
    exit();
} else {
    // Penanganan error database
    $_SESSION['internship_message'] = ['type' => 'error', 'text' => "Gagal menyimpan lowongan ke database. Coba lagi."];
    $_SESSION['old_input'] = $data;
    header('Location: ' . url_for('add_internship'));
    exit();
}
```

```
if ($_SERVER['REQUEST_METHOD'] !== 'POST' || ($user_role ?? 'guest') !== 'company') {
    header('Location: ' . url_for('halaman_utama'));
    exit();
}
```

Menjamin hanya perusahaan yang sudah login **dan** melalui form POST yang dapat menambahkan lowongan magang.

```
$company_id = $_SESSION['user_id'];
$company_name = $_SESSION['user_name'];
```

Mengikat lowongan magang dengan akun perusahaan yang sedang aktif tanpa mempercayai input dari client.

```
$data = $_POST;
```

Mengambil seluruh data form sebagai basis validasi dan pemrosesan berikutnya.

```
$required_fields = ['posisi', 'deskripsi_pekerjaan', 'tanggal_mulai', 'tanggal_selesai'];
```

Menjamin informasi inti lowongan tersedia sebelum data disimpan ke database.

```
    $_SESSION['internship_message'] = ['type' => 'error', 'text' => implode('<br>', $errors)];
```

Memberikan umpan balik ke pengguna dan menjaga UX dengan menyimpan input lama.

```
        if (move_uploaded_file($_FILES['logo_file']['tmp_name'], $target_path)) {
```

Mengelola unggahan logo perusahaan secara aman dan terstruktur.

```
$allowed_ext = ['jpg', 'jpeg', 'png', 'gif'];
```

Mencegah unggahan file berbahaya atau format yang tidak didukung.

```
$file_name = $company_id . '_' . time() . '.' . $file_ext;
```

Mencegah konflik nama file dan mempermudah pelacakan kepemilikan file.

```
$final_data = [
    'company_id'          => $company_id,
    'perusahaan'          => $company_name,
```

Mengemas data lowongan secara rapi sebelum dikirim ke layer Model.

```
$success = Internship::createPosting($final_data);
```

Menyimpan data lowongan magang ke database melalui model **Internship.**

```
if ($success) {
    $_SESSION['internship_message'] = ['type' => 'success', 'text' => "Lowongan magang '{$final_data['posisi']}' berhasil diposting!"];
```

Memberi notifikasi sukses dan mengarahkan perusahaan ke halaman lanjutan.

```
} else {
    // Penanganan error database
    $_SESSION['internship_message'] = ['type' => 'error', 'text' => "Gagal menyimpan lowongan ke database. Coba lagi."];
```

Menangani kegagalan penyimpanan tanpa membuat sistem crash.

4.2.5 Antarmuka Pengguna

- views/pages/login.php

```
<?php
/**
 * login.php
 * Konten utama untuk halaman Login.
```

```
 * File ini mengonversi tampilan Blade menjadi PHP Native murni.
 * Variabel global $is_logged_in, $user_name, dan url_for() sudah tersedia.
 */
$page_title = "Login";

// --- START TEMPLATE INCLUDES ---
require_once __DIR__ . '/../includes/header.php';

// Cek data error atau old input yang dikirim dari handler
$auth_error = $_SESSION['auth_error'] ?? null;
$old_input = $_SESSION['old_input'] ?? [];

// Bersihkan sesi setelah diambil
unset($_SESSION['auth_error']);
unset($_SESSION['old_input']);

// Logika pemuatan konten
?>

<!-- CONTAINER UTAMA HALAMAN (Mereplikasi <x-guest-layout>) -->
<div class="auth-page-container">

    <!-- KARTU FORMULIR -->
    <div class="auth-form-card">

        <?php if ($auth_error): ?>
            <!-- Menampilkan pesan error dari handler -->
            <div class="session-status error-status">
                <?= htmlspecialchars($auth_error) ?>
            </div>
        <?php endif; ?>
```

```
    <!-- Judul form -->
    <div class="form-title">Login</div>

    <!-- FORM LOGIC: action diarahkan ke handler POST via index.php -->
    <form method="POST" action="index.php?page=login_submit" class="login-form">

        <!-- Email Address -->
        <div class="form-group">
            <label for="email" class="form-label">Email</label>

            <!-- Menggunakan old input dari sesi agar nilai tidak hilang saat validasi gagal -->
            <input id="email" type="email" name="email"
                value="<?= htmlspecialchars($old_input['email'] ?? '') ?>"
                required autofocus autocomplete="username"
                class="form-input">

            <!-- Catatan: Pengecekan error spesifik (@error) diabaikan karena diringkas menjadi
$auth_error -->
        </div>

