

UNIVERSITAS GUNADARMA
DIREKTORAT PROGRAM DIPLOMA TIGA TEKNOLOGI INFORMASI



TULISAN ILMIAH

**PEMBUATAN SITUS WEB PERDAGANGAN DARING UNTUK
AZKA GARDEN**

Nama	: Roberto Ocaviantyo Tahta Laksmana
NPM	: 31120049
Program Studi	: Manajemen Informatika
Pembimbing	: Dr. Romdhoni Susiloatmadja, SPd., MMSI

Diajukan Guna Memperoleh Gelar Ahli Madya

Jakarta

2025

PERNYATAAN ORIGINALITAS DAN PUBLIKASI

Saya yang bertanda tangan di bawah ini:

Nama : Roberto Ocaviantyo Tahta Laksmana
NPM : 31120049
Judul Penulisan Ilmiah : Pembuatan Situs Web Perdagangan Daring untuk Azka garden
Tanggal Sidang : Agustus 2025
Tanggal Lulus : Agustus 2025

Menyatakan bahwa tulisan ini adalah merupakan hasil karya saya sendiri dan dapat di publikasikan sepenuhnya oleh Universitas Gunadarma. Segala kutipan dalam bentuk apapun telah mengikuti kaidah, etika yang berlaku. Mengenai isi dan tulisan merupakan tanggung jawab penulis, bukan Universitas Gunadarma.

Demikian pernyataan ini dibuat dengan sebenarnya dan dengan penuh kesadaran.

Depok, 14 Agustus 2025



(Roberto Ocaviantyo Tahta Laksmana)



LEMBAR PENGESAHAN

Judul Penulisan Ilmiah : Pembuatan Situs Web Perdagangan Daring untuk
(PI) Azka garden
Nama : Roberto Ocaviantyo Tahta Laksmana
NPM : 31120049
Tanggal Sidang : Agustus 2025
Tanggal Lulus : Agustus 2025

Menyetujui

Pembimbing

Kepala Bagian Sidang Ujian

(Dr. Romdhoni Susiloatmadja, S.Pd., MMSI) (Dr. Edi Sukirman, SSi., MM., MIKom)

Kaprodi Manajemen Informatika

(Dr. Romdhoni Susiloatmadja, S.Pd., MMSI)

ABSTRAK

Roberto Ocaviantyo Tahta Laksmana, 31120049.

PEMBUATAN SITUS WEB PERDAGANGAN DARING UNTUK AZKA GARDEN

PI. Program Studi Manajemen Informatika, Direktorat Program Diploma Tiga Teknologi Informasi, Universitas Gunadarma, 2025

Kata Kunci: perdagangan daring, situs web, pemesanan daring, pembayaran daring, React, Supabase, pengujian kotak hitam, evaluasi penerimaan sistem oleh pengguna

(xxxiv + 136 + Lampiran)

Penelitian ini bertujuan mengembangkan prototipe situs web perdagangan daring untuk Azka Garden, yaitu sebuah Usaha Mikro Kecil dan Menengah (UMKM) yang menjual tanaman hias, yang sebelumnya beroperasi melalui media sosial. Situs web ini dibangun dengan React 18 dan TypeScript untuk antarmuka pengguna, serta Supabase untuk basis data PostgreSQL, autentikasi, dan Edge Functions guna memisahkan logika bisnis, antarmuka pengguna, dan manajemen data. Pengembangan sistem mengikuti tahapan *System Development Life Cycle* (SDLC) model Waterfall, mulai dari perencanaan, analisis, perancangan, implementasi, hingga pengujian. Pengujian sistem dilakukan melalui pengujian kotak hitam dan penerimaan sistem oleh pengguna untuk memastikan fungsionalitas sesuai kebutuhan pengguna. Hasil uji coba menunjukkan bahwa sistem dapat berjalan optimal di berbagai perangkat dan peramban, serta mendukung pengalaman pengguna yang baik. Situs web ini diharapkan dapat memperluas jangkauan pasar dan meningkatkan efisiensi operasional Azka Garden. Dengan struktur modular dan fondasi teknis yang kuat, sistem ini memiliki potensi untuk dikembangkan lebih lanjut, termasuk ke platform seluler.

