Jobsheet Praktikum: Pertemuan 1

Menyiapkan Proyek Laravel dan Repositori Git Simple POS

Mata Kuliah	Pemrograman Web Lanjut (SIB245007)
Pertemuan	1 (Minggu 1)
Durasi	2 sesi × 170 menit
Sub-CPMK	Sub-CPMK 1: Mahasiswa mampu memahami konsep dasar web framework serta menerapkan routing, controller, dan pengelolaan basis data dalam pengembangan aplikasi web.
Kode Akhir	branch `chapter-01` di `github.com/se-polinema/simple-pos` (untuk membandingkan hasil akhirmu)

A. Capaian Praktikum

Setelah menyelesaikan jobsheet ini, kamu mampu:

1. Menyiapkan proyek Laravel 13 baru dengan SQLite sebagai basis data, lengkap dengan migrasi dan data contoh.
2. Menjalankan server pengembangan Laravel dan memverifikasi halaman selamat datang tampil dengan benar.
3. Mengenali bagian struktur folder Laravel yang akan sering disentuh sepanjang semester.
4. Menginisialisasi repositori Git dan membuat commit pertama mengikuti konvensi `increment N`.

B. Persiapan dan Prasyarat

- **Alat**: PHP 8.2 ke atas, Composer, Node.js (untuk `npm`), dan Git.
- **Kelanjutan kode**: tidak ada, pertemuan ini dimulai dari nol, proyek dibuat langsung di komputermu.
- **Verifikasi cepat** sebelum mulai, jalankan keempatnya satu per satu di terminal:

  ```
  php -v
  composer -V
  node -v
  git --version
  ```

  Kalau salah satu perintah tidak dikenali atau versi PHP di bawah 8.2, instal/perbarui dulu sebelum lanjut ke Langkah 1.

C. Langkah Kerja

Langkah 1: Konfigurasi identitas Git (sekali saja per komputer)

Git menandai setiap commit dengan nama dan email pembuatnya. Kalau komputer yang kamu pakai belum pernah dipakai Git sebelumnya, atur dulu supaya Langkah 7 nanti tidak gagal di tengah jalan.

```
git config --global user.name "Nama Kamu"
git config --global user.email "email@kamu.com"
```

> ✓ **Checkpoint:** perintah tidak mencetak apa-apa (berhasil secara diam-diam). Verifikasi dengan `git config --global user.name`: harus mencetak nama yang baru saja diisi.

Langkah 2: Membuat proyek Laravel baru

`composer create-project` menjalankan Composer, pengelola dependensi PHP: ia membaca `composer.json`, mengunduh setiap paket ke folder `vendor/`, lalu menghasilkan

`vendor/autoload.php`: berkas yang membuat setiap class di proyek ini langsung bisa dipakai tanpa `require`/`include` manual seperti PHP polos.

```
composer create-project laravel/laravel simple-pos
cd simple-pos
cp .env.example .env
php artisan key:generate
```

> ✓ **Checkpoint:** `composer create-project` mencetak daftar paket yang diunduh, diakhiri baris seperti `Application ready in simple-pos. You can now start using Composer!`. Perintah `key:generate` mencetak `INFO Application key set successfully.`

> ⚠ **Jika gagal:** kalau `composer create-project` berhenti dengan pesan yang menyinggung versi PHP (mis. `requires php ^8.2`), jalankan `php -v` untuk memastikan versi terpasang 8.2 ke atas. Laravel 13 tidak bisa dipasang di versi yang lebih lama. Kalau proses berhenti karena timeout jaringan, ulangi perintah yang sama; Composer melanjutkan dari paket yang belum terunduh.

Langkah 3: Memasang dependensi frontend

Selain `composer.json`, proyek Laravel terbaru juga menyertakan `package.json`: daftar dependensi JavaScript untuk toolchain frontend (Vite dan Tailwind CSS) yang mulai dipakai Pertemuan 3. `npm install` adalah padanan `composer install` di dunia JavaScript: membaca `package.json`, lalu mengunduh setiap paket ke folder `node_modules/`.

```
npm install
```

> ✓ **Checkpoint:** output diakhiri baris seperti `added N packages in Ns`, tanpa baris `npm error` di antaranya.

Langkah ini belum wajib untuk menjalankan Simple POS hari ini. Halaman bawaan Laravel tetap tampil meski `node_modules/` belum ada. Menjalankannya sekarang menghindari jeda instalasi mendadak begitu Pertemuan 3 mulai memakai Tailwind dan Alpine.js lewat Vite.

Langkah 4: Menghubungkan ke SQLite dan menjalankan migrasi

Proyek ini memakai SQLite alih-alih MySQL/PostgreSQL: seluruh basis data disimpan dalam satu berkas biasa, tanpa proses server terpisah yang harus dinyalakan dan diberi kredensial. Pastikan baris berikut ada di `.env` milikmu (bawaan Laravel 13 sudah mengatur ini secara default, cukup diverifikasi, bukan diubah):

```
# .env
DB_CONNECTION=sqlite
```

Buat berkas basis datanya, lalu jalankan seluruh migrasi disertai data contoh:

```
touch database/database.sqlite
php artisan migrate:fresh --seed
```

> ✓ **Checkpoint:** daftar migrasi tercetak satu per satu, masing-masing diakhiri kata `DONE`, tanpa satu pun baris error, diikuti pesan bahwa seeder selesai berjalan.