        <!-- Password -->
        <div class="form-group">
            <label for="password" class="form-label">Password</label>
                    <input id="password" type="password" name="password" required
autocomplete="current-password"
                class="form-input">
        </div>

        <div class="form-actions">
            <!-- Tombol Register -->
```

```
            <a href="<?= url_for('register') ?>" class="btn-register">
                Register
            </a>

            <!-- Tombol Submit -->
            <button type="submit" class="btn-login-submit">
                Log in
            </button>
        </div>
    </form>
  </div>
</div>

<?php
// --- END TEMPLATE INCLUDES ---
require_once __DIR__ . '/../includes/footer.php';
?>
```

```
$page_title = "Login";
```

Menentukan identitas halaman yang akan digunakan oleh template global (misalnya <title> di header).

```
require_once __DIR__ . '/../includes/header.php';
```

Menyatukan halaman login dengan kerangka UI global (navbar, head HTML, CSS, dll).

```
$auth_error = $_SESSION['auth_error'] ?? null;
$old_input = $_SESSION['old_input'] ?? [];
```

Mencegah pesan error dan input lama muncul kembali saat halaman direfresh.

```
<div class="auth-page-container">
```

Menjadi pembungkus utama tampilan login agar konsisten dengan halaman autentikasi lainnya.

```
    <?php if ($auth_error): ?>
```

Menampilkan pesan kesalahan **hanya jika** proses login sebelumnya gagal.

```
            <?= htmlspecialchars($auth_error) ?>
```

Mencegah serangan XSS dengan memastikan pesan yang ditampilkan aman.

```
        <form method="POST" action="index.php?page=login_submit" class="login-form">
```

Mengirimkan data login ke handler autentikasi melalui Front Controller (index.php).

```
                value="<?= htmlspecialchars($old_input['email'] ?? '') ?>"
```

Menjaga pengalaman pengguna agar tidak perlu mengetik ulang email jika login gagal.

```
            <input id="email" type="email" name="email"
```

Memberikan validasi awal sebelum data dikirim ke server.

```
            <a href="<?= url_for('register') ?>" class="btn-register">
```

Mengarahkan pengguna baru ke halaman pendaftaran tanpa hardcode URL.

```
            <button type="submit" class="btn-login-submit">
```

Memicu pengiriman kredensial ke handler backend.

```
require_once __DIR__ . '/../includes/footer.php';
```

Menutup struktur HTML dan memuat elemen UI global bagian bawah.

- view/pages/pendaftaran/php

```
<?php
/**
 * pendaftaran.php
 * View untuk formulir pendaftaran magang dengan sistem Multi-Step.
 */

global $is_logged_in, $user_role;

/**
 * 1. PROTEKSI AKSES & OTORISASI
```

```
 */
if (!$is_logged_in || $user_role !== 'user') {
    $_SESSION['auth_error'] = "Anda harus login sebagai Pendaftar untuk mengakses formulir.";
    header('Location: ' . url_for('login'));
    exit;
}

require_once ROOT_PATH . 'app/models/Pendaftar.php';
require_once ROOT_PATH . 'app/models/Internship.php';

/**
 * 2. PRE-LOADING DATA PROFIL & VALIDASI LOWONGAN
 */
$user_id       = $_SESSION['user_id'];
$profile       = Pendaftar::getProfileByUserId($user_id);
$existing_nik   = $profile['NIK'] ?? '';
$existing_email = $profile['email'] ?? $_SESSION['user_email'] ?? '';

// Validasi ID Lowongan dari parameter URL
$lowongan_id = filter_var($_GET['lowongan_id'] ?? null, FILTER_VALIDATE_INT);
$lowongan    = $lowongan_id ? Internship::getPostingById($lowongan_id) : false;

if (!$lowongan) {
    $_SESSION['message'] = [
        'type' => 'error',
        'text' => "ID Lowongan tidak valid atau lowongan tidak ditemukan. Silakan pilih lowongan
kembali."
    ];
    header('Location: ' . url_for('info'));
    exit;
}
```

```php
// Konfigurasi metadata halaman
$page_title = "Pendaftaran Magang - " . htmlspecialchars($lowongan['posisi']);

/**
 * 3. MANAJEMEN STATE FORM (Error Handling & Sticky Data)
 */
$success_message = $_SESSION['success_message'] ?? null;
$form_errors     = $_SESSION['form_errors'] ?? [];
$old_data        = $_SESSION['form_data'] ?? [];

// Flash Session Clean-up: Menghapus data sesi setelah dimuat ke variabel
unset($_SESSION['success_message']);
unset($_SESSION['form_errors']);
unset($_SESSION['form_data']);
?>

<div class="form-title">Pendaftaran Internship: <?= htmlspecialchars($lowongan['posisi']) ?></div>