Daftar Pustaka (2004 – 2025)

KATA PENGANTAR

Puji dan syukur saya panjatkan ke hadirat Tuhan Yang Maha Esa atas segala rahmat dan kasih-Nya, sehingga Tulisan Ilmiah berjudul “Pembuatan Situs Web Perdagangan Daring untuk Azka garden” ini dapat diselesaikan dengan baik. Tulisan Ilmiah ini disusun untuk memenuhi syarat kelulusan pada Program Studi Diploma Tiga Manajemen Informatika Universitas Gunadarma.

Selama proses penyusunan Tulisan Ilmiah ini, berbagai tantangan dan hambatan dihadapi. Namun, dengan bantuan dan dukungan dari berbagai pihak, penulisan ini akhirnya dapat diselesaikan. Oleh karena itu, ucapan terima kasih yang sebesar-besarnya disampaikan kepada semua pihak yang telah memberikan bantuan, baik secara langsung maupun tidak. Ucapan terima kasih disampaikan kepada:

1. Prof. Dr. E.S. Margianti, S.E., M.M., selaku Rektor Universitas Gunadarma.
2. Prof. Drs. Rudi Irawan, MSc., PhD., selaku Direktur Program Diploma Tiga Teknologi Informasi Universitas Gunadarma.
3. Dr. Romdhoni Susiloatmadja, S.Pd., MMSI, selaku Kepala Program Studi Manajemen Informatika.
4. Dr. Edi Sukirman, SSi., MM., MIKom., selaku Kepala Bagian Sidang Ujian.
5. Dr. Romdhoni Susiloatmadja, S.Pd., MMSI, selaku dosen pembimbing yang dengan sabar memberikan arahan, bimbingan, dan waktunya selama penyusunan Tulisan Ilmiah ini.
6. Para dosen Universitas Gunadarma yang telah memberikan ilmu dan bimbingan selama studi.
7. Orang tua tercinta, Bapak Ignatius Warsito dan Ibu Emilia Martina Evi Natalia, yang selalu memberikan dukungan dan doa tanpa henti.
8. Teman-teman seperjuangan yang selalu memberikan semangat dan dukungan selama proses penulisan ini.

Menyadari bahwa Tulisan Ilmiah ini masih belum sempurna, maka kritik

dan saran yang membangun sangat diterima untuk perbaikan di masa depan. Semoga Tulisan Ilmiah ini tidak hanya bermanfaat bagi penulis, tetapi juga memberikan kontribusi bagi pembaca dan pihak-pihak yang berkepentingan.

Semoga Tuhan Yang Maha Esa senantiasa memberikan rahmat dan bimbingan-Nya kepada kita semua, serta memberkati setiap langkah dalam menuntut ilmu.

Depok, 14 Agustus 2025.

Penulis,



Roberto Ocaviantyo Tahta Laksmana

DAFTAR ISI

PERNYATAAN ORIGINALITAS DAN PUBLIKASI	ii
LEMBAR PENGESAHAN	iii
ABSTRAK	iv
KATA PENGANTAR	v
DAFTAR ISI	vii
DAFTAR TABEL	xv
DAFTAR GAMBAR	xvii
DAFTAR LAMPIRAN	xxii
1. PENDAHULUAN.....	1
1.1 Latar Belakang	1
1.2 Batasan Masalah.....	1
1.3 Tujuan Penelitian.....	2
1.4 Metode Penelitian.....	2
1.5 Sistematika Penulisan.....	3
2. TINJAUAN PUSTAKA	4
2.1 Web, Situs Web, dan Perdagangan Daring	4
2.2 Arsitektur Aplikasi Web Modern	4
2.3 Protokol Web, Keamanan Transport, dan URL.....	5
2.4 Keamanan dan Kepatuhan.....	5
2.5 Basis Data dan Desain Skema.....	6
2.6 Sistem Pembayaran Daring	6
2.7 Struktur Navigasi Web	7
2.7.1 Struktur Navigasi Linear	7
2.7.2 Struktur Navigasi Hirarkis.....	7
2.7.3 Struktur Navigasi Jaringan	8
2.7.4 Struktur Navigasi Campuran	8
2.8 Metodologi Pengembangan Perangkat Lunak.....	9

