BAB IV
KESIMPULAN

4.1.Kesimpulan

Berdasarkan hasil implementasi dan pengujian yang telah dilakukan selama kegiatan magang sebagai *Back-End Developer*, dapat ditarik kesimpulan yang menjawab rumusan masalah sebagai berikut:

1. Arsitektur *Modular Monolith* berhasil diimplementasikan pada *framework* Laravel yang mendistribusikan layanan utama Manajemen Akun, Pemesanan, dan *Admin* ke unit server terpisah namun tetap terintegrasi pada lapisan data. Struktur ini terbukti efektif meminimalisir ketergantungan antar modul dan meningkatkan skalabilitas infrastruktur untuk setiap beban kerja operasional dibandingkan arsitektur monolitik tradisional.
2. Penerapan konsep *Multi-Tenant* pada modul autentikasi sukses menangani kompleksitas hak akses, di mana sistem mampu membedakan peran pengguna secara dinamis berdasarkan konteks *merchant* yang dikelola. Integrasi fitur pendukung seperti verifikasi OTP dan sistem *referral* juga berjalan aman tanpa mengganggu logika inti, menjamin fleksibilitas akses bagi *User*, *Merchant*, maupun *Admin*.
3. Logika bisnis pada modul operasional *Venue*, *Promotion*, *Notification* berhasil diekspos melalui API yang efisien, mencakup penanganan relasi polimorfik fasilitas dan validasi jadwal promosi yang kompleks. Berdasarkan pengujian *Black Box Testing* pada 38 kasus uji, seluruh fitur terbukti valid dengan tingkat keberhasilan 100%, menandakan kesiapan API dalam mendukung operasional bisnis secara optimal.

4.2.Saran

Mengingat adanya batasan masalah mengenai ruang lingkup pengerjaan yang hanya berfokus pada logika *backend* dan pengujian manual, diberikan saran pengembangan untuk penyempurnaan sistem di masa mendatang:

1. Disarankan untuk menambahkan pengujian *Automated Unit Testing* dan *Feature Testing* menggunakan tool seperti PHPUnit. Hal ini krusial untuk mendeteksi *bug* regresi secara dini dan menjamin stabilitas logika bisnis setiap kali dilakukan penambahan fitur baru di masa depan.
2. Sistem saat ini fokus pada operasional transaksional. Pengembangan selanjutnya sebaiknya menambahkan mekanisme *Centralized Logging* seperti Sentry, Laravel Telescope, atau Laravel Pulse untuk memantau kesehatan aplikasi secara *real-time*, serta membangun fitur analitik data untuk melacak perilaku pengguna dan tren penjualan yang berguna bagi pengambilan keputusan bisnis *merchant*.
3. Untuk memitigasi risiko serangan robot atau spam pada formulir publik, disarankan mengintegrasikan verifikasi Captcha seperti Google reCAPTCHA atau Cloudflare Turnstile, khususnya pada halaman login dan registrasi pengguna.
4. Perlu dilakukan agenda pemeliharaan rutin untuk memperbarui versi *framework* Laravel dan dependensi pendukung. Langkah ini penting untuk menutup celah keamanan yang mungkin muncul serta menjaga performa aplikasi tetap optimal mengikuti perkembangan teknologi PHP terbaru.