<div class="form-wrapper">

    <?php if (!empty($form_errors)): ?>
        <div class="session-status error-status">
            <h3>Pendaftaran Gagal:</h3>
            <ul>
                <?php foreach ($form_errors as $error): ?>
                    <li><?= htmlspecialchars($error) ?></li>
                <?php endforeach; ?>
            </ul>
        </div>
```

```
  <?php endif; ?>

  <div class="form-container">
    <form id="multiStepForm"
          method="POST"
          action="index.php?page=submit_pendaftaran"
          enctype="multipart/form-data"
          novalidate>

        <input type="hidden" name="lowongan_id" value="<?= htmlspecialchars($lowongan_id)
?>">

        <div class="form-step active">
          <div class="form-header">1. Data Diri</div>

          <div class="input-group">
            <label>Nama Lengkap</label>
            <input type="text" name="nama" placeholder="Masukkan nama lengkap"
                   value="<?= htmlspecialchars($old_data['nama'] ?? $_SESSION['user_name'] ??
'') ?>" required>
          </div>

          <div class="input-group">
            <label>NIK (Nomor Induk Kependudukan) *</label>
            <input type="text" name="NIK" placeholder="Masukkan 16 digit NIK"
                   value="<?= htmlspecialchars($old_data['NIK'] ?? $existing_nik) ?>"
                   maxlength="16" pattern="\d{16}" required
                   title="NIK harus 16 digit angka.">
            <small class="form-text text-muted">NIK ini akan disimpan ke profil Anda.</small>
          </div>
```

```
            <div class="input-group">
                <label>Jenis Kelamin</label>
                <select name="jenis_kelamin" required>
                    <option value ="" disabled selected hidden>Pilih</option>
                        <option value ="lakilaki" <?= ($old_data['jenis_kelamin'] ?? '') === 'lakilaki' ?
'selected' : '' ?>>Laki-laki</option>
                    <option value ="perempuan" <?= ($old_data['jenis_kelamin'] ?? '') === 'perempuan'
? 'selected' : '' ?>>Perempuan</option>
                </select>
            </div>

            <div class="input-group">
                <label>Tanggal Lahir</label>
                <input type="date" name="tanggal_lahir"
                    value="<?= htmlspecialchars($old_data['tanggal_lahir'] ?? '') ?>" required>
            </div>

            <div class="input-group">
                <label>Alamat Domisili</label>
                <input type="text" name="alamat" placeholder="Alamat"
                    value="<?= htmlspecialchars($old_data['alamat'] ?? '') ?>" required>
            </div>

            <div class="input-group">
                <label>Nomor HP / WhatsApp</label>
                <input type="text" name="nomor_hp" placeholder="08xxxxxxxxxx"
                    value="<?= htmlspecialchars($old_data['nomor_hp'] ?? '') ?>" required>
            </div>

            <div class="input-group">
                <label>Email Aktif</label>
```

```
            <input type="email" name="email" placeholder="example@gmail.com"
                value="<?= htmlspecialchars($old_data['email'] ?? $existing_email) ?>" required>
        </div>

        <div class="form-navigation">
            <button type="button" class="btn-next">Selanjutnya</button>
        </div>
    </div>

    <div class="form-step">
        <div class="form-header">2. Data Pendidikan</div>

        <div class="input-group">
            <label>Pendidikan Terakhir</label>
            <select name="pendidikan" required>
                <option value ="" disabled selected hidden>Pilih</option>
                 <option value = "SD" <?= ($old_data['pendidikan'] ?? '') === 'SD' ? 'selected' : '' ?>>SD</option>
                <option value = "SMP" <?= ($old_data['pendidikan'] ?? '') === 'SMP' ? 'selected' : '' ?>>SMP</option>
                <option value = "SMA / SMK / MA" <?= ($old_data['pendidikan'] ?? '') === 'SMA / SMK / MA' ? 'selected' : '' ?>>SMA / SMK / MA</option>
                 <option value = "Diploma D4" <?= ($old_data['pendidikan'] ?? '') === 'Diploma D4' ? 'selected' : '' ?>>Diploma D4 / D3</option>
                <option value = "Sarjana S1" <?= ($old_data['pendidikan'] ?? '') === 'Sarjana S1' ? 'selected' : '' ?>>Sarjana S1</option>
            </select>
        </div>

        <div class="input-group">
            <label>Asal sekolah / Universitas</label>
```

```
            <input type="text" name="instansi" placeholder="Ex: Universitas Indonesia / SMAN
1 Surabaya"
                value="<?= htmlspecialchars($old_data['instansi'] ?? '') ?>" required>
        </div>

        <div class="input-group">
            <label>Program studi</label>
            <input type="text" name="prodi" placeholder="Nama program studi"
                value="<?= htmlspecialchars($old_data['prodi'] ?? '') ?>" required>
        </div>

