

Module Pembekalan & Training Sertifikasi Kompetensi Analis Program BNSP 2024

Created By : Ridho Suhaebi Arrowi

DAFTAR ISI

DAFTAR ISI.....	2
J.620100.002.01/Menganalisis Skalabilitas Perangkat Lunak	4
1. Megumpulkan Kebutuhan Skalabilitas.....	4
2. Menganalisis Kebutuhan Skalabilitas	4
J.620100.020.02/Menggunakan SQL.....	7
1. Mempersiapkan perangkat lunak aplikasi data deskripsi/SQL.....	7
2. Menggunakan fitur aplikasi SQL.....	7
3. Membuat stored procedure	8
4. Membuat function.....	9
5. Membuat trigger	10
6. Melakukan perintah commit dan rollback.....	11
7. Mengisi Table.....	11
J.620100.021.02/Menerapkan Akses Basis Data.....	13
1. Membuat berbagai operasi terhadap basis data.....	13
2. Membuat prosedur akses terhadap basis data.....	14
3. Membuat koneksi basis data.....	14
4. Menguji program basis data.....	15
J.620100.022.02/Mengimplementasikan Algoritma Pemrograman	21
1. Menjelaskan varian dan invarian.....	21
2. Membuat alur logika (Flowchart)	21
3. Menerapkan teknik dasar algoritma umum.....	23
4. Menggunakan prosedur dan fungsi.....	23
5. Mengidentifikasi kompleksitas waktu algoritma.....	24
J.620100.023.02/Membuat Dokumen Kode Program	25
1. Melakukan identifikasi kode program.....	25
2. Membuat dokumentasi modul program.....	27
J.620100.025.02/Melakukan Debugging	31
1. Mempersiapkan kode program.....	31
2. Melakukan debugging.....	31
3. Memperbaiki program.....	32
J.620100.031.01/Melakukan Profiling Program	33
1. Mengumpulkan data waktu eksekusi komponen – komponen yang ada pada program.	33
2. Menentukan bottleneck performa yang ada pada program.....	34
3. Merancang solusi untuk menghilangi/mengurangi bottleneck.....	34

4. Menentukan kompleksitas algoritma	34
J.620100.032.01/Menerapkan Code Review	36
1. Mengevaluasi kesesuaian kode dengan spesifikasi nya.....	36
2. Memperbaiki kode sesuai dengan codingguidlines dan best practices.....	37
3. Membuat pengecualian penulisan kode terhadap coding-guidlines.....	38
J.620100.033.02/Melaksanakan Pengujian Unit Program	38
1. Menentukan kebutuhan uji coba dalam pengembangan.....	38
2. Mempersiapkan dokumentasi uji coba.	40
3. Mempersiapkan data uji.....	41
4. Melaksanakan prosedur uji coba.	42
5. Mengevaluasi hasil uji coba.	46
J.620100.034.02/Melaksanakan Pengujian Integrasi Program	47
1. Mempersiapkan dokumentasi peralatan dan lingkungan pengujian integrasi.....	47
2. Mempersiapkan data uji.....	49
3. Melaksanakan pengujian integrasi.....	50
4. Menganalisis data pengujian integrasi.....	52
5. Melaporkan hasil pengujian integrasi.....	52
6. Melaporkan dokumen pengujian.	53

J.620100.002.01/Menganalisis Skalabilitas Perangkat Lunak

1. Megumpulkan Kebutuhan Skalabilitas

1.1. Mengidentifikasi lingkup (scope) sistem.

Sistem Management Karyawan Berbasis Website (SMKW) ini digunakan untuk mendata seluruh daftar karyawan tetap dan probation di PT. XYZ. Fitur-fitur utama yang termasuk dalam lingkup sistem ini adalah:

- Login (Autentikasi pengguna untuk mengakses sistem)
- Register (Pendaftaran pengguna baru)
- Logout (Keluar dari sistem)
- Forgot Password (Pemulihan kata sandi yang terlupa)
- Profile (Tampilan dan pengelolaan profil pengguna)
- Reset Password (Reset kata sandi pengguna)
- Email Verification (Verifikasi alamat email pengguna)
- Dashboard (Tampilan utama setelah login)
- Management Karyawan (Modul untuk mengelola data karyawan, termasuk penambahan, pengeditan, dan penghapusan data karyawan)

1.2. Mengidentifikasi lingkungan operasi aplikasi.

- Hardware
 - Laptop / PC
- Software
 - Web Browser
 - XAMPP
 - Laravel Framework
 - Visual Studio Code
 - Visual Paradigm
 - Sistem Operasi (Windows)

2. Menganalisis Kebutuhan Skalabilitas

2.1. Menganalisis masalah skalabilitas berdasarkan lingkup dan lingkungan operasi sistem.

PT. XYZ menghadapi tantangan besar dalam operasionalnya, terutama dalam hal pencatatan data yang masih dilakukan secara manual. Metode manual ini rentan terhadap kesalahan manusia (Human Error) yang dapat berdampak signifikan pada akurasi dan efisiensi data karyawan. Kesalahan ini dapat mengakibatkan berbagai masalah, seperti:

- **Ketidakakuratan Data:** Data yang dicatat secara manual rentan terhadap kesalahan penulisan, kehilangan data, dan inkonsistensi.
- **Efisiensi Rendah:** Proses manual memakan waktu dan tenaga lebih banyak, sehingga mengurangi produktivitas.
- **Kesulitan dalam Akses Data:** Data manual sulit diakses dan dicari kembali, terutama ketika volume data meningkat.
- **Keamanan Data:** Data manual lebih rentan terhadap kerusakan fisik dan kehilangan

Untuk mengatasi masalah ini, PT. XYZ berkomitmen untuk memulai proses digitalisasi dengan mengimplementasikan sistem pencatatan data karyawan berbasis website. Sistem ini diharapkan dapat:

- **Mengurangi Human Error:** Otomatisasi proses pencatatan data dapat mengurangi kesalahan yang disebabkan oleh faktor manusia.
- **Meningkatkan Efisiensi:** Proses digital lebih cepat dan efisien dibandingkan dengan metode manual.
- **Mempermudah Akses Data:** Data digital mudah diakses, diolah, dan dicari kembali.
- **Meningkatkan Keamanan Data:** Data digital dapat dilindungi dengan enkripsi dan backup rutin.

2.2. Menganalisis kompleksitas aplikasi sesuai dengan kebutuhan pemrosesan dan jumlah data/pengguna yang akan terlibat.

Membangun sistem pencatatan data karyawan berbasis website untuk PT. XYZ melibatkan berbagai aspek kompleksitas, yang dapat dianalisis berdasarkan beberapa faktor kunci:

- Volume Data
- Kebutuhan Pemrosesan:
- Keamanan dan Privasi:
- Integrasi Sistem:
- Pengembangan dan Pemeliharaan:

2.3. Menganalisis kebutuhan perangkat keras.

- Laptop / Pc
- 11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz 3.11 GHz
- 8.00 GB RAM
- 64-bit
- 512 GB SSD

2.4. Mendokumentasikan Hasil Analisis



System > About

RIDHO
Vivobook_ASUSLaptop X3400PA_K3400PA

Device specifications

Device name	RIDHO
Processor	11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz 3.11 GHz
Installed RAM	8.00 GB (7.69 GB usable)
Device ID	0A3394A5-0A98-4667-B9E2-6DB89C6C31997
Product ID	00342-20765-06406-AA0EM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Related links Domain or workgroup System protection Advanced system settings

Windows specifications

Edition	Windows 11 Home
Version	23H2
Installed on	10/15/2022
OS build	22631.3737
Experience	Windows Feature Experience Pack 1000.22700.1009.0
Microsoft Services Agreement	
Microsoft Software License Terms	

phpMyAdmin

Server: database:3306 > Database: db-karyawan

Tabel

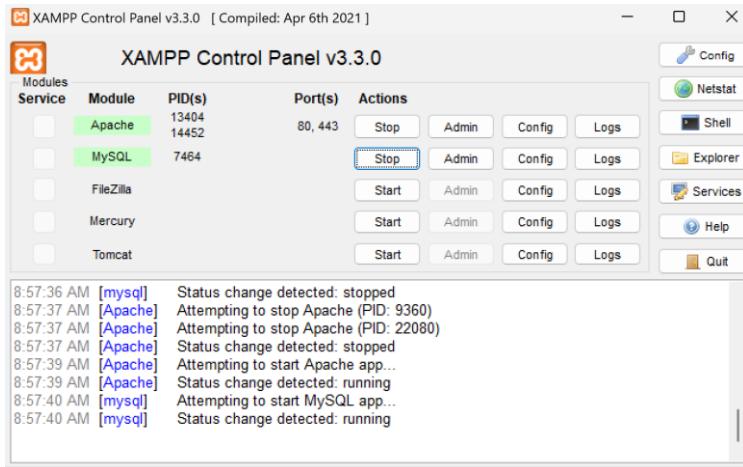
Tabel	Tindakan	Baris	Jenis	Penyortiran	Ukuran	Beban
failed_jobs	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	32,0 KB	-
karyawans	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	1	InnoDB	utf8mb4_unicode_ci	16,0 KB	-
migrations	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	8	InnoDB	utf8mb4_unicode_ci	16,0 KB	-
password_resets	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	32,0 KB	-
password_reset_tokens	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	16,0 KB	-
personal_access_tokens	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	48,0 KB	-
telescope_entries	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	457	InnoDB	utf8mb4_unicode_ci	560,0 KB	-
telescope_entries_tags	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	366	InnoDB	utf8mb4_unicode_ci	112,0 KB	-
telescope_monitoring	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	16,0 KB	-
users	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	2	InnoDB	utf8mb4_unicode_ci	32,0 KB	-

J.620100.020.02/Menggunakan SQL

1. Mempersiapkan perangkat lunak aplikasi data deskripsi/SQL

1.1. Memasang perangkat lunak aplikasi SQL

1.2. Menjalankan perangkat lunak aplikasi SQL



2. Menggunakan fitur aplikasi SQL.

2.1. Mengidentifikasi fitur pengolahan DDL.

```
1 CREATE TABLE karyawan (
2     id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
3     nama VARCHAR(255),
4     jabatan VARCHAR(255),
5     gaji INT,
6     created_at TIMESTAMP NULL DEFAULT NULL,
7     updated_at TIMESTAMP NULL DEFAULT NULL
8 );
9
```

2.2. Mengeksekusi fitur pengolahan DML sesuai kebutuhan.

```

1 -- Membuat / menambahkan karyawan baru
2 INSERT INTO karyawans (nama, jabatan, gaji, created_at, updated_at)
3 VALUES ('Budi Santoso', 'Manager', 12000000, NOW(), NOW());
4
5 INSERT INTO karyawans (nama, jabatan, gaji, created_at, updated_at)
6 VALUES ('Siti Aminah', 'Staff', 8000000, NOW(), NOW());
7
8 -- Memperbarui gaji karyawan dengan id = 1
9 UPDATE karyawans
10 SET gaji = 13000000, updated_at = NOW()
11 WHERE id = 1;
12
13 -- Memperbarui jabatan karyawan dengan nama 'Siti Aminah'
14 UPDATE karyawans
15 SET jabatan = 'Senior Staff', updated_at = NOW()
16 WHERE nama = 'Siti Aminah';
17
18 -- Menghapus karyawan dengan id = 2
19 DELETE FROM karyawans
20 WHERE id = 2;
21
22 -- Menghapus karyawan yang bergaji kurang dari 9000000
23 DELETE FROM karyawans
24 WHERE gaji < 9000000;
25
26 -- Mengambil semua data karyawan
27 SELECT * FROM karyawans;
28
29 -- Mengambil data karyawan dengan jabatan 'Manager'
30 SELECT * FROM karyawans
31 WHERE jabatan = 'Manager';

```

3. Membuat stored procedure

3.1. Membuat stored procedure dengan perintah SQL.

```

1 DELIMITER $$ 
2
3 CREATE PROCEDURE AddKaryawan (
4     IN p_nama VARCHAR(255),
5     IN p_jabatan VARCHAR(255),
6     IN p_gaji INT
7 )
8 BEGIN
9     INSERT INTO karyawans (nama, jabatan, gaji, created_at, updated_at)
10    VALUES (p_nama, p_jabatan, p_gaji, NOW(), NOW());
11 END $$ 
12
13 DELIMITER ;

```

3.2. Menguji dan memeriksa prosedur input dan outputnya.

- **Input**

```
1 CALL AddKaryawan('Budi Santoso', 'Manager', 12000000);
2
3
4 CALL UpdateKaryawan(1, 'Budi Santoso', 'Senior Manager', 15000000);
5
6
7 CALL DeleteKaryawan(1);
8
9
10 CALL GetKaryawan(1);
11
12
13 CALL GetAllKaryawans();
```

- **Output**

The screenshot shows a list of five rows of SQL code and their execution results:

