BAB IV
KESIMPULAN

4.1. Kesimpulan

Laporan ini telah berhasil merancang dan mengimplementasikan mekanisme RBAC pada arsitektur RESTful API Volunteerin.id. Model *Static Core* RBAC yang dikembangkan secara terstruktur mengintegrasikan elemen pengguna, peran (*Volunteer*, *Partner*, *Admin*), izin, dan sesi berbasis JWT, serta menerapkan prinsip *Least Privilege* secara ketat. Implementasi teknis pada lingkungan Node.js dan Express.js direalisasikan melalui mekanisme *middleware* bertingkat yang secara efektif memisahkan logika otentikasi dan otorisasi, serta menegakkan validasi kepemilikan data (*data ownership*) untuk mencegah akses ilegal antar-pengguna.

Validasi terhadap keandalan sistem telah dilakukan secara komprehensif melalui pengujian teknis internal dan evaluasi eksternal. Pengujian unit dan integrasi menunjukkan tingkat keberhasilan sempurna (100% pass rate) pada seluruh skenario uji, membuktikan bahwa sistem mampu menangani berbagai kasus batas dan memitigasi risiko keamanan seperti eskalasi hak akses. Kualitas implementasi ini diperkuat secara signifikan oleh pencapaian sebagai Juara 1 dalam Hackathon IT FEST 5.0, di mana stabilitas dan responsivitas keamanan sistem mendapatkan validasi objektif dengan skor total 93.625 dari dewan juri dalam simulasi lingkungan produksi nyata.

Secara keseluruhan, laporan ini menyimpulkan bahwa integrasi RBAC pada platform Volunteerin.id telah menghasilkan fondasi keamanan yang kokoh, stabil, dan siap produksi (*production-ready*). Keberhasilan ini tidak hanya menjawab seluruh rumusan masalah penelitian, tetapi juga memberikan kontribusi nyata dalam menciptakan ekosistem kerelawanan digital yang aman dan terpercaya. Dengan arsitektur yang terskala dan terdokumentasi dengan baik, sistem ini siap mendukung operasional platform yang berkelanjutan serta menjadi referensi teknis yang valid bagi pengembangan sistem serupa di masa depan.

4.2. Saran

Meskipun implementasi RBAC pada Volunteerin.id telah berjalan optimal, terdapat sejumlah rekomendasi strategis guna meningkatkan ketahanan (*robustness*), skalabilitas, dan keamanan sistem secara berkelanjutan di masa depan.

1. Implementasi *Audit Logging* dan *Monitoring*

   Menambahkan mekanisme pencatatan jejak audit (*audit trail*) yang komprehensif pada setiap aktivitas autentikasi dan otorisasi. Penyimpanan log aktivitas (mencakup *User ID*, *endpoint*, dan *timestamp*) yang terintegrasi dengan *dashboard monitoring*. Dimana ini diperlukan untuk mendeteksi anomali keamanan, memfasilitasi analisis forensik, serta menjamin kepatuhan terhadap regulasi pelindungan data.

2. Penerapan *Rate Limiting* dan *Proteksi Brute-Force*

   Untuk memitigasi risiko serangan *brute-force* pada endpoint autentikasi, sistem perlu dilengkapi dengan middleware *rate limiting*. Mekanisme ini sebaiknya mencakup pembatasan jumlah percobaan login, penguncian akun sementara setelah kegagalan berulang, serta verifikasi *CAPTCHA*. Langkah ini krusial untuk melindungi kredensial pengguna dari upaya akses ilegal.

3. Peningkatan ke *Permission-Level Authorization*

   Sistem dapat ditingkatkan dari kontrol berbasis peran (*role-level*) menjadi berbasis izin (*permission-level*) yang lebih granular. Dengan mendefinisikan matriks izin yang spesifik (seperti pemisahan hak *create*, *read*, *update*, *delete*), sistem akan memiliki fleksibilitas lebih tinggi dalam membedakan akses terhadap sumber daya milik sendiri (*own resources*) maupun sumber daya publik, tanpa perlu menambah kompleksitas struktur peran.

4. Migrasi ke *Dynamic* RBAC

   Untuk meningkatkan fleksibilitas sistem di masa depan, disarankan melakukan transisi dari konfigurasi peran yang bersifat statis (*hardcoded*) menuju mekanisme *Dynamic* RBAC berbasis basis data. Dengan membangun panel administrasi khusus, pengelolaan hak akses dapat dilakukan secara langsung

tanpa perlu melakukan *deployment* ulang aplikasi. Hal ini akan mempermudah *administrator* dalam mengubah kebijakan akses sekaligus meningkatkan efisiensi operasional sistem.