        <div class="input-group">
            <label>Jurusan</label>
            <input type="text" name="jurusan" placeholder="Nama Jurusan"
                value="<?= htmlspecialchars($old_data['jurusan'] ?? '') ?>" required>
        </div>

        <div class="input-group">
            <label>Semester</label>
            <input type="text" name="semester" placeholder="Ex: 7, jika bukan mahasiswa isi -
"
                value="<?= htmlspecialchars($old_data['semester'] ?? '') ?>" required>
        </div>

        <div class="input-group">
            <label>IPK (opsional)</label>
            <input type="text" name="ipk" placeholder="Ex: 3.75"
                value="<?= htmlspecialchars($old_data['ipk'] ?? '') ?>">
        </div>

        <div class="form-navigation">
```

```
                <button type="button" class="btn-prev">Sebelumnya</button>
                <button type="button" class="btn-next">Selanjutnya</button>
            </div>
        </div>

        <div class="form-step">
            <div class="form-header">3. Informasi Magang</div>

            <div class="input-group">
                <label>Posisi / Jabatan yang dilamar</label>
                <input type="text" name="jabatan" value="<?= htmlspecialchars($old_data['jabatan'] ?? '') ?>" required>
            </div>

            <div class="input-group">
                <label>Alasan mengikuti magang</label>
                <input type="text" name="alasan" value="<?= htmlspecialchars($old_data['alasan'] ?? '') ?>" required>
            </div>

            <div class="input-group">
                <label>Sumber info lowongan</label>
                <select name="sumber" required>
                    <option value="" disabled selected hidden>Pilih</option>
                    <option value="Social media (Instagram, Facebook, X)" <?= ($old_data['sumber'] ?? '') === 'Social media (Instagram, Facebook, X)' ? 'selected' : '' ?>>Social media (Instagram, Facebook, X)</option>
                    <option value="Media Partner" <?= ($old_data['sumber'] ?? '') === 'Media Partner' ? 'selected' : '' ?>>Media Partner</option>
                    <option value="Website Internship" <?= ($old_data['sumber'] ?? '') === 'Website Internship' ? 'selected' : '' ?>>Website Internship</option>
```

```
            </select>
        </div>

        <div class="form-navigation">
            <button type="button" class="btn-prev">Sebelumnya</button>
            <button type="button" class="btn-next">Selanjutnya</button>
        </div>
    </div>

    <div class="form-step">
        <div class="form-header">4. Dokumen Lampiran</div>

        <div class="input-group">
            <label>Upload CV (Format PDF)</label>
            <input type="file" id="cvFile" name="cv" accept=".pdf" required>
        </div>

        <div class="input-group">
            <label>Portofolio (Opsional)</label>
            <input type="file" id="portfolioFile" name="portofolio" accept=".pdf">
        </div>

        <div class="form-navigation">
            <button type="button" class="btn-prev">Sebelumnya</button>
            <button type="submit" class="btn-submit">Kirim</button>
        </div>
    </div>
</form>
</div>
</div>
```

```php
if (!$is_logged_in || $user_role !== 'user') {
    $_SESSION['auth_error'] = "Anda harus login sebagai Pendaftar untuk mengakses formulir.";
    header('Location: ' . url_for('login'));
    exit;
}
```

Menjamin hanya user berperan pendaftar yang dapat mengakses formulir. Ini mencegah:

- Guest mengakses form
- Perusahaan/Admin mengisi pendaftaran magang

```php
require_once ROOT_PATH . 'app/models/Pendaftar.php';
require_once ROOT_PATH . 'app/models/Internship.php';
```

Menghubungkan View dengan data:

- Pendaftar → data profil user
- Internship → data lowongan magang

```php
$profile       = Pendaftar::getProfileByUserId($user_id);
```

Mengambil data yang **sudah pernah diisi** agar:

- NIK
- Email

tidak perlu diinput ulang.

```php
$lowongan_id = filter_var($_GET['lowongan_id'] ?? null, FILTER_VALIDATE_INT);
$lowongan    = $lowongan_id ? Internship::getPostingById($lowongan_id) : false;
```

Memastikan:

- ID lowongan valid

- Lowongan benar-benar ada di database

Jika gagal → user dikembalikan ke halaman info.

```
$page_title = "Pendaftaran Magang - " . htmlspecialchars($lowongan['posisi']);
```

Menyesuaikan judul halaman berdasarkan posisi magang yang dipilih.

```
$form_errors     = $_SESSION['form_errors'] ?? [];
$old_data        = $_SESSION['form_data'] ?? [];
```

- Menampilkan error validasi
- Mengisi ulang input lama jika submit gagal

```
unset($_SESSION['form_errors']);
unset($_SESSION['form_data']);
```

Menjamin data error **hanya muncul sekali** (tidak berulang saat refresh).