- Row 1: MySQL memberikan hasil kosong (atau nol baris). (Pencarian dilakukan dalam 0.0134 detik.)
CALL AddKaryawan('Budi Santoso', 'Manager', 12000000);
[Edit dikotak] [Ubah] [Buat kode PHP]
- Row 2: MySQL memberikan hasil kosong (atau nol baris). (Pencarian dilakukan dalam 0.0097 detik.)
CALL UpdateKaryawan(1, 'Budi Santoso', 'Senior Manager', 15000000);
[Edit dikotak] [Ubah] [Buat kode PHP]
- Row 3: MySQL memberikan hasil kosong (atau nol baris). (Pencarian dilakukan dalam 0.0007 detik.)
CALL DeleteKaryawan(1);
[Edit dikotak] [Ubah] [Buat kode PHP]
- Row 4: MySQL memberikan hasil kosong (atau nol baris). (Pencarian dilakukan dalam 0.0033 detik.)
CALL GetKaryawan(1);
[Edit dikotak] [Ubah] [Buat kode PHP]
⚠ Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available. ⚡
- Row 5: Menampilkan baris 0 - 1 (total 2). Pencarian dilakukan dalam 0.0012 detik.
CALL GetAllKaryawans();
[Edit dikotak] [Ubah] [Buat kode PHP]

4. Membuat function

4.1. Membuat function dengan perintah SQL.

```
1 DELIMITER //
2
3 CREATE FUNCTION TotalGajiKaryawan()
4 RETURNS INT
5 DETERMINISTIC
6 BEGIN
7     DECLARE total INT;
8     SELECT SUM(gaji) INTO total FROM karyawans;
9     RETURN total;
10 END //
11
12 DELIMITER ;
13
14
15
```

4.2. Menulis perintah SQL pada function secara efisien

```
1 SELECT TotalGajiKaryawan() AS total_gaji;
2
3 SELECT GetNamaKaryawan(1) AS nama_karyawan;
4
5 SELECT RataRataGajiKaryawan() AS rata_rata_gaji;
6
7 SELECT GetJabatanKaryawan(1) AS jabatan_karyawan;
```

5. Membuat trigger

5.1. Mendefinisikan trigger dengan perintah SQL.

```
1 DELIMITER $$ 
2
3 CREATE TRIGGER before_insert_karyawans
4 BEFORE INSERT ON karyawans
5 FOR EACH ROW
6 BEGIN
7     IF NEW.jabatan IS NULL OR NEW.jabatan = '' THEN
8         SET NEW.jabatan = 'Staff';
9     END IF;
10 END$$
11
12 DELIMITER ;
```

5.2. Menguji kesesuaian hasil trigger

The screenshot shows the phpMyAdmin interface for a database named 'db-karyawan'. The left sidebar lists tables such as 'failed_jobs', 'karyawans', 'log_karyawan', 'log_perubahan_gaji', 'migrations', 'password_resets', 'personal_access_tokens', 'telescope_entries', 'telescope_entries_tags', 'telescope_monitoring', and 'users'. The 'log_karyawan' table is selected. The main area displays the following SQL query:

```
SELECT * FROM `log_karyawan`
```

The results table shows one row:

	id	karyawan_id	action	created_at
	1		INSERT	2024-06-27 03:46:07

Below the table are various operations like 'Ubah', 'Salin', 'Hapus', and 'Ekspor'.

6. Melakukan perintah commit dan rollback

6.1. Melakukan perubahan data dengan perintah commit.

6.2. Melakukan pembatalan penulisan dengan perintah rollback

```
1 -- Mulai transaksi
2 BEGIN;
3
4 -- Lakukan operasi insert
5 INSERT INTO karyawans (nama, jabatan, gaji, created_at, updated_at)
6 VALUES ('John Doe', 'Staff', 5000, NOW(), NOW());
7
8 -- Cek jika insert berhasil
9 IF ROW_COUNT() = 1 THEN
10    -- Lakukan operasi update
11    UPDATE karyawans SET gaji = 8000, updated_at = NOW() WHERE id = 1;
12
13    -- Jika semua operasi berhasil, lakukan commit
14    COMMIT;
15 ELSE
16    -- Jika insert gagal, lakukan rollback
17    ROLLBACK;
18 END IF;
19 |
```

7. Mengisi Table

7.1. Mengisi table data dengan perintah DML

```
1 INSERT INTO karyawans (nama, jabatan, gaji, created_at, updated_at)
2 VALUES ('Budi Santoso', 'Manager', 1200000, NOW(), NOW());
3
4 INSERT INTO karyawans (nama, jabatan, gaji, created_at, updated_at)
5 VALUES ('Siti Aminah', 'Staff', 800000, NOW(), NOW());
6
```

7.2. Membangkitkan indeks.

```
1 -- Hapus indeks yang ada
2 DROP INDEX idx_nama ON karyawans;
3
4 -- Buat ulang indeks
5 CREATE INDEX idx_nama ON karyawans(nama);
```

7.3. Membentuk view tabel sesuai kebutuhan.

```
1 CREATE VIEW vw_karyawan_nama_jabatan AS
2 SELECT nama, jabatan
3 FROM karyawans;
4
5 SHOW CREATE VIEW vw_karyawan_nama_jabatan;
```

J.620100.021.02/Menerapkan Akses Basis Data

1. Membuat berbagai operasi terhadap basis data.

1.1. Menyimpan / mengubah data kedalam format basis data.

```
1 -- Membuat / menambahkan karyawan baru
2 INSERT INTO karyawans (nama, jabatan, gaji, created_at, updated_at)
3 VALUES ('Budi Santoso', 'Manager', 12000000, NOW(), NOW());
4
5 -- Membuat / menambahkan karyawan baru
6 INSERT INTO karyawans (nama, jabatan, gaji, created_at, updated_at)
7 VALUES ('Siti Aminah', 'Staff', 8000000, NOW(), NOW());
8
9 -- Memperbarui gaji karyawan dengan id = 1
10 UPDATE karyawans
11 SET gaji = 13000000, updated_at = NOW()
12 WHERE id = 1;
13
14 -- Memperbarui jabatan karyawan dengan nama 'Siti Aminah'
15 UPDATE karyawans
16 SET jabatan = 'Senior Staff', updated_at = NOW()
17 WHERE nama = 'Siti Aminah';
18
19
```

1.2. Menghasilkan query untuk informasi yang di inginkan

```
1 -- Mengambil semua data karyawan
2 SELECT * FROM karyawans;
3
4 -- Mengambil data karyawan dengan jabatan 'Manager'
5 SELECT * FROM karyawans
6 WHERE jabatan = 'Manager';
7
8 -- Mengambil nama dan gaji semua karyawan yang bergaji lebih dari 10 juta
9 SELECT nama, gaji FROM karyawans
10 WHERE gaji > 10000000;
11 |
```

1.3. Mempercepat akses menggunakan indeks

```

1 -- Membuat index pada kolom 'nama' di tabel 'karyawan'
2 CREATE INDEX idx_nama ON karyawan(nama);
3
4 -- Melakukan SELECT untuk menguji apakah indeks berfungsi dengan baik
5 SELECT * FROM karyawan WHERE nama = 'John Doe';
6
7 -- Menampilkan informasi mengenai index yang ada pada tabel 'karyawan'
8 SHOW INDEX FROM karyawan;
9

```

2. Membuat prosedur akses terhadap basis data.

2.1. Menerapkan library akses basis data

```

You, 2 days ago | 1 author (You)
<?php

// Menerapkan library akses basis data - Eloquent ORM (Menerapkan Akses Basis Data di Laravel)
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Karyawan extends Model
{
    use HasFactory;
    protected $fillable = [
        'nama', 'jabatan', 'gaji', '_token',
    ];
}

```

2.2. Menerapkan perintah akses data yang relevan dengan teknologi atau jenis baru data untuk mengakses data.

```

31
32     // mengambil data karyawan (Menerapkan Akses Basis Data di Laravel)
33     $query = Karyawan::query();
34
35     // Algoritma searching karyawan
36     if ($search) {
37         $query->where('nama', 'LIKE', "%{$search}%")
38             ->orWhere('jabatan', 'LIKE', "%{$search}%");
39     }
40

```

OUTPUT PROBLEMS DEBUG CONSOLE TEST RESULTS ROBOT OUTPUT PORTS DEVDB TERMINAL

3. Membuat koneksi basis data.

3.1. Memilih teknologi koneksi yang sesuai.

3.2. Menentukan keamanan koneksi

```

11  # Memilih Teknologi Koneksi yang Sesuai
12  DB_CONNECTION=mysql
13  DB_HOST=127.0.0.1
14  DB_PORT=3306
15  DB_DATABASE=db-karyawan
16  DB_USERNAME=Example User
17  # Menentukan Keamanan Koneksi
18  DB_PASSWORD=Example-Password
19  # Enabled Telescope
20  TELESCOPE_ENABLED=true
21
22  BROADCAST_DRIVER=log
23  CACHE_DRIVER=file
24  FILESYSTEM_DISK=local
25  QUEUE_CONNECTION=sync
26  SESSION_DRIVER=file
27  SESSION_LIFETIME=120
28
29  MEMCACHED_HOST=127.0.0.1
30

```

OUTPUT PROBLEMS DEBUG CONSOLE TEST RESULTS ROBOT OUTPUT PORTS DEVDB TERMINAL

3.3. Menentukan has setiap pengguna.

```

You, 2 days ago | 1 author (You)
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 return new class extends Migration
8 {
9
10     /**
11      * Run the migrations.
12      */
13     public function up()
14     {
15         Schema::table('users', function (Blueprint $table) {
16             $table->string('role')->default('user'); // Default role: 'user'
17         });
18
19         /**
20          * Reverse the migrations.
21          */
22         public function down(): void
23         {
24             Schema::table('users', function (Blueprint $table) {
25                 //
26             });
27         }
28     };
29

```

OUTPUT PROBLEMS DEBUG CONSOLE TEST RESULTS ROBOT OUTPUT PORTS DEVDB TERMINAL

	id	name	email	email_verified_at	password	remember_token	created_at	updated_at	role	
Salin	Hapus	1	Admin	admin@gmail.com	NULL	\$2y\$12\$N4IXNzmkC4rtfq2ME/fHugGWM65MYXtII3oAGm6R...	NULL	2024-06-27 04:14:43	2024-06-27 04:14:43	admin
Salin	Hapus	2	user	user@gmail.com	NULL	\$2y\$12\$apJhoUEJIE1aAXUbHUknezRxoU43jAHvoNP7ZDrAs...	NULL	2024-06-28 04:03:55	2024-06-28 04:03:55	user

Jumlah baris: 2 Pencarian dilakukan dalam 0.0000 detik.

Dikotak] [Ubah] [Jelaskan SQL] [Buat kode PHP] [Segarkan]

can semua | Jumlah baris: 25 | Saring baris: Cari di tabel ini | Sort by key: Tidak ada

Lihat Semua Dengan pilihan: [Ubah](#) [Salin](#) [Hapus](#) [Eksport](#)

can semua | Jumlah baris: 25 | Saring baris: Cari di tabel ini | Sort by key: Tidak ada

isi kueri

[Salin ke clipboard](#) [Eksport](#) [Tampilkan bagan](#) [Buat tampilan](#)

4. Menguji program basis data.

4.1. Menyiapkan scenario pengujian.

- Create

Nama
test

Jabatan
test

Gaji
1000000

SUBMIT

- **Read**

Cari Karyawan

Nama	Jabatan	Gaji	Aksi
test	test	1000000	<button>Edit</button> <button>Delete</button>