2.9 PlantUML.....	9
2.10 Bahasa Pemodelan Terpadu (UML)	9
2.10.1 Notasi dalam Diagram Kasus Penggunaan	10
2.10.2 Notasi dalam Diagram Aktivitas	11
2.10.3 Notasi dalam Diagram Kelas.....	12
2.11 Pengujian Perangkat Lunak.....	13
3. PEMBAHASAN	15
3.1 Gambaran Umum Sistem	15
3.2 Tahap Perencanaan.....	15
3.3 Tahap Analisis	16
3.3.1 Analisis Kebutuhan Fungsional.....	16
3.3.2 Analisis Kebutuhan Nonfungsional.....	16
3.3.2.1 Spesifikasi Perangkat Keras	17
3.3.2.2 Spesifikasi Perangkat Lunak	17
3.3.2.3 Keamanan dan Keandalan	18
3.3.2.4 Kinerja dan Skalabilitas.....	18
3.3.2.5 Pengalaman dan Antarmuka Pengguna (UX dan UI)	19
3.3.3 Analisis Kebutuhan Infrastruktur	19
3.3.3.1 Server Lokal dan Produksi	19
3.3.3.2 Manajemen Basis Data	20
3.3.3.3 Pemeliharaan Versi dan Kolaborasi	20
3.4 Tahap Perancangan.....	21
3.4.1 Perancangan Diagram Kasus Penggunaan	21
3.4.1.1 Diagram Kasus Penggunaan Pengguna	23
3.4.1.2 Diagram Kasus Penggunaan Administrator.....	23
3.4.1.3 Diagram Kasus Penggunaan Pengembang	24
3.4.2 Perancangan Diagram Aktivitas	25
3.4.2.1 Diagram Aktivitas Pengguna Mengelola Akun	26
3.4.2.2 Diagram Aktivitas Pengguna Melakukan Proses Pembayaran.....	26

3.4.2.3 Diagram Aktivitas Pengguna Melihat Status Langganan dan Pesanan	27
.....	
3.4.2.4 Diagram Aktivitas Pengguna Memberi Ulasan (Direncanakan)	28
3.4.2.5 Diagram Aktivitas Administrator Menambahkan Produk (Direncanakan)	29
3.4.2.6 Diagram Aktivitas Administrator Menangani Pesanan (Direncanakan)	29
3.4.2.7 Diagram Aktivitas Pengembang Melakukan Peluncuran Sistem	30
3.4.3 Perancangan Diagram Kelas	31
3.4.3.1 Tabel UserProfile	31
3.4.3.2 Tabel UserProfile dengan Relasi	32
3.4.3.3 Tabel UserProfile dengan Relasi	32
3.4.3.4 Tabel Role	33
3.4.3.5 Tabel Role dengan Relasi	33
3.4.3.6 Tabel UserSession.....	34
3.4.3.7 Tabel StripeCustomer	34
3.4.3.8 Tabel StripeCustomer dengan Relasi.....	34
3.4.3.9 Tabel StripeSubscription.....	35
3.4.3.10 Tabel StripeSubscription dengan Relasi	36
3.4.3.11 Tabel StripeOrder.....	36
3.4.3.12 Tabel StripeOrder dengan Relasi	37
3.4.3.13 Tabel StripeUser Subscriptions.....	37
3.4.3.14 Tabel StripeUser Subscriptions dengan Relasi	38
3.4.3.15 Tabel StripeUserOrders	38
3.4.3.16 Tabel StripeUserOrders dengan Relasi	39
3.4.3.17 Tabel ProductCategory	39
3.4.3.18 Tabel ProductCategory dengan Relasi	39
3.4.3.19 Tabel Product	40
3.4.3.20 Tabel Product dengan Relasi.....	41
3.4.3.21 Tabel InventoryMovement	41