```
        <div class="form-step active">
        <div class="form-step">
```

Membagi form besar menjadi beberapa tahap:

1. Data diri
2. Pendidikan
3. Informasi magang
4. Dokumen

meningkatkan UX & keterbacaan

```
        <input type="hidden" name="lowongan_id" value="<?= htmlspecialchars($lowongan_id) ?>">
```

Menjaga relasi pendaftar ↔ lowongan saat form dikirim ke handler.

```
value="<?= htmlspecialchars($old_data['nama'] ?? $_SESSION['user_name'] ?? '') ?>" required>
```

Menghindari kehilangan data saat validasi gagal.

```
                <button type="button" class="btn-prev">Sebelumnya</button>
                <button type="button" class="btn-next">Selanjutnya</button>
```

Navigasi antar tahap tanpa submit form.

```
            action="index.php?page=submit_pendaftaran"
```

Mengirim seluruh data ke Controller (form_pendaftaran.php) untuk:

- Validasi
- Upload
- Transaksi database

- view/admin/dashboard.php

```
<?php
// Pastikan koneksi $pdo sudah ada dari index.php / connection.php

// Jumlah pengguna (role = user)
$stmtUser = $pdo->prepare("SELECT COUNT(*) FROM users WHERE role = 'user'");
$stmtUser->execute();
$totalUser = $stmtUser->fetchColumn();

// Jumlah perusahaan (role = company)
$stmtCompany = $pdo->prepare("SELECT COUNT(*) FROM users WHERE role = 'company'");
$stmtCompany->execute();
$totalCompany = $stmtCompany->fetchColumn();

// Jumlah lowongan magang (sesuaikan nama tabel!)
$totalLowongan = $pdo
    ->query("SELECT COUNT(*) FROM lowongan_magang") // atau nama tabel kamu
```

```
    ->fetchColumn();
?>

<div class="admin-dashboard">
    <h1>Dashboard</h1>

    <div class="admin-stats">
        <a href="index.php?page=admin_users" class="admin-card link-card">
            <h2><?= $totalUser ?></h2>
            <p>Pengguna</p>
        </a>

        <a href="index.php?page=admin_company" class="admin-card link-card">
            <h2><?= $totalCompany ?></h2>
            <p>Perusahaan</p>
        </a>

        <a href="index.php?page=admin_lowongan" class="admin-card link-card">
            <h2><?= $totalLowongan ?></h2>
            <p>Lowongan Magang</p>
        </a>
    </div>
</div>
```

```
$stmtUser = $pdo->prepare("SELECT COUNT(*) FROM users WHERE role = 'user'");
$stmtUser->execute();
$totalUser = $stmtUser->fetchColumn();
```

Mengambil data agregat (COUNT) langsung dari database menggunakan PDO. Dipakai untuk statistik ringkas, bukan data detail.

```
$stmtCompany = $pdo->prepare("SELECT COUNT(*) FROM users WHERE role = 'company'");
```

Memisahkan statistik berdasarkan peran pengguna**,** yaitu:

- User (pendaftar)
- Company (perusahaan)

```
$totalLowongan = $pdo
    ->query("SELECT COUNT(*) FROM lowongan_magang") // atau nama tabel kamu
    ->fetchColumn();
```

Menghitung total lowongan magang yang tersedia.

```
<div class="admin-dashboard">
```

Menyediakan overview sistem bagi admin tanpa detail teknis.

Digunakan sebagai:

- Entry point monitoring
- Navigasi cepat ke modul lain

```
        <a href="index.php?page=admin_users" class="admin-card link-card">
```

Menggabungkan:

- Informasi (jumlah)
- Navigasi (klik menuju halaman detail)

Ini mengurangi langkah admin (UX-oriented).

```
            <h2><?= $totalUser ?></h2>
```

Menampilkan data **real-time** langsung dari database saat halaman dimuat.

```
        <a href="index.php?page=admin_lowongan" class="admin-card link-card">
```

Menggunakan satu entry point (index.php) untuk:

- Kontrol halaman
- Pemisahan logika via parameter

- view/admin/users.php

```
// PROSES TOGGLE STATUS
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['user_id'], $_POST['current_status'])) {
    $userId = $_POST['user_id'];
    $currentStatus = $_POST['current_status'];
```

Bagian ini berfungsi untuk mengecek apakah halaman menerima permintaan POST serta memastikan data user_id dan current_status tersedia. Proses ini terjadi saat admin menekan tombol status pengguna. Data user_id digunakan untuk menentukan pengguna yang akan diubah statusnya, sedangkan current_status digunakan untuk mengetahui status pengguna saat ini.