- **Update**

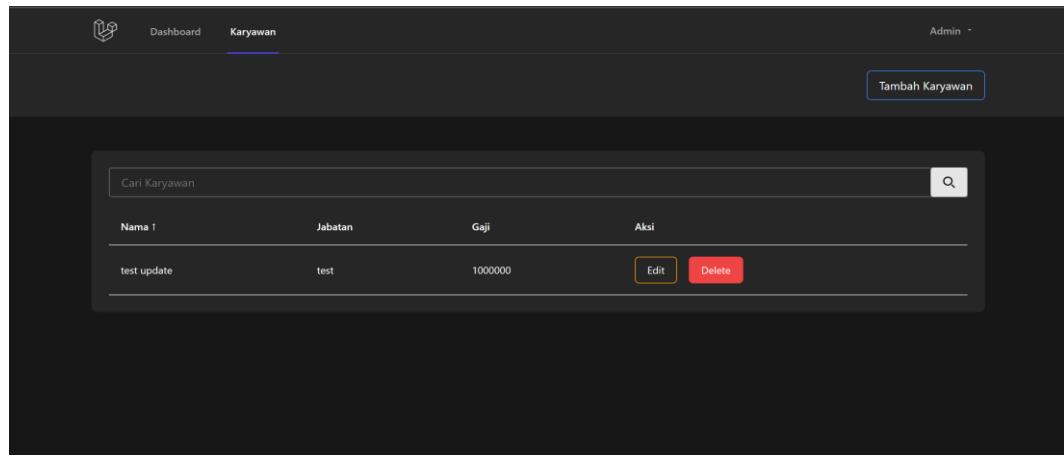
Nama
test Update

Jabatan
test

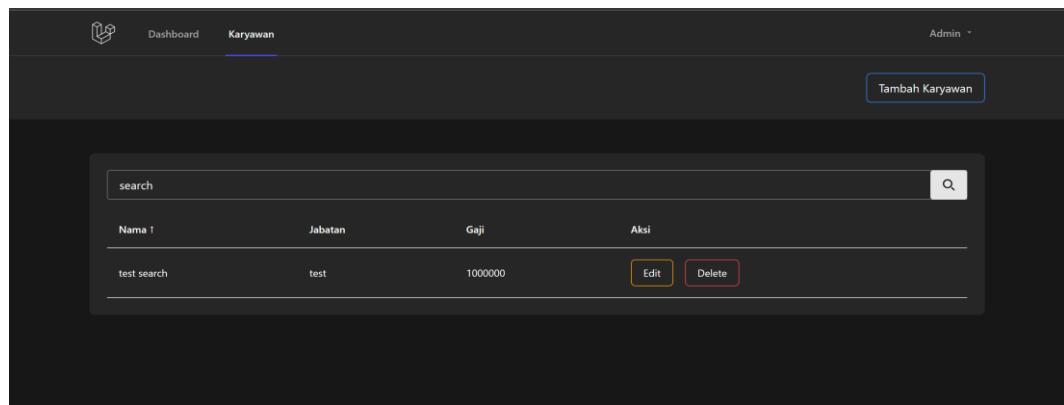
Gaji
1000000

SUBMIT

- **Delete**



- **Search**



4.2. Membaca logika pemrograman mengacu pada kinerja statement akses data.

```

9   // Membaca Logika Pemrograman Mengacu pada Kinerja Statement Akses (Operasi CRUD)
10
11 2 references | 0 implementations | You, 2 days ago | 1 author (You)
12 class KaryawanController extends Controller
13 {
14
15     // Menentukan metode yang sesuai
16     // 0 references | 0 overrides
17     public function index(Request $request)
18     {
19         $search = $request->query('search');
20
21         // Contoh field yang sesuai (Melakukan perbaikan).
22         // $sortField = $request->query('sortField', 'nama'); // Default sorting field
23         // You, 2 days ago * first commit
24         // $sortOrder = $request->query('sortOrder', 'asc'); // Default sorting order
25
26         // mengambil data karyawan (Menerapkan Akses Basis Data di Laravel)
27         $query = Karyawan::query();
28
29         // Algoritma searching karyawan
30         if ($search) {
31             $query->where('nama', 'LIKE', "%{$search}%")
32             ->orWhere('jabatan', 'LIKE', "%{$search}%");
33         }
34
35         // Algoritma sorting pada query
36         $karyawans = $query->orderBy($sortField, $sortOrder)->paginate(10);
37
38         // var_dump($karyawans); // Menggunakan var_dump untuk debugging
39
40         return view('karyawans.index', compact('karyawans', 'sortField', 'sortOrder'));
41     }

```

```

42     0 references | 0 overrides
43     public function create()
44     {
45         return view('karyawans.create');
46     }
47
48     // Menentukan metode yang sesuai
49     0 references | 0 overrides
50     public function store(Request $request)
51     {
52         // Validasi input
53         $request->validate([
54             // Menjelaskan tipe data sesuai kaidah pemrograman
55             'nama' => 'required|string|max:255',
56             'jabatan' => 'required|string|max:255',
57             'gaji' => 'required|integer|min:0',
58         ]);
59
60         // Menjelaskan variable sesuai kaidah pemrograman
61         $data = $request->except('_token');
62
63         // Logika tambahan sebelum penyimpanan
64         if (strlen($data['nama']) < 3) {
65             return redirect()->back()->withErrors(['nama' => 'Nama terlalu pendek.']);
66         }
67
68         // Menyimpan Data ke Database (Menerapkan Akses Basis Data di Laravel)
69         Karyawan::create($data);
70
71         return redirect()->route('karyawans.index')->with('success', 'Karyawan created successfully.');
72     }
73
74
75     // Menentukan metode yang sesuai
76     0 references | 0 overrides
77     public function edit(Karyawan $karyawan)
78     {
79         return view('karyawans.edit', compact('karyawan'));
80     }
81
82
83     // Menentukan metode yang sesuai
84     0 references | 0 overrides
85     public function update(Request $request, Karyawan $karyawan)
86     {
87         // Validasi input
88         $request->validate([
89             'nama' => 'required|string|max:255',
90             'jabatan' => 'required|string|max:255',
91             'gaji' => 'required|integer|min:0',
92         ]);
93
94         // Menjelaskan variable sesuai kaidah pemrograman
95         $data = $request->except('_token');
96
97         // Logika tambahan sebelum penyimpanan
98         if (strlen($data['nama']) < 3) {
99             return redirect()->back()->withErrors(['nama' => 'Nama terlalu pendek.']);
100        }
101
102        // Mengupdate Data ke Database (Menerapkan Akses Basis Data di Laravel)
103        $karyawan->update($data);
104
105        return redirect()->route('karyawans.index')->with('success', 'Karyawan updated successfully.');
106    }
107
108
109     // Menentukan metode yang sesuai
110     0 references | 0 overrides
111     public function destroy(Karyawan $karyawan)
112     {
113         // Menjelaskan variable sesuai kaidah pemrograman
114         $karyawan->delete();
115
116         return redirect()->route('karyawans.index')->with('success', 'Karyawan deleted successfully.');
117     }

```

4.3. Membaca data uji perfomansi mengacu pada kinerja statement akses data

Controller Action	App\Http\Controllers\KaryawanController@index
Middleware	web, auth, verified, admin
Path	/karyawans
Status	200
Duration	240 ms
IP Address	127.0.0.1
Memory usage	24 MB
Tags	Auth:1
Method	POST
Controller Action	App\Http\Controllers\KaryawanController@store
Middleware	web, auth, verified, admin
Path	/karyawans
Status	302
Duration	325 ms
IP Address	127.0.0.1
Memory usage	24 MB
Tags	Auth:1
Method	PUT
Controller Action	App\Http\Controllers\KaryawanController@update
Middleware	web, auth, verified, admin
Path	/karyawans/3
Status	302
Duration	328 ms
IP Address	127.0.0.1
Memory usage	24 MB
Tags	Auth:1

Method	DELETE
Controller Action	App\Http\Controllers\KaryawanController@destroy
Middleware	web, auth, verified, admin
Path	/karyawans/1
Status	302
Duration	219 ms
IP Address	127.0.0.1
Memory usage	24 MB
Tags	Auth:1

Controller Action	App\Http\Controllers\KaryawanController@index
Middleware	web, auth, verified, admin
Path	/karyawans?search=search
Status	200
Duration	258 ms
IP Address	127.0.0.1
Memory usage	24 MB
Tags	Auth:1

J.620100.022.02/Mengimplementasikan Algoritma Pemrograman

1. Menjelaskan varian dan invarian.

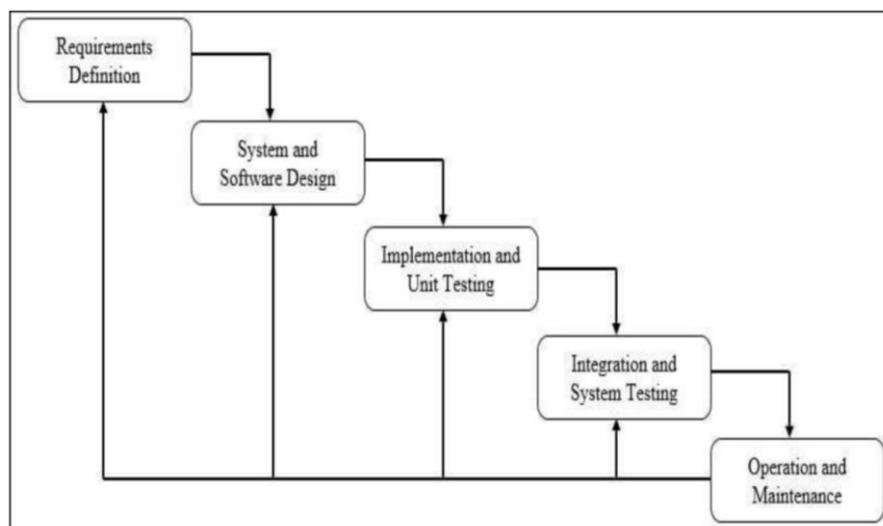
1.1. Menjelaskan tipe data sesuai kaidah pemrograman.

- **Char** : tipe data yang berisi 1 huruf atau 1 karakter
- **Integer** : tipe data yang berupa angka
- **Float** : tipe data yang berupa bilangan pecahan
- **Bollean** : tipe data untuk menentukan true or false

1.2. Menjelaskan variabel sesuai kaidah pemrograman

- Nama variabel tidak boleh di dahului dengan symbol dan angka
- Nama variabel tidak boleh menggunakan kata kunci yang sudah ada pada bahasa
- Nama variabel bersifat case sensitive
- Disarankan menggunakan underscore untuk nama variabel lebih dari 2 suku kata.

1.3. Menentukan metode yang sesuai



2. Membuat alur logika (Flowchart)

2.1. Metode sesuai di tentukan.

2.2. Komponen yang di butuhkan di tentukan.

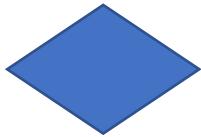
- **Terminator** : Simbol yang menyatakan awal atau akhir suatu program.



- **Process** : Simbol yang menyatakan suatu proses yang di lakukan komputer.



- **Decision :** Simbol yang menyatakan kondisi tertentu yang akan menghasilkan dua kemungkinan jawaban yaitu ya dan tidak.



- **Input/Output :** Simbol yang menyatakan proses input atau output tanpa peralatan.



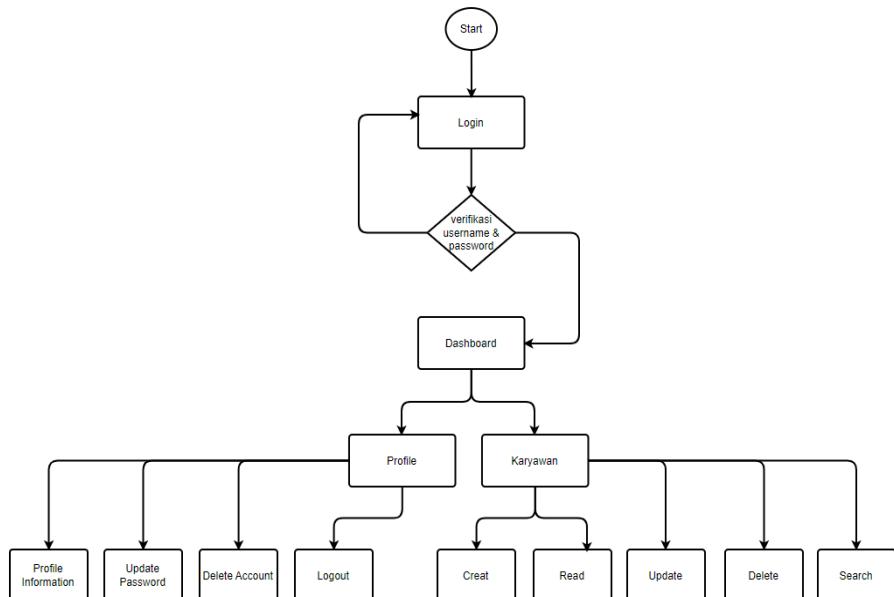
- **Flow :** Simbol yang digunakan untuk menggabungkan antara symbol yang satu dengan symbol lain.



- **Off-page Reference :** Simbol yang digunakan untuk keluar atau masuk ke penyambungan proses dalam lembar kerja yang berbeda.



2.3. Relasi antar komponen di tetapkan.



2.4. Alur mulai dan selesai ditetapkan.

- Alur mulai pada halaman login dan berakhir setelah logout.