3.4.3.22 Tabel InventoryMovement dengan Relasi	42
3.4.3.23 Tabel Order.....	42
3.4.3.24 Tabel Order dengan Relasi	43
3.4.3.25 Tabel OrderItem	43
3.4.3.26 Tabel OrderItem dengan Relasi.....	44
3.4.3.27 Tabel Review.....	44
3.4.3.28 Tabel Review dengan Relasi	45
3.4.3.29 Tabel ChatThread	45
3.4.3.30 Tabel ChatThread dengan Relasi	46
3.4.3.31 Tabel ChatMessage	46
3.4.3.32 Tabel ChatMessage dengan Relasi.....	47
3.4.3.33 Tabel ChatThreadParticipant.....	47
3.4.3.34 Tabel ChatThreadParticipant dengan Relasi	48
3.4.3.35 Tabel PriceWhitelist	48
3.4.3.36 Tabel PriceWhitelist dengan Relasi	49
3.4.3.37 Tabel WebhookLog	49
3.4.3.38 Tabel WebhookLog dengan Relasi.....	50
3.4.3.39 Tabel AuditLog.....	50
3.4.3.40 Tabel AuditLog dengan Relasi	51
3.4.4 Perancangan Struktur Tabel Nilai Basis Data	51
3.4.4.1 Arsitektur Basis Data.....	52
3.4.4.2 Struktur Tabel Sistem.....	52
3.4.4.3 Relasi Antar Tabel Sistem Azka Garden.....	69
3.4.4.4 Unique Constraints dan Composite Keys	72
3.4.4.5 Check Constraints untuk Validasi Business Rules.....	73
3.4.4.6 Ringkasan Statistik Basis Data Azka Garden	74
3.4.5 Perancangan Struktur Navigasi Situs Web Perdagangan Dinamis Azka Garden	75
3.4.5.1 Arsitektur Navigasi Sistem	76
3.4.5.2 Struktur Navigasi Berdasarkan Tipe	76

3.4.5.3 Struktur Navigasi Hirarki (Hierarchical Navigation)	76
3.4.5.4 Struktur Navigasi Jaringan (Network Navigation)	77
3.4.5.5 Struktur Navigasi Campuran (Hybrid Navigation).....	77
3.4.5.6 Struktur Navigasi Linear (Linear Navigation).....	78
3.4.5.7 Struktur Navigasi Utama Azka Garden	79
3.4.5.8 Struktur Navigasi Responsif.....	80
3.4.5.9 Alur Navigasi Perjalanan Pengguna	80
3.4.5.10 Arsitektur Komponen Navigasi	81
3.4.6 Perancangan Antarmuka Pengguna	82
3.4.6.1 Perancangan Halaman Beranda (/)	82
3.4.6.2 Perancangan Halaman Tentang Kami (/about)	83
3.4.6.3 Perancangan Halaman Kontak (/contact)	84
3.4.6.4 Perancangan Halaman FAQ (/faq).....	84
3.4.6.5 Perancangan Halaman Katalog Produk (/products).....	85
3.4.6.6 Perancangan Halaman Koleksi Premium (/stripe-products).....	86
3.4.6.7 Perancangan Halaman Panduan Perawatan (/care-guide)	86
3.4.6.8 Perancangan Halaman Blog & Tips (/blog).....	87
3.4.6.9 Perancangan Halaman Login (/login).....	87
3.4.6.10 Perancangan Halaman Registrasi (/register).....	88
3.4.6.11 Perancangan Halaman Profil (/profile)	88
3.4.6.12 Perancangan Halaman Pesanan (/orders)	88
3.4.6.13 Perancangan Halaman Keranjang (/cart)	89
3.4.6.14 Perancangan Halaman Wishlist (/wishlist).....	90
3.4.6.15 Perancangan Halaman Customer Service (/customer-service)	90
3.4.6.16 Perancangan Halaman Ulasan dan Komentar (/reviews).....	91
3.4.6.17 Perancangan Portal Administrator (/admin/login)	91
3.4.6.18 Perancangan Portal Pengembang (/developer/login)	92
3.4.6.19 Perancangan Halaman Kebijakan Privasi (/privacy).....	92
3.4.6.20 Perancangan Halaman Syarat dan Ketentuan (/terms).....	93
3.4.6.21 Perancangan Halaman Kebijakan Cookie (/cookies).....	94