```
    $newStatus = ($currentStatus === 'active') ? 'banned' : 'active';
```

Logika ini digunakan untuk mengganti status pengguna:

- Jika status sebelumnya *active*, maka diubah menjadi *banned*
- Jika status sebelumnya *banned*, maka diubah menjadi *active*

```
    $stmt = $pdo->prepare("UPDATE users SET status = ? WHERE id = ?");
    $stmt->execute([$newStatus, $userId]);
```

Query ini digunakan untuk memperbarui status pengguna pada tabel users berdasarkan id pengguna. Penggunaan prepare dan execute bertujuan untuk meningkatkan keamanan dari serangan SQL Injection.

```
    header("Location: index.php?page=admin_users");
    exit;
```

Setelah status berhasil diperbarui, halaman akan direfresh agar perubahan status langsung ditampilkan.

```php
// ambil semua user yang role = user
$stmt = $pdo->prepare("SELECT id, name, email, status FROM users WHERE role = 'user'");
$stmt->execute();
$users = $stmt->fetchAll();
```

Query ini digunakan untuk mengambil seluruh data pengguna yang memiliki role user, kemudian hasilnya disimpan dalam variabel $users dalam bentuk array.

```php
<div class="admin-dashboard">
    <h1>Pengguna</h1>
```

Bagian ini merupakan struktur utama halaman dashboard admin untuk menampilkan data pengguna.

```php
        <?php if (empty($users)) : ?>
            <p>Tidak ada pengguna.</p>
```

Jika data pengguna kosong, maka sistem akan menampilkan pesan bahwa tidak terdapat pengguna yang terdaftar.

```php
            <?php foreach ($users as $user) : ?>
```

Perulangan ini digunakan untuk menampilkan setiap data pengguna satu per satu dalam bentuk kartu pengguna.

```php
                        <strong class="user-name">
                            <?= htmlspecialchars($user['name']) ?>
                        </strong>
                        <span class="user-email">
                            <?= htmlspecialchars($user['email']) ?>
                        </span>
```

Fungsi htmlspecialchars() digunakan untuk mencegah serangan Cross-Site Scripting (XSS) dengan mengamankan data sebelum ditampilkan ke halaman web.

```php
                        <form method="POST" style="display:inline;">
                            <input type="hidden" name="user_id" value="<?= $user['id']; ?>">
                            <input type="hidden" name="current_status" value="<?= $user['status']; ?>">
```

Form ini digunakan untuk mengirim data ke server saat admin mengubah status pengguna. Input tersembunyi digunakan untuk mengirimkan ID pengguna dan status saat ini tanpa menampilkannya ke layar.

```
                    <?php if ($user['status'] === 'active') : ?>
```

Jika status pengguna aktif, tombol akan menampilkan label Active dengan kelas CSS tertentu. Jika status banned, maka tombol akan menampilkan label Banned.

Tombol ini sekaligus berfungsi sebagai toggle, sehingga admin dapat langsung mengubah status hanya dengan satu klik.

- view/admin/company.php

```
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['company_id'], $_POST['current_status'])) {
    $companyId = $_POST['company_id'];
    $currentStatus = $_POST['current_status'];
```

Kode ini melakukan pengecekan apakah permintaan yang diterima merupakan metode POST serta memastikan data company_id dan current_status tersedia. Proses ini dijalankan ketika admin menekan tombol status perusahaan.

Variabel $companyId digunakan untuk menyimpan ID perusahaan yang akan diubah statusnya, sedangkan $currentStatus menyimpan status perusahaan sebelum diubah.

```
    $newStatus = ($currentStatus === 'active') ? 'banned' : 'active';
```

Logika ini digunakan untuk menentukan status baru perusahaan:

- Jika status sebelumnya active, maka diubah menjadi banned
- Jika status sebelumnya banned, maka diubah menjadi active

```
    $stmt = $pdo->prepare("UPDATE users SET status = ? WHERE id = ?");
    $stmt->execute([$newStatus, $companyId]);
```

Query ini digunakan untuk memperbarui status perusahaan pada tabel users berdasarkan ID perusahaan. Penggunaan prepare dan execute bertujuan untuk meningkatkan keamanan sistem terhadap serangan SQL Injection.

```
    // refresh halaman
    header("Location: index.php?page=admin_company");
    exit;
```

Setelah proses perubahan status berhasil dilakukan, halaman akan diarahkan ulang (refresh) agar data status yang ditampilkan langsung diperbarui.

```
$stmt = $pdo->prepare("SELECT id, name, email, status FROM users WHERE role = 'company'");
$stmt->execute();
$users = $stmt->fetchAll();
```

Bagian ini digunakan untuk mengambil seluruh data akun perusahaan yang memiliki role company dari tabel users. Data yang diambil meliputi ID, nama perusahaan, email, dan status akun.