3. Menerapkan teknik dasar algoritma umum.

3.1. Membuat algoritma sorting.

```
// Contoh field yang sesuai (Melakukan perbaikan).
$sortField = $request->query('sortField', 'nama'); // Default sorting field

$sortOrder = $request->query('sortOrder', 'asc'); // Default sorting order

// Algoritma sorting pada query
$karyawans = $query->orderBy($sortField, $sortOrder)->paginate(10);

// var_dump($karyawans); // Menggunakan var_dump untuk debugging

return view('karyawans.index', compact('karyawans', 'sortField', 'sortOrder'));
```

3.2. Membuat algoritma searching

```
$search = $request->query('search');

$query = Karyawan::query();

// Algoritma searching karyawan
if ($search) {
    $query->where('nama', 'LIKE', "%{$search}%")
        ->orWhere('jabatan', 'LIKE', "%{$search}%");
}
```

4. Menggunakan prosedur dan fungsi.

4.1. Menggunakan prosedur.

```
namespace App\Procedures;

use App\Models\Karyawan;

2 references | 0 implementations
class KaryawanProcedures
{
    // Prosedur: Menambahkan Karyawan
    1 reference | 0 overrides
    public function addKaryawan($data)
    {
        return Karyawan::create($data);
    }

    // Prosedur: Mengambil Semua Karyawan
    1 reference | 0 overrides
    public function getAllKaryawans()
    {
        return Karyawan::all();
    }

    // Prosedur: Mengupdate Karyawan
    1 reference | 0 overrides
    public function updateKaryawan($karyawan, $data)
    {
        return $karyawan->update($data);
    }

    // Prosedur: Menghapus Karyawan
    1 reference | 0 overrides
    public function deleteKaryawan($karyawan)
    {
        return $karyawan->delete();
    }
}
```

4.2. Menggunakan fungsi

```
3 namespace App\Functions;
4
5 use App\Models\Karyawan;
6
7 class KaryawanFunctions
8 {
9     // Fungsi: Mengecek Apakah Karyawan dengan Nama Tertentu Ada di Database
10    0 references | 0 overrides
11    public function checkKaryawanExists($nama)
12    {
13        return Karyawan::where('nama', $nama)->exists();
14    }
15
16    // Fungsi: Mengambil Karyawan Berdasarkan ID
17    0 references | 0 overrides
18    public function getKaryawanById($id)
19    {
20        return Karyawan::find($id);
21    }
}
```

5. Mengidentifikasi kompleksitas waktu algoritma.

5.1. Mengidentifikasi kompleksitas waktu algoritma.

Views (15)	Queries (4)	Models (2)
Query		Duration
4 queries, 0 of which are duplicated.		30.99ms
<code>select * from 'karyawans' where 'nama' LIKE '%search%' or 'jabatan' LIKE '%search%' order by 'nama' asc...</code>		2.42ms
<code>select count(*) as aggregate from 'karyawans' where 'nama' LIKE '%search%' or 'jabatan' LIKE '%search%'</code>		8.95ms
<code>select * from 'karyawans'</code>		4.99ms
<code>select * from 'users' where 'id' = 1 limit 1</code>		14.63ms

5.2. Mengidentifikasi kompleksitas penggunaan memory algoritma.

Request Details	
Time	June 28th 2024, 3:52:44 PM (3:33m ago)
Hostname	RIDHO
Method	GET
Controller Action	App\Http\Controllers\KaryawanController@index
Middleware	web, auth, verified, admin
Path	/karyawans?search=search
Status	200
Duration	367 ms
IP Address	127.0.0.1
Memory usage	24 MB
Tags	Auth:1

J.620100.023.02/Membuat Dokumen Kode Perogram

1. Melakukan identifikasi kode program.

1.1. Mengidentifikasi modul program.

- Login (Autentikasi pengguna untuk mengakses sistem)
- Register (Pendaftaran pengguna baru)
- Logout (Keluar dari sistem)
- Forgot Password (Pemulihan kata sandi yang terlupa)
- Profile (Tampilan dan pengelolaan profil pengguna)
- Reset Password (Reset kata sandi pengguna)
- Dashboard (Tampilan utama setelah login)
- Management Karyawan (Modul untuk mengelola data karyawan, termasuk penambahan, pengeditan, dan penghapusan data karyawan)

1.2. Mengidentifikasi parameter yang di gunakan

Requests		
Verb	Path	Status
GET	/karyawans?search=	200
GET	/karyawans?search=search	200
GET	/karyawans	200
POST	/karyawans	302
GET	/karyawans/create	200
GET	/karyawans	200
DELETE	/karyawans/2	302
GET	/karyawans	200
PUT	/karyawans/3	302

1.3. Menjelaskan cara kerja algoritma.

1.4. Memberikan komentar setiap baris kode termasuk data, eksepsi, fungsi, prosedur, dan class (bila ada)

```
public function index(Request $request)
{
    $search = $request->query('search');

    // Contoh field yang sesuai (Melakukan perbaikan).
    $sortField = $request->query('sortField', 'nama'); // Default sorting field

    $sortOrder = $request->query('sortOrder', 'asc'); // Default sorting order

    // mengambil data karyawan (Menerapkan Akses Basis Data di Laravel)
    $query = Karyawan::query();

    // Algoritma searching karyawan
    if ($search) {
        $query->where('nama', 'LIKE', "%{$search}%")
            ->orWhere('jabatan', 'LIKE', "%{$search}%");
    }

    // Menggunakan prosedur untuk mendapatkan semua karyawan
    $karyawans = $this->procedures->getAllKaryawans();

    // Algoritma sorting pada query
    $karyawans = $query->orderBy($sortField, $sortOrder)->paginate(10);

    // var_dump($karyawans); // Menggunakan var_dump untuk debugging

    return view('karyawans.index', compact('karyawans', 'sortField', 'sortOrder'));
}
```

```
1 <?php
2
3 namespace App\Functions;
4
5 use App\Models\Karyawan;
6
7
8 class KaryawanFunctions
9 {
10     // Fungsi: Mengecek Apakah Karyawan dengan Nama Tertentu Ada di Database
11     // 0 references | 0 overrides
12     public function checkKaryawanExists($nama)
13     {
14         return Karyawan::where('nama', $nama)->exists();
15     }
16
17     // Fungsi: Mengambil Karyawan Berdasarkan ID
18     // 0 references | 0 overrides
19     public function getKaryawanById($id)
20     {
21         return Karyawan::find($id);
22     }
23 }
```

```
1 <?php
2
3 namespace App\Procedures;
4
5 use App\Models\Karyawan;
6
7
8 class KaryawanProcedures
9 {
10     // Prosedur: Menambahkan Karyawan
11     // 1 reference | 0 overrides
12     public function addKaryawan($data)
13     {
14         return Karyawan::create($data);
15     }
16
17     // Prosedur: Mengambil Semua Karyawan
18     // 1 reference | 0 overrides
19     public function getAllKaryawans()
20     {
21         return Karyawan::all();
22     }
23
24     // Prosedur: Mengupdate Karyawan
25     // 1 reference | 0 overrides
26     public function updateKaryawan($karyawan, $data)
27     {
28         return $karyawan->update($data);
29     }
30
31     // Prosedur: Menghapus Karyawan
32     // 1 reference | 0 overrides
33     public function deleteKaryawan($karyawan)
34     {
35         return $karyawan->delete();
36     }
37 }
```

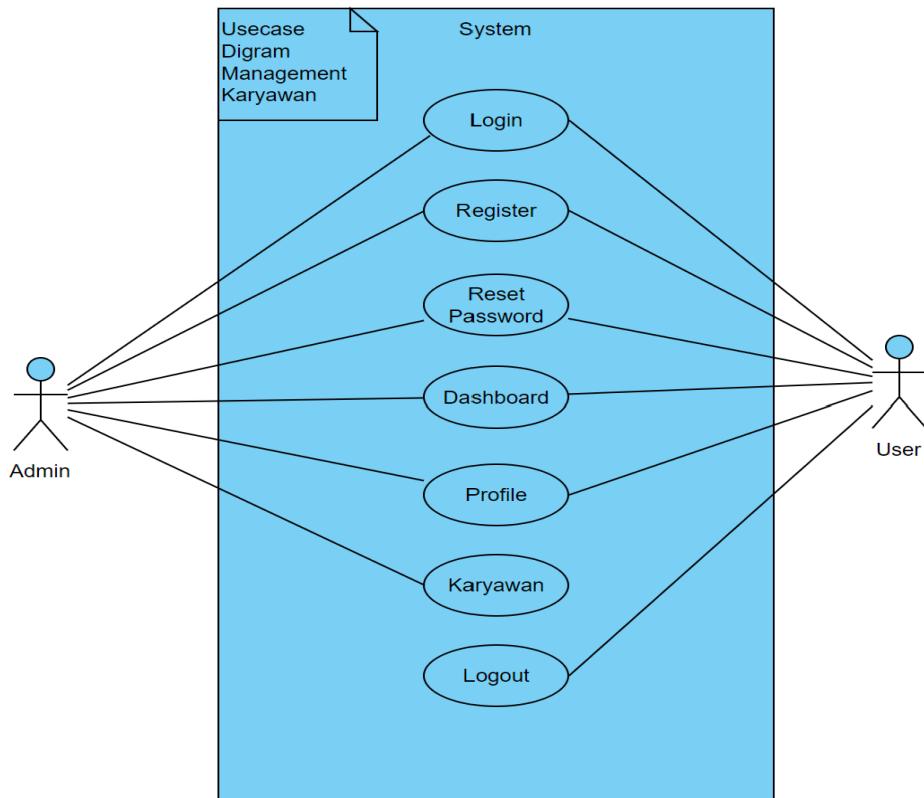
2. Membuat dokumentasi modul program.

2.1. Membuat dokumentasi modul sesuai dengan identitas untuk memudahkan pelacakan.

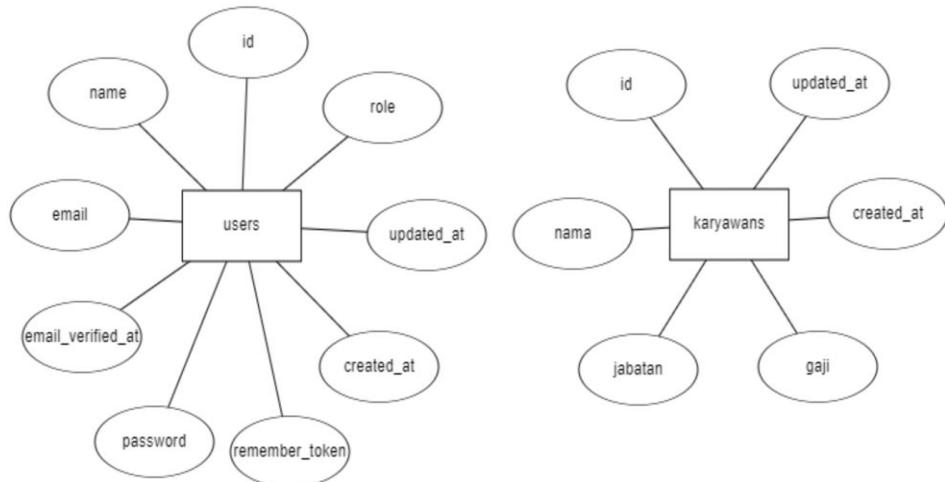
2.2. Terapkan identifikasi dokumen sesuai dengan identitas untuk memudahkan pelacakan

2.3. Jelaskan kegunaan modul.

➤ Usecase Diagram :



➤ Er diagram :



Dokumentasi Modul :

- **Login**

- Kegunaan: Modul ini digunakan untuk mengotentikasi pengguna yang ingin mengakses sistem. Pengguna memasukkan kredensial mereka (email dan password) dan sistem memverifikasi keabsahan informasi tersebut.
- Rute: Route::get('login', [AuthenticatedSessionController::class, 'create']) ->name('login');
- Controller: LoginController
- View: resources/views/auth/login.blade.php

- **Register**

- Kegunaan: Modul ini digunakan untuk pendaftaran pengguna baru. Pengguna mengisi formulir pendaftaran dengan informasi yang diperlukan, seperti nama, email, dan password.
- Rute: Route::post('register', [RegisteredUserController::class, 'store']);
- Controller: RegisterController
- View: resources/views/auth/register.blade.php

- **Logout**

- Kegunaan: Modul ini digunakan untuk keluar dari sistem. Pengguna yang sudah login dapat keluar dengan aman dari akun mereka.
- Rute: Route::post('logout', [AuthenticatedSessionController::class, 'destroy']) ->name('logout');
- Controller: AuthenticatedSessionController

- **Profile**

- Kegunaan: Modul ini digunakan untuk tampilan dan pengelolaan profil pengguna. Pengguna dapat melihat, mengedit informasi profil mereka & menghapus account mereka.
- Rute:

- Route::get('/profile', [ProfileController::class, 'edit'])->>name('profile.edit');
- Route::patch('/profile', [ProfileController::class, 'update'])->>name('profile.update');
- Route::delete('/profile', [ProfileController::class, 'destroy'])->>name('profile.destroy');
- Controller: ProfileController
- View:
 - resources/views/profile/edit.blade.php
 - resources/views/profile/partials/update-password.blade.php
 - resources/views/profile/partials/update-profile.blade.php
 - resources/views/profile/partials/delete-user.blade.php

- o **Reset Password**

- Kegunaan: Modul ini digunakan untuk reset kata sandi pengguna. Setelah pengguna meminta reset kata sandi, mereka dapat mengatur ulang kata sandi mereka melalui tautan yang dikirim ke email mereka.
- Rute:
 - Route::get('reset-password/{token}', [NewPasswordController::class, 'create'])->>name('password.reset');
 - Route::post('reset-password', [NewPasswordController::class, 'store'])->>name('password.store');
- Controller: NewPasswordController
- View: resources/views/auth/passwords/reset-password.blade.php

- o **Dashboard**

- Kegunaan: Modul ini digunakan untuk tampilan utama setelah login. Pengguna dapat melihat ringkasan informasi dan navigasi ke berbagai bagian aplikasi.
- Rute: Route::get('/dashboard', function () {

- return view('dashboard');

- })->name('dashboard');
- Controller: Tidak ada (closure)
- View: resources/views/dashboard.blade.php

- **Management Karyawan**

- Kegunaan: Modul ini digunakan untuk mengelola data karyawan, termasuk penambahan, pengeditan, penghapusan dan pencarian data karyawan.
- Rute: Route::middleware('admin')->resource('/karyawans', KaryawanController::class);
- Controller: KaryawanController
- View:
 - Index: resources/views/karyawans/index.blade.php
 - Create: resources/views/karyawans/create.blade.php
 - Edit: resources/views/karyawans/edit.blade.php

2.4. Merevisi dokumen sesuai perubahan kode program.

- Penambahan prosedur & fungsi baru pada KaryawanController.
- Perubahan pada model Karyawan.
- Modifikasi rute yang digunakan dalam web.php.
- Pembaruan tampilan (view) yang digunakan dalam modul Karyawan.