3.4.6.22 Perancangan Halaman Kebijakan Pengembalian (/returns)	94
3.4.6.23 Perancangan Halaman Informasi Pengiriman (/shipping)	95
3.4.6.24 Perancangan Halaman Karir (/careers)	96
3.4.6.25 Perancangan Halaman Pencarian (/search)	96
3.5 Tahap Implementasi	97
3.5.1 Implementasi Basis Data PostgreSQL dengan Supabase	98
3.5.1.1 Setup dan Konfigurasi Basis Data	98
3.5.1.2 Implementasi Tabel Categories.....	99
3.5.1.3 Implementasi Tabel Users.....	99
3.5.1.4 Implementasi Tabel Products.....	99
3.5.1.5 Implementasi Tabel Cart_Items	100
3.5.1.6 Implementasi Tabel Orders	100
3.5.1.7 Implementasi Tabel Order_Items.....	101
3.5.1.8 Implementasi Tabel Product_Reviews.....	101
3.5.1.9 Implementasi Tabel Wishlist.....	102
3.5.1.10 Implementasi Tabel Roles.....	102
3.5.1.11 Implementasi Tabel User_Profiles	103
3.5.1.12 Implementasi Tabel User_Sessions	103
3.5.1.13 Implementasi Tabel Inventory_Movements	104
3.5.1.14 Implementasi Tabel Stripe Integration	104
3.5.1.15 Implementasi Tabel Audit_Logs	105
3.5.1.16 Batasan Pemeriksaan untuk Validasi Aturan Bisnis.....	106
3.5.1.17 Ringkasan Statistik Basis Data Akhir	107
3.5.2 Implementasi Halaman Utama	107
3.5.2.1 Implementasi Halaman Beranda (/)	107
3.5.2.2 Implementasi Halaman Tentang Kami (/about).....	108
3.5.2.3 Implementasi Halaman Kontak (/contact).....	108
3.5.2.4 Implementasi Halaman Tanya Jawab (/faq).....	109
3.5.3 Implementasi Halaman Produk dan Layanan.....	109
3.5.3.1 Implementasi Halaman Katalog Produk (/products)	109

3.5.3.2 Implementasi Halaman Koleksi Premium (/stripe-products)	110
3.5.3.3 Implementasi Halaman Panduan Perawatan (/care-guide)	110
3.5.3.4 Implementasi Halaman Blog dan Tips (/blog).....	111
3.5.4 Implementasi Halaman Akun dan Transaksi	111
3.5.4.1 Implementasi Halaman Login (/login).....	112
3.5.4.2 Implementasi Halaman Registrasi (/register)	112
3.5.4.3 Implementasi Halaman Profil (/profile).....	113
3.5.4.4 Implementasi Halaman Pesanan (/orders)	113
3.5.4.5 Implementasi Halaman Keranjang (/cart).....	114
3.5.4.6 Implementasi Halaman Daftar Keinginan (/wishlist).....	114
3.5.5 Implementasi Halaman Komunikasi	115
3.5.5.1 Implementasi Halaman Layanan Pelanggan (/customer-service)...	115
3.5.5.2 Implementasi Halaman Ulasan dan Komentar (/reviews)	115
3.5.6 Implementasi Portal Khusus.....	116
3.5.6.1 Implementasi Portal Administrator (/admin/login).....	116
3.5.6.2 Implementasi Portal Pengembang (/developer/login)	116
3.5.7 Implementasi Halaman Informasi Hukum	117
3.5.7.1 Implementasi Halaman Kebijakan Privasi (/privacy).....	117
3.5.7.2 Implementasi Halaman Syarat dan Ketentuan (/terms)	118
3.5.7.3 Implementasi Halaman Kebijakan Kuki (/cookies).....	118
3.5.7.4 Implementasi Halaman Kebijakan Pengembalian (/returns)	119
3.5.7.5 Implementasi Halaman Informasi Pengiriman (/shipping).....	119
3.5.8 Implementasi Halaman Lainnya.....	120
3.5.8.1 Implementasi Halaman Karier (/careers).....	120
3.5.8.2 Implementasi Halaman Pencarian (/search)	121
3.6 Tahap Pengujian	121
3.6.1 Pengujian Kotak Hitam	122
3.6.2 Evaluasi Penerimaan Sistem oleh Pengguna.....	125
3.7 Hosting dan Penerapan.....	126
3.7.1 Konfigurasi Proyek Netlify (Setup eCommerce)	126