```
<div class="admin-dashboard">
    <h1>Perusahaan</h1>
```

Bagian ini merupakan struktur utama halaman dashboard admin yang digunakan untuk menampilkan daftar perusahaan.

```
        <?php if (empty($users)) : ?>
            <p>Tidak ada perusahaan.</p>
```

Jika data perusahaan tidak tersedia atau kosong, maka sistem akan menampilkan pesan bahwa tidak terdapat perusahaan yang terdaftar.

```
            <?php foreach ($users as $user) : ?>
```

Perulangan ini digunakan untuk menampilkan setiap data perusahaan satu per satu dalam bentuk kartu perusahaan.

```
                    <strong class="user-name">
                        <?= htmlspecialchars($user['name']) ?>
                    </strong>
                    <span class="user-email">
                        <?= htmlspecialchars($user['email']) ?>
                    </span>
```

Fungsi htmlspecialchars() digunakan untuk mengamankan data yang ditampilkan pada halaman web guna mencegah serangan Cross-Site Scripting (XSS).

```
                    <form method="POST" style="display:inline;">
                        <input type="hidden" name="company_id" value="<?= $user['id']; ?>">
                        <input type="hidden" name="current_status" value="<?= $user['status']; ?>">
```

Form ini digunakan sebagai media pengiriman data saat admin melakukan perubahan status perusahaan.

Input tersembunyi digunakan untuk mengirimkan ID perusahaan dan status saat ini tanpa perlu menampilkannya ke halaman.

```
                        <?php if ($user['status'] === 'active') : ?>
```

Jika status perusahaan aktif, maka sistem akan menampilkan tombol dengan label *Active*. Sebaliknya, jika status perusahaan banned, tombol akan menampilkan label *Banned*. Tombol ini berfungsi sebagai toggle status yang memungkinkan admin mengubah status perusahaan secara langsung.

- view/admin/lowongan.php

```
<?php
$lowongan_list = Internship::getAllPostingsForAdmin();
?>
```

Baris kode ini digunakan untuk mengambil seluruh data lowongan magang dari database melalui method getAllPostingsForAdmin() pada kelas Internship. Method ini mengembalikan data

lowongan dalam bentuk array yang berisi informasi posisi, perusahaan, ringkasan, hingga status perusahaan.

```
<div class="admin-dashboard">
    <h1>Lowongan Magang</h1>
```

Bagian ini merupakan struktur utama halaman dashboard admin yang menampilkan daftar lowongan magang.

```
    <?php if (empty($lowongan_list)) : ?>
        <p>Tidak ada lowongan.</p>
```

Jika tidak terdapat data lowongan magang, sistem akan menampilkan pesan bahwa tidak ada lowongan yang tersedia.

```
        <?php foreach ($lowongan_list as $post) : ?>
```

Perulangan ini digunakan untuk menampilkan setiap data lowongan magang satu per satu dalam bentuk kartu lowongan.

```
                    <strong class="user-name">
                        <?= htmlspecialchars($post['posisi']) ?>
                    </strong>
                    <span class="user-email">
                        <?= htmlspecialchars($post['perusahaan']) ?>
                    </span>
                    <small>
                        <?= htmlspecialchars($post['ringkasan']) ?>
                    </small>
```

Bagian ini menampilkan informasi singkat lowongan yang meliputi posisi magang, nama perusahaan, serta ringkasan lowongan. Fungsi htmlspecialchars() digunakan untuk mengamankan data sebelum ditampilkan ke halaman web guna mencegah serangan Cross-Site Scripting (XSS).

```
<div class="button-detail-lowongan">
    <button class="btn-detail"
            onclick="toggleDetail(<?= $post['id'] ?>)">
        Detail Selengkapnya
    </button>
```

Tombol ini digunakan untuk menampilkan atau menyembunyikan detail lowongan magang secara dinamis tanpa perlu memuat ulang halaman.

```
<!-- DETAIL (HIDDEN) -->
<div class="lowongan-detail" id="detail-<?= $post['id'] ?>">
```

Bagian ini berisi detail lengkap lowongan magang dan secara default disembunyikan menggunakan CSS. Setiap detail memiliki ID unik berdasarkan ID lowongan.

```
    <p><strong>Kualifikasi Jurusan:</strong><br>
        <?= nl2br(htmlspecialchars($post['kualifikasi_jurusan'])) ?>
    </p>
```

Fungsi nl2br() digunakan untuk mengubah baris baru pada teks menjadi tag
, sehingga format paragraf tetap terjaga saat ditampilkan.

Detail yang ditampilkan meliputi:

- Kualifikasi jurusan
- Durasi magang
- Lokasi penempatan
- Deskripsi pekerjaan
- Requirements
- Periode magang (tanggal mulai dan selesai)
- Website perusahaan (opsional)
- Akun Instagram perusahaan (opsional)

Data website dan Instagram hanya ditampilkan jika tersedia, sehingga tampilan halaman tetap rapi dan dinamis.