J.620100.025.02/Melakukan Debugging

1. Mempersiapkan kode program.

1.1. Menyiapkan kode program sesuai spesifikasi.

```
// Menentukan metode yang sesuai
0 references | 0 overrides
public function index(Request $request)
{
    $search = $request->query('search');

    // Contoh field yang sesuai (Melakukan perbaikan).
    $sortField = $request->query('sortField', 'nama'); // Default sorting field

    $sortOrder = $request->query('sortOrder', 'asc'); // Default sorting order

    // mengambil data karyawan (Menerapkan Akses Basis Data di Laravel)
    $query = Karyawan::query();

    // Algoritma searching karyawan
    if ($search) {
        $query->where('nama', 'LIKE', "%{$search}%")
            ->orWhere('jabatan', 'LIKE', "%{$search}%");
    }

    // Menggunakan prosedur untuk mendapatkan semua karyawan
    $karyawans = $this->procedures->getAllKaryawans();

    // Algoritma sorting pada query
    $karyawans = $query->orderBy($sortField, $sortOrder)->paginate(10);| You, 2

    return view('karyawans.index', compact('karyawans', 'sortField', 'sortOrder'));
}
```

1.2. Menyiapkan debugging tools untuk melihat proses suatu modul.

```
{
    "name": "laravel/laravel",
    "type": "project",
    "description": "The skeleton application for the Laravel framework.",
    "keywords": ["laravel", "framework"],
    "license": "MIT",
    "require": {
        "php": "^8.1",
        "guzzlehttp/guzzle": "^7.2" (v7.8.1),
        "laravel/framework": "^10.10" (v10.48.14),
        "laravel/sanctum": "^3.3" (v3.3.3),
        "laravel/telescope": "^5.1" (v5.1.0),
        "laravel/tinker": "^2.8" (v2.9.0)
    },
    "require-dev": {
        "fakerphp/faker": "^1.9.1" (v1.23.1),
        "laravel/breeze": "^1.29" (v1.29.1),
        "laravel/pint": "^1.0" (v1.16.1),
        "laravel/sail": "^1.18" (v1.29.3),
        "mockery/mockery": "1.4.4" (1.6.12),
        "nunomaduro/collision": "7.0" (v7.10.0),
        "phpunit/php-code-coverage": "10.1" (10.1.14),
        "phpunit/phpunit": "10.1" (10.5.24),
        "spatie/laravel-ignition": "2.0" (2.8.0)
    }
},
```

2. Melakukan debugging.

2.1. Kompilasi kode program menggunakan beberapa menu debug sesuai bahasa pemrograman yang di gunakan.

```

public function index(Request $request)
{
    $search = $request->query('search');

    // Contoh field yang sesuai (Melakukan perbaikan).
    $sortField = $request->query('sortField', 'example_debugging'); // Default sorting field

    $sortOrder = $request->query('sortOrder', 'asc'); // Default sorting order

    var_dump($sortField); // Menggunakan var_dump untuk debugging

    // mengambil data karyawan (Menerapkan Akses Basis Data di Laravel)
    $query = Karyawan::query();

    // Algoritma searching karyawan
    if ($search) {
        $query->where('nama', 'LIKE', "%{$search}%")
            ->orWhere('jabatan', 'LIKE', "%{$search}%");
    }

    You, 2 days ago * first commit
    // Menggunakan prosedur untuk mendapatkan semua karyawan
    $karyawans = $this->procedures->getAllKaryawans();

    // Algoritma sorting pada query
    $karyawans = $query->orderBy($sortField, $sortOrder)->paginate(10);

    var_dump($karyawans); // Menggunakan var_dump untuk debugging

    return view('karyawans.index', compact('karyawans', 'sortField', 'sortOrder'));
}

```

2.2. Melakukan analisis kriteria kode program yang lolos proses build.

2.3. Melakukan analisis kriteria aplikasi yang berhasil di eksekusi.

2.4. Melakukan pencatatan kesalahan yang terjadi pada kode program baik sintaks, semantik, maupun logika program.

The screenshot shows a developer tool interface with two main sections. On the left, a sidebar lists various monitoring categories: Requests, Commands, Schedule, Jobs, Batches, Cache, Dumps, Events, and Exceptions. The 'Exceptions' tab is selected. On the right, the 'Exceptions' section displays two entries:

Type	#	Happened	Resolved
Illuminate\Database\QueryException	3	7s ago	●
Symfony\Component\Mailer\Exception\TransportException	1	2h ago	●

Below this, a detailed view of an exception is shown in a code editor-like window. The window has tabs for Message, Location, Context, and Stacktrace. The Stacktrace tab is active, displaying the following code snippet with line numbers 820 through 839. Line 829 is highlighted with a red background:

```

820     // message to include the bindings with SQL, which will make this exception a
821     // lot more helpful to the developer instead of just the database's errors.
822     catch (Exception $e) {
823         if ($this->isUniqueConstraintError($e)) {
824             throw new UniqueConstraintViolationException(
825                 $this->getName(), $query, $this->prepareBindings($bindings), $e
826             );
827         }
828     }
829     throw new QueryException(
830         $this->getName(), $query, $this->prepareBindings($bindings), $e
831     );
832 }
833 }
834 /**
835 * Determine if the given database exception was caused by a unique constraint violation.
836 *
837 * @param \Exception $exception
838 * @return bool
839 */

```

3. Memperbaiki program.

3.1. Merumuskan perbaikan terhadap kesalahan kompilasi maupun build.

3.2. Melakukan perbaikan dan simpan hasil perbaikan tersebut.

```

24     public function index(Request $request)
25     {
26         $search = $request->query('search');
27
28         // Contoh field yang sesuai (Melakukan perbaikan). You, 2 days ago • first co
29         $sortField = $request->query('sortField', 'nama'); // Default sorting field
30
31         $sortOrder = $request->query('sortOrder', 'asc'); // Default sorting order
32

```

Requests			
Verb	Path	Status	Duration
GET	/karyawans	200	471ms
GET	/karyawans	200	376ms
GET	/karyawans	200	474ms
GET	/_ignition/health-check	200	623ms
GET	/karyawans	500	779ms

J.620100.031.01/Melakukan Profiling Program

- Mengumpulkan data waktu eksekusi komponen – komponen yang ada pada program.

1.1. Mengukur waktu eksekusi function, procedure, atau method program.

GET	/karyawans?search=search	200	367ms	8h ago	...
GET	/karyawans	200	359ms	8h ago	...
POST	/karyawans	302	253ms	8h ago	...
GET	/karyawans/create	200	353ms	8h ago	...
GET	/karyawans	200	355ms	8h ago	...
DELETE	/karyawans/2	302	297ms	8h ago	...
GET	/karyawans	200	319ms	8h ago	...
PUT	/karyawans/3	302	345ms	8h ago	...
GET	/karyawans/3/edit?_token=z0pPDENQoZDf1pBueqNIG8...	200	368ms	8h ago	...
GET	/karyawans	200	980ms	8h ago	...
GET	/karyawans	200	328ms	8h ago	...
GET	/dashboard	200	578ms	8h ago	...
POST	/login	302	715ms	8h ago	...

1.2. Mengukur penggunaan memory eksekusi function, procedure, atau method program.

1.3. Mengidentifikasi modul – module pada program teridentifikasi bermasalah.

Request Details	
Time	June 28th 2024, 3:52:44 PM (8h ago)
Hostname	RIDHO
Method	GET
Controller Action	App\Http\Controllers\KaryawanController@index
Middleware	web, auth, verified, admin
Path	/karyawans?search=search
Status	200
Duration	367 ms
IP Address	127.0.0.1
Memory usage	24 MB
Tags	Auth:1

Requests				Search Tag	
Verb	Path	Status	Duration	Happened	
GET	/karyawans	200	471ms	18m ago	...
GET	/karyawans	200	376ms	18m ago	...
GET	/karyawans	200	474ms	18m ago	...
GET	/_ignition/health-check	200	623ms	24m ago	...
GET	/karyawans	500	779ms	24m ago	...
GET	/karyawans	200	443ms	26m ago	...

2. Menentukan bottleneck performa yang ada pada program.

2.1. Mengidentifikasi bottleneck performa pada program.

2.2. Mengidentifikasi dampak negatif bottleneck terhadap performa.

3. Merancang solusi untuk menghilangi/mengurangi bottleneck.

3.1. Menjelaskan rancangan metode.

3.2. Menunjukkan peningkatan performa rancangan metode.

4. Menentukan kompleksitas algoritma.

4.1. Mengidentifikasi algoritma pada program terindikasi bermasalah

4.2. Menentukan metode untuk mengukur kompleksitas terhadap algoritma.

4.3. Mengidentifikasi kompleksitas algoritma yang berdampak penurunan performa.

```

public function index(Request $request) You, 2 days ago • F
{
    $search = $request->query('search');

    // Contoh field yang sesuai (Melakukan perbaikan).
    $sortField = $request->query('sortField', 'nama'); // Default

    $sortOrder = $request->query('sortOrder', 'asc'); // Default

    // mengambil data karyawan (Menerapkan Akses Basis Data di Layar)
    $query = Karyawan::query();

    // // Algoritma searching karyawan
    // if ($search) {
    //     $query->where('nama', 'LIKE', "%{$search}%")
    //         ->orWhere('jabatan', 'LIKE', "%{$search}%");
    // }

    // Algoritma searching karyawan simulasi bottleneck
    if ($search) {
        $query->where(function ($q) use ($search) {
            // Menambahkan beberapa kondisi where untuk simulasi
            for ($i = 0; $i < 1000; $i++) {
                $q->orWhere('nama', 'LIKE', "%{$search}%")
                    ->orWhere('jabatan', 'LIKE', "%{$search}%");
            }
        });
    }
}

```

```

public function index(Request $request)      You, 2 days ago • first commit
{
    $search = $request->query('search');

    // Contoh field yang sesuai (Melakukan perbaikan).
    $sortField = $request->query('sortField', 'nama'); // Default sorting field

    $sortOrder = $request->query('sortOrder', 'asc'); // Default sorting order

    // mengambil data karyawan (Menerapkan Akses Basis Data di Laravel)
    $query = Karyawan::query();

    // Algoritma searching karyawan
    if ($search) {
        $query->where('nama', 'LIKE', "%{$search}%")
            ->orWhere('jabatan', 'LIKE', "%{$search}%");
    }

    // Algoritma searching karyawan simulasi bottleneck
    // if ($search) {
    //     $query->where(function ($q) use ($search) {
    //         // Menambahkan beberapa kondisi where untuk simulasi bottleneck
    //         for ($i = 0; $i < 1000; $i++) {
    //             $q->orWhere('nama', 'LIKE', "%{$search}%")
    //                 ->orWhere('jabatan', 'LIKE', "%{$search}%");
    //         }
    //     });
    // }

    // Menggunakan prosedur untuk mendapatkan semua karyawan
    $karyawans = $this->procedures->getAllKaryawans();

```

Requests

Verb	Path	Status	Duration	Happened
GET	/karyawans?search=search	200	444ms	1s ago
GET	/karyawans?search=search	200	968ms	16s ago