> ⚠ **Jika gagal:** pesan `database file does not exist` berarti Langkah `touch` di atas belum dijalankan atau salah lokasi (harus persis `database/database.sqlite`). Pesan `could not find driver` berarti ekstensi PHP `pdo_sqlite` belum aktif: cek `php -m | grep sqlite`, aktifkan di `php.ini` bila belum muncul, lalu ulangi.

Langkah 5: Menjalankan server pengembangan

```
php artisan serve
```

> ✓ **Checkpoint:** baris `INFO Server running on [http://127.0.0.1:8000]` tercetak, terminal tetap terbuka menunggu request. Buka `http://127.0.0.1:8000` di browser. Halaman selamat datang Laravel, bukan pesan error, adalah tanda seluruh langkah sebelumnya berhasil. Tekan `Ctrl+C` untuk menghentikan server saat sudah selesai diverifikasi.

> ⚠ **Jika gagal:** pesan yang menyebut port 8000 sudah dipakai berarti ada proses lain (mungkin `php artisan serve` sesi sebelumnya yang belum dimatikan) memakai port itu; jalankan `php artisan serve --port=8001` dan buka `http://127.0.0.1:8001` sebagai gantinya.

Langkah 6: Mengenal struktur proyek

Sebelum menulis kode fitur pertama di Pertemuan 2, kenali dulu bagian struktur folder Laravel yang akan sering disentuh sepanjang semester. Buka keempat folder berikut lewat editor kode, lalu isi tabel di bawah tanpa menulis kode apa pun.

1. `routes/` (khususnya `routes/web.php`): tempat setiap alamat URL yang bisa diakses pengguna aplikasi didaftarkan.
2. `app/Http/Controllers/`: tempat class yang memproses setiap permintaan (menerima input, memanggil model, memilih tampilan) akan bertambah satu demi satu mulai Pertemuan 2.
3. `database/migrations/`: tempat perubahan skema basis data didefinisikan sebagai kode, bukan diklik lewat aplikasi basis data terpisah.
4. `vendor/`: berisi seluruh paket Composer pihak ketiga, bandingkan dengan tiga folder di atas.

> ✓ **Checkpoint (isi tabel ini di laporanmu):**

Pertanyaan	Jawabanmu
Berapa banyak berkas migrasi bawaan di `database/migrations/`?	
Sebutkan 2 nama tabel yang dibuat migrasi bawaan tersebut	
Apakah `vendor/` boleh diedit manual? Kenapa?	
Apakah `vendor/` ikut ter-commit ke Git? Kenapa?	

Langkah 7: Menginisialisasi Git dan membuat commit `increment 1`

`composer create-project` tidak otomatis menyiapkan repositori Git, jadi langkah ini dilakukan manual dari dalam folder `simple-pos/`.

```
git init
```

> ✓ **Checkpoint:** pesan `Initialized empty Git repository in .../simple-pos/.git/`.

Tambahkan seluruh berkas proyek ke area staging, lalu buat commit pertama:

```
git add .
git commit -m "increment 1: proyek Laravel kosong"
```

> ✓ **Checkpoint:** ringkasan jumlah berkas yang di-commit, diakhiri baris seperti `N files changed, M insertions(+)`. Perhatikan bahwa `vendor/`, `node_modules/`, dan `.env` tidak ikut disebutkan dalam daftar. `.gitignore` bawaan Laravel sudah mengecualikan ketiganya.

Verifikasi commit itu tersimpan:

```
git log --oneline
```

> ✓ **Checkpoint:** satu baris berisi hash pendek diikuti pesan `increment 1: proyek Laravel kosong` persis seperti yang baru saja ditulis.

> ⚠ **Jika gagal:** kalau `git commit` menolak dengan pesan yang meminta `user.name`/`user.email`, kembali ke Langkah 1, jalankan kedua perintah `git config --global` di sana, lalu ulangi `git commit` di langkah ini.

D. Tugas dan Deliverable

Kumpulkan hal berikut sesuai format yang diminta asisten/dosen:

- Screenshot halaman selamat datang Laravel di `http://127.0.0.1:8000`.
- Output `git log --oneline` yang menunjukkan commit `increment 1` milikmu.
- Tabel isian Langkah 6 (eksplorasi struktur folder).
- **Tugas mandiri:** jelaskan dengan kata-katamu sendiri, dalam 3-5 kalimat, mengapa sebuah warung dengan satu kasir lebih cocok memakai arsitektur monolith dibanding microservices, sementara sebuah platform e-commerce nasional dengan jutaan pengguna sering memakai microservices.

E. Kriteria Penilaian

Komponen	Bobot	Kriteria Lengkap (100%)	Kriteria Minimum
Langkah kerja tuntas	40%	Proyek berjalan, seluruh Langkah 1-7 selesai	Server berjalan, sebagian langkah selesai
Checkpoint terverifikasi	30%	Screenshot welcome page + `git log` + tabel eksplorasi lengkap dan benar	Sebagian checkpoint terbukti
Tugas mandiri	20%	Penjelasan monolith vs microservices tepat dan berdiri sendiri (bukan salinan materi)	Jawaban ada meski belum lengkap
Kerapian commit	10%	Pesan commit persis `increment 1: proyek Laravel kosong`, `vendor/`/`node_modules`/`.env` tidak ter-commit	Commit ada meski pesan/isi kurang rapi