```
<div class="user-actions">
    <?php if ($post['company_status'] === 'active') : ?>
        <span class="statusLowonganActive">Status : Active</span>
    <?php else : ?>
        <span class="statusLowonganBanned">Status : Company Banned</span>
    <?php endif; ?>
</div>
```

Bagian ini menampilkan status lowongan berdasarkan status akun perusahaan:

- Jika status perusahaan active, maka lowongan ditampilkan sebagai *Active*
- Jika status perusahaan banned, maka lowongan diberi keterangan *Company Banned*

Hal ini menunjukkan bahwa status lowongan bergantung pada status akun perusahaan yang membuat lowongan tersebut.

```
<script>
function toggleDetail(id) {
    const detail = document.getElementById('detail-' + id);
    detail.classList.toggle('show');
}
</script>
```

Fungsi JavaScript ini digunakan untuk menampilkan dan menyembunyikan detail lowongan magang dengan cara menambahkan atau menghapus class show pada elemen detail.

BAB V
PENUTUP

5.1 Pembagian tugas project

Berikut merupakan table daftar anggota serta pembagian pekerjaan masing masing anggota dalam project tugas akhir.

Nama	NIM	Pembagian Tugas
Nazzala Risky Nafi'a	24091397009	- Halaman Pendaftaran Lowongan Magang, dan Dashboard Admin (Termasuk CSS, CSS Dark Mode, dan JavaScript Halaman Tersebut) - Pengerjaan Laporan Bab II (Daftar Pengguna, Daftar Fitur) dan Menjelaskan Code yang berkaitan dengan Actor Admin pada Bab IV. - Penambahan Database User Admin pada Tabel Users - Mengerjakan PPT slide Daftar Pengguna dan Daftar Fitur - Membantu dalam diskusi dan Pengerjaan pada Project
Zaryan Nugraha Islah	24091397010	- Halaman Informasi Lowongan Magang, Login Page, Register Page, Tampilan Actor Pelamar Magang, Dashboard Perusahaan (Termasuk CSS, CSS Dark Mode, dan JavaScript Halaman Tersebut). - Membuat dan Menjelaskan Keseluruhan dari Laporan Bab III dan Bab IV (**Kecuali** Penjelasan Code Actor Admin) - Membuat Perancangan Database (Termasuk Tabel) serta Menghubungkan dari Web ke Database. - Membantu dalam pengerjaan PPT slide Flowchart, Use Case, Architecture Design Diagram, Basis Data). - Membantu dalam diskusi dan Pengerjaan pada Project
Randy Pradana Bintang O	24091397018	- Halaman Beranda/Halaman Utama Web (Termasuk CSS, CSS Dark Mode, dan JavaScript Halaman Tersebut)

		- Pengerjaan Laporan Bab I (Latar Belakang, Permasalahan, Tujuan Project) dan Bab V (Kesimpulan) - Mencarikan Template PPT dan mengerjakan slide Analisis Masalah & Solusi, Tujuan Project, Manfaat Project. - Membantu dalam diskusi dan Pengerjaan pada Project

5.2 Kesimpulan

Website Campus Intern berhasil dikembangkan sebagai sistem informasi lowongan magang berbasis web menggunakan PHP Native dan database MySQL. Sistem ini dirancang dengan tiga jenis peran pengguna, yaitu user, company, dan admin, yang masing-masing memiliki hak akses dan fitur yang berbeda sesuai dengan kebutuhan. Pengguna (*user*) dapat melihat dan mencari informasi lowongan magang, perusahaan (*company*) dapat mengelola dan mempublikasikan lowongan magang, sedangkan admin bertugas mengelola data pengguna, perusahaan, serta lowongan magang yang tersedia di dalam sistem. Dengan adanya pembagian role tersebut, pengelolaan sistem menjadi lebih terstruktur dan aman.

Fitur-fitur utama yang telah diimplementasikan meliputi manajemen akun pengguna dan perusahaan, pengelolaan status akun (aktif dan banned), penampilan daftar lowongan magang, mendaftar lowongan, membuat lowongan magang baru serta tampilan detail lowongan magang secara dinamis. Sistem ini juga telah menerapkan pengamanan dasar untuk mencegah kesalahan dan penyalahgunaan data. Dengan demikian, website Campus Intern dapat menjadi media yang membantu mahasiswa dalam memperoleh informasi magang serta memudahkan perusahaan dan admin dalam mengelola data magang secara efektif dan efisien.