Request Details

Time	June 28th 2024, 11:48:22 PM (1:02m ago)
Hostname	RIDHO
Method	GET
Controller Action	App\Http\Controllers\KaryawanController@index
Middleware	web, auth, verified, admin
Path	/karyawans?search=search
Status	200
Duration	968 ms
IP Address	127.0.0.1
Memory usage	26 MB
Tags	Auth:1

Request Details

Time	June 28th 2024, 11:48:22 PM (1:02m ago)
Hostname	RIDHO
Method	GET
Controller Action	App\Http\Controllers\KaryawanController@index
Middleware	web, auth, verified, admin
Path	/karyawans?search=search
Status	200
Duration	968 ms
IP Address	127.0.0.1
Memory usage	26 MB
Tags	Auth:1

J.620100.032.01/Menerapkan Code Review

1. Mengevaluasi kesesuaian kode dengan spesifikasi nya.

1.1. Mengidentifikasi kesesuaian kode dengan spesifikasi nya

```
0 references | 0 overrides
public function store(Request $request)    You, 4 weeks ago * first commit
{
    // Validasi input
    $request->validate([
        // Menjelaskan tipe data sesuai kaidah pemrograman
        'nama' => 'required|string|max:255',
        'jabatan' => 'required|string|max:255',
        'gaji' => 'required|integer|min:0',
    ]);

    // Menjelaskan variable sesuai kaidah pemrograman
    $data = $request->except('_token');

    // Logika tambahan sebelum penyimpanan
    if (strlen($data['nama']) < 3) {
        return redirect()->back()->withErrors(['nama' => 'Nama terlalu pendek.']);
    }

    // Menyimpan Data ke Database (Menerapkan Akses Basis Data di Laravel)
    $this->procedures->addKaryawan($data);

    return redirect()->route('karyawans.index')->with('success', 'Karyawan created successfully.');
}
```

1.2. Mengidentifikasi ketidak – sesuaian kode dengan ketentuan

```
// Menentukan metode yang sesuai    You, 4 weeks ago * first commit
0 references | 0 overrides
public function store(Request $request)
{
    // Validasi input
    $request->validate([
        // Menjelaskan tipe data sesuai kaidah pemrograman
        'nama' => 'required|string|max:255',
        'jabatan' => 'required|string|max:255',
        'gaji' => 'required|integer|min:0',
    ]);

    // Menjelaskan variable sesuai kaidah pemrograman
    $data = $request->except('_token');

    // Logika tambahan sebelum penyimpanan
    if (strlen($data['nama']) < 100) { // Validasi ini mungkin tidak sesuai dengan ketentuan
        return redirect()->back()->withErrors(['nama' => 'Nama terlalu pendek.']);
    }

    // Menyimpan Data ke Database (Menerapkan Akses Basis Data di Laravel)
    $this->procedures->addKaryawan($data);

    return redirect()->route('karyawans.index')->with('success', 'Karyawan created successfully.');
}
```

```

0 references | 0 overrides
public function store(Request $request) You, 4 weeks ago + first commit
{
    // Validasi input
    $request->validate([
        // Menjelaskan tipe data sesuai kaidah pemrograman
        'nama' => 'required|string|max:255',
        'jabatan' => 'required|string|max:255',
        'gaji' => 'required|integer|min:0',
    ]);

    // Menjelaskan variable sesuai kaidah pemrograman
    $data = $request->except('_token');

    // Logika tambahan sebelum penyimpanan
    if (strlen($data['nama']) < 3) { // Sesuai dengan spesifikasi
        return redirect()->back()->withErrors(['nama' => 'Nama terlalu pendek.']);
    }

    // Menyimpan Data ke Database (Menerapkan Akses Basis Data di Laravel)
    $this->procedures->addKaryawan($data);

    return redirect()->route('karyawans.index')->with('success', 'Karyawan created successfully.');
}

```

2. Memperbaiki kode sesuai dengan codingguidlines dan best practices.

2.1. Memperbaiki kode yang tidak sesuai coding – guidelines tanpa merubah spesifikasi nya

2.2. Memperbaiki kode yang tidak menerapkan bestpractices.

```

0 references | 0 overrides
public function store(Request $request) {
    // Validasi input
    $request->validate([
        // Menjelaskan tipe data sesuai kaidah pemrograman
        'nama' => 'required|string|max:255',
        'jabatan' => 'required|string|max:255',
        'gaji' => 'required|integer|min:0',
    ]);

    // Menjelaskan variable sesuai kaidah pemrograman
    $data = $request->except('_token');

    // Logika tambahan sebelum penyimpanan
    if (strlen($data['nama']) < 3) { // Sesuai dengan spesifikasi
        return redirect()->back()->withErrors(['nama' => 'Nama terlalu pendek.']);
    }

    // Menyimpan Data ke Database (Menerapkan Akses Basis Data di Laravel)
    $this->procedures->addKaryawan($data);

    return redirect()->route('karyawans.index')->with('success', 'Karyawan created successfully.');
}

```

```

public function store(Request $request)
{
    // Validasi input
    $request->validate([
        // Menjelaskan tipe data sesuai kaidah pemrograman
        'nama' => 'required|string|max:255',
        'jabatan' => 'required|string|max:255',
        'gaji' => 'required|integer|min:0',
    ]);

    // Menjelaskan variable sesuai kaidah pemrograman
    $data = $request->except('_token');

    // Logika tambahan sebelum penyimpanan
    if (strlen($data['nama']) < 3) {
        return redirect()->back()->withErrors(['nama' => 'Nama terlalu pendek.']);
    }

    // Menyimpan Data ke Database (Menerapkan Akses Basis Data di Laravel)
    $this->procedures->addKaryawan($data);

    return redirect()->route('karyawans.index')->with('success', 'Karyawan created successfully.');
}

```

3. Membuat pengecualian penulisan kode terhadap coding-guidelines.

3.1. Mengidentifikasi kode yang memang sebaiknya tidak perlu coding-guidelines.

3.2. menulis komentar yang menjelaskan kode pengecualian.

```
public function store(Request $request)      You, 4 weeks ago • First commit
{
    var_dump($request->all());
    die();

    // Validasi input
    $request->validate([
        // Menjelaskan tipe data sesuai kaidah pemrograman
        'nama' => 'required|string|max:255',
        'jabatan' => 'required|string|max:255',
        'gaji' => 'required|integer|min:0',
    ]);

    // Menjelaskan variable sesuai kaidah pemrograman
    $data = $request->except('_token');

    // Logika tambahan sebelum penyimpanan
    if (strlen($data['nama']) < 3) {
        return redirect()->back()->withErrors(['nama' => 'Nama terlalu pendek.']);
    }

    // Menyimpan Data ke Database (Menerapkan Akses Basis Data di Laravel)
    $this->procedures->addKaryawan($data);    Unreachable code detected

    return redirect()->route('karyawans.index')->with('success', 'Karyawan created successfully.');
}

0 references | 0 overrides
public function store(Request $request) {
    // Debugging code, not following best practices
    var_dump($request->all());
    die(); // This will be removed in production

    $validatedData = $request->validate([
        // Unreachable code detected
        'nama' => 'required|string|max:255',
        'jabatan' => 'required|string|max:255',
        'gaji' => 'required|integer|min:0',
    ]);

    if (strlen($validatedData['nama']) < 3) {
        return redirect()->back()->withErrors(['nama' => 'Nama terlalu pendek.']);
    }

    $this->procedures->addKaryawan($validatedData);    Unreachable code detected

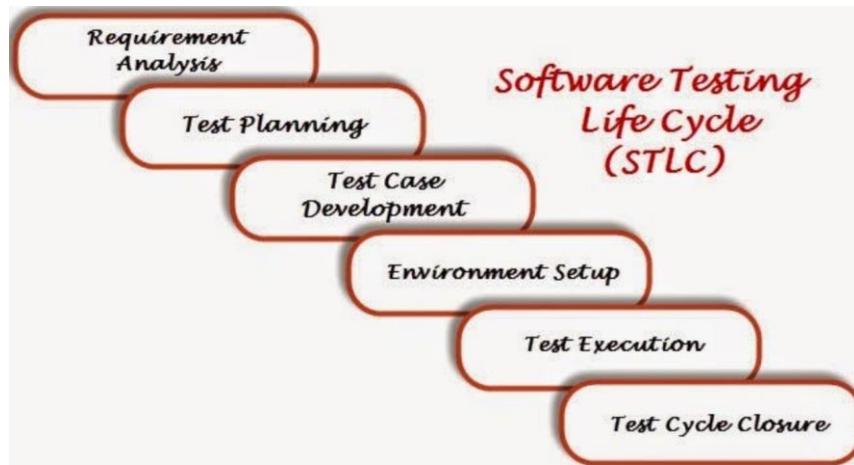
    return redirect()->route('karyawans.index')->with('success', 'Karyawan created successfully.');
}
```

J.620100.033.02/Melaksanakan Pengujian Unit Program

1. Menentukan kebutuhan uji coba dalam pengembangan.

1.1. Mengidentifikasi prosedur uji coba aplikasi sesuai dengan software testing live cycle

- Phase 1 — Requirement Analysis
- Phase 2 — Test Planning
- Phase 3 — Test Case Development
- Phase 4 — Environment Setup
- Phase 5 — Test Execution
- Phase 6 — Test Cycle Closure



1.2. Menentukan tools uji coba

- PHPUnit
- Laravel testing tools (Faker, Laravel Factories)

```
You, 4 days ago | 1 author (You) |  Install |  Update |  Check
1 [ You, 4 days ago • first commit
2
3 "name": "laravel/laravel",
4 "type": "project",
5 "description": "The skeleton application for the Laravel framework.",
6 "keywords": ["laravel", "framework"],
7 "license": "MIT",
8 "require": {
9     "php": "^8.1",
10    "guzzlehttp/guzzle": "^7.2" (v7.8.1),
11    "laravel/framework": "^10.10" (v10.48.14),
12    "laravel/sanctum": "^3.3" (v3.3.3),
13    "laravel/telescope": "^5.1" (v5.1.0),
14    "laravel/tinker": "^2.8" (v2.9.0)
15 },
16 "require-dev": {
17     "fakerphp/faker": "^1.9.1" (v1.23.1),
18     "laravel/breeze": "^1.29" (v1.29.1),
19     "laravel/pint": "^1.0" (v1.16.1),
20     "laravel/sail": "^1.18" (v1.29.3),
21     "mockery/mockery": "^1.4.4" (1.6.12),
22     "nunomaduro/collision": "^7.0" (v7.10.0),
23     "phpunit/php-code-coverage": "^10.1" (10.1.14),
24     "phpunit/phpunit": "^10.1" (10.5.24),
25     "spatie/laravel-ignition": "^2.0" (2.8.0)
26 }
```

1.3. Mengidentifikasi standar dan kondisi uji coba.

- Data karyawan yang valid harus memiliki nama, jabatan, dan gaji.
- Menguji batasan input untuk setiap field.

```
0 references | 0 overrides
public function test_karyawan_requires_nama_jabatan_gaji()
{
    $this->withoutMiddleware(); // Nonaktifkan middleware

    $response = $this->post('/karyawans', [
        'nama' => '',
        'jabatan' => '',
        'gaji' => ''
    ]);

    $response->assertSessionHasErrors(['nama', 'jabatan', 'gaji']);
}
```

2. Mempersiapkan dokumentasi uji coba.

2.1. Menentukan kebutuhan untuk uji coba di tentukan

- Memastikan semua endpoint CRUD berfungsi dengan baik.
- Memastikan validasi input berjalan sesuai aturan.

2.2. Melaksanakan uji coba dengan variasi kondisi.

- Uji dengan data lengkap dan valid.
- Uji dengan data yang tidak lengkap atau tidak valid.

Karyawan (Tests\Feature\Karyawan)

- ✓ Can create karyawan
- ✓ Can read karyawan
- ✓ Can update karyawan
- ✓ Can delete karyawan
- ✓ Cannot create karyawan without required fields
- ✓ Cannot read nonexistent karyawan
- ✓ Cannot update karyawan without required fields
- ✓ Cannot delete nonexistent karyawan

2.3. Membuat scenario uji coba

- Skenario: Pengguna membuat data karyawan baru.
 - 1. Nonaktifkan middleware CSRF.
 - 2. Kirim permintaan POST untuk membuat karyawan dengan data `nama`, `jabatan`, dan `gaji` .
 - 3. Verifikasi bahwa respons menghasilkan status 302 (berhasil dan dialihkan).
 - 4. Pastikan bahwa satu data karyawan telah dibuat di database.
- Skenario: Pengguna membaca data karyawan yang ada.
 - 1. Nonaktifkan middleware CSRF.
 - 2. Buat user menggunakan factory dan autentikasi sebagai user tersebut.
 - 3. Buat data karyawan menggunakan factory.
 - 4. Kirim permintaan GET untuk membaca data karyawan.
 - 5. Verifikasi bahwa respons menampilkan `nama`, `jabatan`, dan `gaji` karyawan.
- Skenario: Pengguna memperbarui data karyawan yang ada.
 - 1. Nonaktifkan middleware CSRF.
 - 2. Buat data karyawan awal menggunakan model Eloquent.
 - 3. Buat user dan autentikasi sebagai user tersebut.
 - 4. Kirim permintaan PUT untuk memperbarui data karyawan dengan ID yang spesifik.
 - 5. Verifikasi bahwa tidak ada error dan dialihkan setelah pembaruan berhasil.

- 6. Muat ulang model dari database.
- 7. Verifikasi bahwa data telah diperbarui dengan benar
- o **Skenario: Pengguna menghapus data karyawan yang ada.**
 - 1. Nonaktifkan middleware CSRF.
 - 2. Buat data karyawan awal menggunakan model Eloquent.
 - 3. Buat user dan autentikasi sebagai user tersebut.
 - 4. Kirim permintaan DELETE untuk menghapus data karyawan dengan ID yang spesifik.
 - 5. Verifikasi bahwa tidak ada error dan dialihkan setelah penghapusan berhasil.

Test Case Scenarios

Positive Testing

`test_can_create_karyawan`

Skenario: Pengguna membuat data karyawan baru.

Langkah:

- 1. Nonaktifkan middleware CSRF.
- 2. Kirim permintaan POST untuk membuat karyawan dengan data 'nama', 'jabatan', dan 'gaji'.
- 3. Verifikasi bahwa respons menghasilkan status 302 (berhasil dan dialihkan).
- 4. Pastikan bahwa satu data karyawan telah dibuat di database.

`test_can_read_karyawan`

Skenario: Pengguna membaca data karyawan yang ada.

Langkah:

- 1. Nonaktifkan middleware CSRF.
- 2. Buat user menggunakan factory dan autentikasi sebagai user tersebut.
- 3. Buat data karyawan menggunakan factory.
- 4. Kirim permintaan GET untuk membaca data karyawan.
- 5. Verifikasi bahwa respons menampilkan 'nama', 'jabatan', dan 'gaji' karyawan.

`test_can_update_karyawan`

Skenario: Pengguna memperbarui data karyawan yang ada.

3. Mempersiapkan data uji.

3.1. Mengidentifikasi data uji unit test

- o Data valid untuk nama, jabatan, gaji.
- o Data tidak valid untuk masing-masing field.

3.2. Membangkitkan data uji unit test.

- o Gunakan Laravel Factories untuk membuat data dummy.

```
 0 references | 0 implementations | 10d, 4 days ago | 1 author (you)
11 class KaryawanFactory extends Factory
12 {
13     /**
14      * The name of the factory's corresponding model.
15      *
16      * @var string
17      */
18     protected $model = Karyawan::class;
19
20     /**
21      * Define the model's default state.          You, 4 days ago • first commit
22      *
23      * @return array<string, mixed>
24      */
25     public function definition(): array
26     {
27         return [
28             'nama' => $this->faker->name,
29             'jabatan' => $this->faker->jobTitle,
30             'gaji' => $this->faker->numberBetween(3000000, 10000000),
31         ];
32     }
33 }
34
```

4. Melaksanakan prosedur uji coba.

4.1. Mendesain scenario uji coba.

- Membuat unit test untuk setiap operasi CRUD di KaryawanTest.

4.2. Mendesain prosedur uji coba dalam algoritma.

- **Create karyawan:** Cek respon status dan database.
 - **Read karyawan:** Cek respon status dan konten.
 - **Update karyawan:** Cek respon status dan perubahan data di database.
 - **Delete karyawan:** Cek respon status dan penghapusan data di database.

```
0 references | 0 implementations | You, 49 minutes ago | 1 author (You)
0 class KaryawanTest extends TestCase
1 {
2     use RefreshDatabase;
3
4     /**
5      * A basic feature test example.
6      *
7      * @return void
8      */
9
10    // Positive Testing
11
12    0 references | 0 overrides
13    public function test_can_create_karyawan()
14    {
15        // Menonaktifkan middleware untuk verifikasi CSRF
16        $this->withoutMiddleware();
17
18        $response = $this->post('/karyawans', [
19            'nama' => 'John Doe',
20            'jabatan' => 'Manager',
21            'gaji' => 5000000,
22        ]);
23
24        $response->assertStatus(302); // Redirected after successful creation
25
26        $this->assertCount(1, Karyawan::all());
27    }

```

```
0 references | 0 overrides
> 38     public function test_can_read_karyawan()
39     {
40         // Menonaktifkan middleware untuk verifikasi CSRF
41         $this->withoutMiddleware();
42
43         $user = User::factory()->create();
44
45         $this->actingAs($user);
46
47         $karyawan = Karyawan::factory()->create();
48
49         $response = $this->get('/karyawans');
50
51         $response->assertSee($karyawan->nama);
52         $response->assertSee($karyawan->jabatan);
53         $response->assertSee($karyawan->gaji);
54     }
55
```

```
0 references | 0 overrides
② 56     public function test_can_update_karyawan()
57     {
58         // Disable CSRF middleware for testing purposes
59         $this->withoutMiddleware();
60
61         // Create initial employee data using Eloquent model
62         $karyawan = Karyawan::create([
63             'id' => 1, // Set specific ID
64             'nama' => 'Initial Name',
65             'jabatan' => 'Initial Jabatan',
66             'gaji' => 500000,
67         ]);
68
69         // Authenticate as a user
70         $user = User::create([
71             'name' => 'Test User',
72             'email' => 'test@example.com',
73             'password' => bcrypt('password'),
74         ]);
75         $this->actingAs($user);
76
77         // Send update request using employee ID
78         $response = $this->put('/karyawans/' . $karyawan->id, [
79             'nama' => 'Initial Name',
80             'jabatan' => 'Initial Jabatan',
81             'gaji' => 500000,
82         ]);
83
84         // Assert no errors and redirect after successful update
85         $response->assertSessionHasNoErrors()
86             ->assertRedirect('/karyawans');
```

```

0 references | 0 overrides
101  public function test_can_delete_karyawan()
102  {
103      // Disable CSRF middleware for testing purposes
104      $this->withoutMiddleware();
105
106      // Create initial employee data using Eloquent model
107      $karyawan = Karyawan::create([
108          'id' => 1, // Set specific ID
109          'nama' => 'Initial Name',
110          'jabatan' => 'Initial Jabatan',
111          'gaji' => 5000000,
112      ]);
113
114      // Authenticate as a user
115      $user = User::create([
116          'name' => 'Test User',
117          'email' => 'test@example.com',
118          'password' => bcrypt('password'),
119      ]);
120
121      $this->actingAs($user);
122
123      // Send delete request using employee ID
124      $response = $this->from('/karyawans')
125          ->delete('/karyawans/' . $karyawan->id);
126
127      // Assert no errors and redirect after successful deletion
128      $response->assertSessionHasNoErrors()
129          ->assertRedirect('/karyawans');
130  }

```

Requests				
Verb	Path	Status	Duration	Happened
GET	/karyawans	200	524ms	2s ago
DELETE	/karyawans/1	302	393ms	3s ago
GET	/karyawans?search=test	200	486ms	5s ago
GET	/karyawans	200	494ms	13s ago
PUT	/karyawans/1	302	431ms	13s ago
GET	/karyawans/1/edit?_token=3kf8GXmkIZGTjjpFeiSo...	200	511ms	15s ago
GET	/karyawans	200	464ms	28s ago
POST	/karyawans	302	441ms	29s ago

The figure consists of two vertically stacked screenshots of the phpMyAdmin database management tool. Both screenshots show the same database structure and table selection.

Top Screenshot:

- Left Panel (Database Structure):** Shows a tree view of databases and tables. Under the 'db-karyawan' database, the 'Tabel' node is expanded, showing tables like 'Baru', 'failed_jobs', 'karyawans', 'migrations', 'password_resets', 'password_reset_tokens', 'personal_access_tokens', 'telescope_entries', 'telescope_entries_tags', 'telescope_monitoring', and 'users'.
- Top Bar:** Shows the server ('Server: database:3306'), database ('Database: db-karyawan'), and table ('Tabel: karyawans'). It also includes tabs for 'Jelajahi', 'Struktur', 'SQL', 'Cari', 'Tambahkan', 'Ekspor', 'Impor', 'Hak Akses', 'Operasi', and 'Trigger'.
- Message Bar:** Displays a green success message: 'Menampilkan baris 0 - 0 (total 1, Pencarian dilakukan dalam 0.0005 detik.)'.
- SQL Query:** Shows the SQL query: 'SELECT * FROM `karyawans`'.
- Table View:** Displays a single row of data with columns: id, nama, jabatan, gaji, created_at, and updated_at. The data is: id=1, nama='test', jabatan='test', gaji=1000000, created_at='2024-07-23 09:36:19', updated_at='2024-07-23 09:36:19'.
- Bottom Buttons:** Includes buttons for 'Ubah' (Edit), 'Salin' (Copy), 'Hapus' (Delete), 'Pilih Semua' (Select All), 'Dengan pilihan' (With Selection), 'Ubah' (Edit), 'Salin' (Copy), 'Hapus' (Delete), and 'Ekspor' (Export).
- Extra Options:** Includes 'Tampilan semua' (Show all), 'Jumlah baris:' (Number of rows: 25), 'Saring baris:' (Filter rows: Cari di tabel ini), and 'Operasi hasil kueri' (Query results operation).
- Bottom Buttons:** Includes 'Cetak' (Print), 'Salin ke clipboard' (Copy to clipboard), 'Ekspor' (Export), 'Tampilkan bagian' (Show section), and 'Buat tampilan' (Create view).

Bottom Screenshot:

- Left Panel (Database Structure):** Same as the top screenshot, showing the 'db-karyawan' database structure.
- Top Bar:** Shows the server ('Server: database:3306'), database ('Database: db-karyawan'), and table ('Tabel: karyawans'). It also includes tabs for 'Jelajahi', 'Struktur', 'SQL', 'Cari', 'Tambahkan', 'Ekspor', 'Impor', 'Hak Akses', 'Operasi', and 'Trigger'.
- Message Bar:** Displays a green success message: 'MySQL memberikan hasil kosong (atau nol baris). (Pencarian dilakukan dalam 0.0004 detik.)'.
- SQL Query:** Shows the SQL query: 'SELECT * FROM `karyawans`'.
- Table View:** Displays an empty table with columns: id, nama, jabatan, gaji, created_at, and updated_at.
- Bottom Buttons:** Includes 'Operasi hasil kueri' (Query results operation) and 'Buat tampilan' (Create view).

4.3. Melaksanakan uji coba.

- Jalankan test case menggunakan PHPUnit.
- Evaluasi hasil uji coba

```

PS D:\Bimbingan-Serkom-Analis-Program\Laravel-Karyawan\karyawan> php artisan test

WARN Test results may not be as expected because the XML configuration file did not pass validation:

Line 32:
- Element 'coverage', attribute 'processUncoveredFiles': The attribute 'processUncoveredFiles' is not allowed.

Line 33:
- Element 'include': This element is not expected.

PASS Tests\Unit\ExampleTest
✓ that true is true                                         0.23s

PASS Tests\Feature\Auth\AuthenticationTest
✓ login screen can be rendered                            2.71s
✓ users can authenticate using the login screen          0.32s
✓ users can not authenticate with invalid password      0.36s
✓ users can logout                                       0.13s

PASS Tests\Feature\Auth\EmailVerificationTest
✓ email verification screen can be rendered             0.11s
✓ email can be verified                                 0.17s
✓ email is not verified with invalid hash               0.14s

PASS Tests\Feature\Auth>PasswordConfirmationTest
✓ confirm password screen can be rendered              0.14s
✓ password can be confirmed                           0.13s
✓ password is not confirmed with invalid password    0.30s

PASS Tests\Feature\Auth>PasswordResetTest
✓ reset password link screen can be rendered           0.14s
✓ reset password link can be requested                0.19s
✓ reset password screen can be rendered               0.12s
✓ password can be reset with valid token              0.15s

PASS Tests\Feature\Auth>PasswordUpdateTest
✓ password can be updated                            0.14s
✓ correct password must be provided to update password 0.12s

PASS Tests\Feature\Auth\RegistrationTest
✓ registration screen can be rendered                 0.10s
✓ new users can register                           0.12s

PASS Tests\Feature\ExampleTest
✓ the application returns a successful response       0.07s

PASS Tests\Feature\KaryawanTest

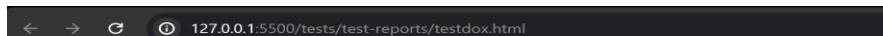
```

5. Mengevaluasi hasil uji coba.

5.1. Mencatat hasil uji coba.

5.2. Menganalisis hasil uji coba.

5.3. Melaporkan prosedur uji coba.



Authentication (Tests\Feature\Auth\Authentication)

- ✓ Login screen can be rendered
- ✓ Users can authenticate using the login screen
- ✓ Users can not authenticate with invalid password
- ✓ Users can logout

Email Verification (Tests\Feature\Auth\EmailVerification)

- ✓ Email verification screen can be rendered
- ✓ Email can be verified
- ✓ Email is not verified with invalid hash

Example (Tests\Feature\Example)

- ✓ The application returns a successful response

Example (Tests\Unit\Example)

- ✓ That true is true

Karyawan (Tests\Feature\Karyawan)

- ✓ Can create karyawan
- ✓ Can read karyawan
- ✓ Can update karyawan
- ✓ Can delete karyawan
- ✓ Cannot create karyawan without required fields
- ✓ Cannot read nonexistent karyawan
- ✓ Cannot update karyawan without required fields
- ✓ Cannot delete nonexistent karyawan

Password Confirmation (Tests\Feature\Auth\PasswordConfirmation)

- ✓ Confirm password screen can be rendered
- ✓ Password can be confirmed
- ✓ Password is not confirmed with invalid password

Password Reset (Tests\Feature\Auth\PasswordReset)

- ✓ Reset password link screen can be rendered
- ✓ Reset password link can be requested
- ✓ Reset password screen can be rendered
- ✓ Password can be reset with valid token

Password Update (Tests\Feature\Auth\PasswordUpdate)

- ✓ Password can be updated
- ✓ Correct password must be provided to update password

Profile (Tests\Feature\Profile)

- ✓ Profile page is displayed
- ✓ Profile information can be updated
- ✓ Email verification status is unchanged when the email address is unchanged
- ✓ User can delete their account
- ✓ Correct password must be provided to delete account

Registration (Tests\Feature\Auth\Registration)

- ✓ Registration screen can be rendered
- ✓ New users can register

5.4. Menyelesaikan kesalahan / error.

- Tidak ada error terdeteksi

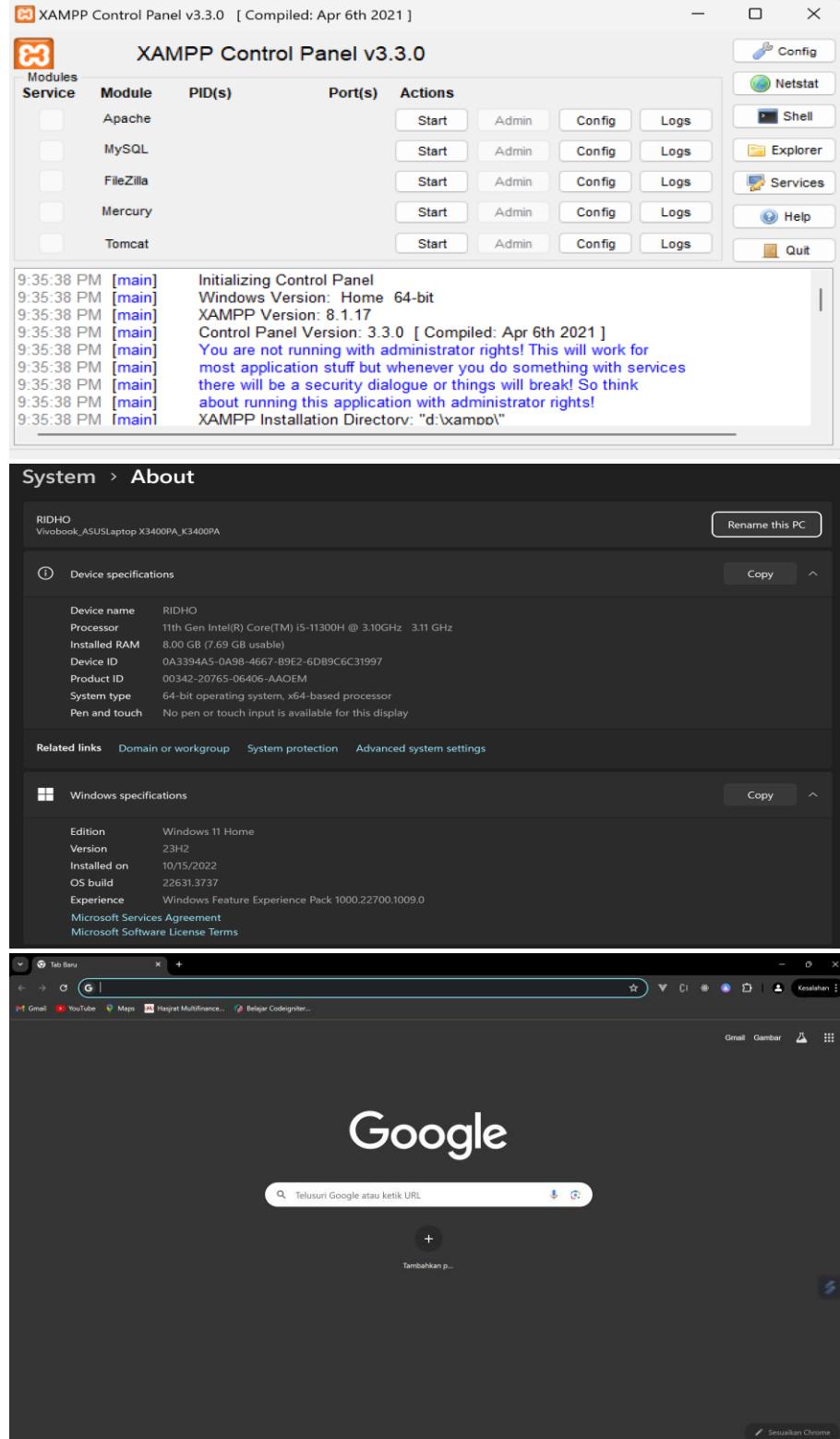
J.620100.034.02/Melaksanakan Pengujian Integrasi Program

1. Mempersiapkan dokumentasi peralatan dan lingkungan pengujian integrasi.

1.1. Menentukan peralatan pengujian sesuai dengan kebutuhan pengujian

- Hardware
 - Laptop / PC

- Software
 - Web Browser
 - XAMPP
 - Sistem Operasi (Windows)



1.2. Menyiapkan dokumen pendukung pengujian

- [db-karyawan.pdf](#)

- o [KaryawanTestScenarios.docx](#)

2. Mempersiapkan data uji.

2.1. Mengidentifikasi data uji integrasi program

2.2. Membangkitkan data uji integrasi program

```

0 references | 0 implementations
8  class KaryawanSeeder extends Seeder
9  {
10     /**
11      * Run the database seeds.
12      */
13     0 references | 0 overrides
14     public function run(): void
15     {
16         DB::table('karyawans')->insert([
17             [
18                 'nama' => 'John Doe',
19                 'jabatan' => 'Developer',
20                 'gaji' => 5000000,
21                 'created_at' => now(),
22                 'updated_at' => now(),
23             ],
24             [
25                 'nama' => 'Jane Smith',
26                 'jabatan' => 'Manager',
27                 'gaji' => 8000000,
28                 'created_at' => now(),
29                 'updated_at' => now(),
30             ],
31             [
32                 'nama' => 'Mark Taylor',
33                 'jabatan' => 'Tester',
34                 'gaji' => 4000000,
                 'created_at' => now(),
            ]
        ]);
    }
}

```

```

0 references | 0 implementations | You, 4 weeks ago | 1 author (You)
class KaryawanFactory extends Factory
{
    /**
     * The name of the factory's corresponding model.
     *
     * @var string
     */
    0 references
    protected $model = Karyawan::class;

    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */
    0 references | 0 overrides
    public function definition(): array
    {
        return [
            'nama' => $this->faker->name,
            'jabatan' => $this->faker->jobTitle,
            'gaji' => $this->faker->numberBetween(3000000, 10000000),
        ];
    }
}

```

3. Melaksanakan pengujian integrasi.

3.1. Menjalankan modul program sesuai dengan prosedur yang ditetapkan

3.2. Mengimplementasikan data atau kondisi sebagai masukan ke dalam program

3.3. Mencatat hasil pengujian kedalam lembar pengujian.

N O	Feature	URAIAN KEGIATAN	HASIL YANG DIHARAPKAN	KETERANGAN
1	- Create	<ul style="list-style-type: none"> - User click button Tambah Karyawan - User mengisi form tambah data karyawan yang berisi field name, jabatan, dan gaji. - User click button submit 	<ul style="list-style-type: none"> - Lolos seluruh validasi request. - Data berhasil tersimpan - Redirect ke halaman karyawan (Read) 	Success
2	- Read	<ul style="list-style-type: none"> - User mengakses halaman karyawan 	<ul style="list-style-type: none"> - User dapat melihat seluruh list karyawan pada halaman karyawan. 	Success
3	- Update	<ul style="list-style-type: none"> - User click button edit karyawan - User mengisi form edit karyawan yang berisi field name, jabatan, dan gaji. - User click button submit 	<ul style="list-style-type: none"> - Lolos seluruh validasi request. - Data berhasil terupdate dan tersimpan di database - Redirect ke halaman karyawan (Read) 	Success
4	- Delete	<ul style="list-style-type: none"> - User click button delete pada halaman karyawan (read) 	<ul style="list-style-type: none"> - Data berhasil terhapus - Redirect ke halaman karyawan (Read) 	Success
5	- Search	<ul style="list-style-type: none"> - User Input nama karyawan pada form pencarian - User click button pencarian 	<ul style="list-style-type: none"> - Data dengan spesifik nama yang user inputkan tampil pada halaman karyawan (read) 	Success
CAPTURE				

The image consists of three vertically stacked screenshots of a web application interface, likely a dashboard for managing employees. The interface is dark-themed with light-colored text and input fields.

Screenshot 1: A form for adding a new employee. It contains three input fields: "Nama" (Name) with placeholder "test", "Jabatan" (Position) with placeholder "test", and "Gaji" (Salary) with placeholder "1000000". Below the inputs is a "SUBMIT" button.

Nama	Jabatan	Gaji	Aksi
Jane Smith	Manager	8000000	<button>Edit</button> <button>Delete</button>
John Doe	Developer	5000000	<button>Edit</button> <button>Delete</button>
Mark Taylor	Tester	4000000	<button>Edit</button> <button>Delete</button>
test	test	1000000	<button>Edit</button> <button>Delete</button>

Screenshot 2: A search bar labeled "Cari Karyawan" (Search Employee) followed by a table listing four employees. The table has columns: Nama (Name), Jabatan (Position), Gaji (Salary), and Aksi (Actions). Each row includes an "Edit" button and a "Delete" button.

Screenshot 3: The same search bar and table as in Screenshot 2, showing the same four employees listed.

The image consists of two vertically stacked screenshots of a web application interface. Both screenshots have a dark background and a header with a logo, 'Dashboard', and 'Karyawan'.

Top Screenshot (Modal View):

- Fields:**
 - Nama:** test edit
 - Jabatan:** test
 - Gaji:** 1000000
- Buttons:** A white 'SUBMIT' button at the bottom left of the form.

Bottom Screenshot (List View):

- Search Bar:** Cari Karyawan
- Table Headers:** Nama, Jabatan, Gaji, Aksi
- Data Rows:**

Nama	Jabatan	Gaji	Aksi
Jane Smith	Manager	8000000	<button>Edit</button> <button>Delete</button>
John Doe	Developer	5000000	<button>Edit</button> <button>Delete</button>
Mark Taylor	Tester	4000000	<button>Edit</button> <button>Delete</button>
test edit	test	1000000	<button>Edit</button> <button>Delete</button>

4. Menganalisis data pengujian integrasi.

- 4.1. Menganalisa yang terkait sesuai dengan standar pengembangan perangkat lunak
 - 4.2. Mengavaluasi data hasil keluaran kesesuaian nya dengan data di rencanakan
 - 4.3. Mencatat status pada lembar pengujian dari hasil perbandingan data tersebut ke dalam lembar hasil uji.
 - 4.4. Mencatat kondisi data yang tidak sesuai dan perkiraan kondisi tersebut kedalam lembar hasil uji.
5. Melaporkan hasil pengujian integrasi.
 - 5.1. Mencatat peralatan yang di gunakan untuk pengujian ke dalam lembar peralatan pengujian

- 5.2. Mencatat kondisi yang terjadi selama pengujian ke dalam lembar pengujian.**
- 5.3. Mencatat data yang di implementasikan dan data hasil pengujian.**
- 5.4. Mencatat analisis hasil pengujian sesuai dengan standar dokumentasi pengembangan perangkat lunak yang berlaku.**
- 6. Melaporkan dokumen pengujian.**
 - 6.1. Mendokumentasikan hasil pengujian menjadi laporan.**
 - 6.2. Melaporkan dokumentasi hasil pengujian.**
 - 6.3. Mengarsipkan dokumentasi hasil pengujian.**
 - Link Terkait : [UAT-KARYAWAN.docx](#)