3.7.2 Integrasi Netlify dengan GitHub	127
3.7.3 Pemilihan Repository Target	127
3.7.4 Konfigurasi Build dan Direktori Publik	127
3.7.5 Publikasi Deployment ke Produksi	128
3.7.6 Verifikasi Log Build dan Artefak	128
3.8 Uji Coba Pada Beberapa Perangkat	129
3.8.1 Metodologi Validasi Pengujian.....	129
3.8.2 Analisis Hasil Pengujian Multi-Platform	131
3.8.3 Optimasi Berdasarkan Hasil Pengujian.....	132
4. PENUTUP	133
4.1 Kesimpulan.....	133
4.2 Saran.....	133
DAFTAR PUSTAKA.....	135
LAMPIRAN <i>LISTING PROGRAM</i>.....	136

DAFTAR TABEL

Tabel 2.1. Notasi dalam Diagram Kasus Penggunaan	10
Sumber: (Fawareh et al., 2024)	10
Tabel 2.2. Notasi dalam Diagram Aktivitas	11
Sumber: (Fowler, 2004)	11
Tabel 2.3. Notasi dalam Diagram Kelas.....	12
Sumber: (Rumbaugh dkk. 2004)	12
Tabel 3.1 Struktur Tabel Categories (Kategori Tanaman).....	52
Tabel 3.2 Struktur Tabel Users (Pelanggan).....	53
Tabel 3.3 Struktur Tabel Products (Produk Tanaman)	54
Tabel 3.4 Struktur Tabel Products (Produk Tanaman)	56
Tabel 3.5 Struktur Tabel Orders (Pesanan)	57
Tabel 3.6 Struktur Tabel Order_Items (Detail Pesanan)	59
Tabel 3.7 Struktur Tabel Product_Reviews (Review Produk)	60
Tabel 3.8 Struktur Tabel Wishlist (Daftar Keinginan)	61
Tabel 3.9 Struktur Tabel Roles (Manajemen Role).....	61
Tabel 3.10 Struktur Tabel User_Profiles (Profil Extended)	62
Tabel 3.11 Struktur Tabel User_Sessions (Session Tracking).....	63
Tabel 3.12 Struktur Tabel Inventory_Movements (Pergerakan Stok).....	64
Tabel 3.13 Struktur Tabel Stripe_Customers (Customer Stripe).....	65
Tabel 3.14 Struktur Tabel Stripe_Orders (Order Tracking Stripe).....	66
Tabel 3.15 Struktur Tabel Audit_Logs (Sistem Audit)	68
Tabel 3.16 Struktur Tabel Relasi Antar Tabel Sistem Azka Garden	69
Tabel 3.17 Struktur Tabel Unique Constraints dan Composite Keys.....	72
Tabel 3.18 Struktur Tabel Check Constraints untuk Validasi Business Rules	73
Tabel 3.19 Struktur Tabel Ringkasan Statistik Basis Data Azka Garden.....	75
Tabel 3.20 Hasil Pengujian Kotak Hitam Situs Web Azka Garden.....	122
Tabel 3.21 Hasil Evaluasi Penerimaan Sistem oleh Pengguna	125

Tabel 3.22 Hasil Uji Coba Multi-Platform..... 130

DAFTAR GAMBAR

Gambar 3.1 Diagram Kasus Penggunaan Utama	22
Gambar 3.2 Diagram Kasus Penggunaan untuk Aktor Pengguna	23
Gambar 3.3 Diagram Kasus Penggunaan untuk Aktor Administrator	24
Gambar 3.4. Diagram Kasus Penggunaan untuk Aktor Pengembang.....	24
Gambar 3.5. Diagram Aktivitas Utama.....	25
Gambar 3.6. Diagram Aktivitas Pengguna Mengelola Akun.....	26
Gambar 3.7. Diagram Aktivitas Pengguna Melakukan Proses Pembayaran	27
Gambar 3.8. Diagram Aktivitas Pengguna Melihat Status Langganan dan Pesanan	28
Gambar 3.9. Diagram Aktivitas Pengguna Memberi Ulasan.....	28
Gambar 3.10. Diagram Aktivitas Administrator Menambahkan Produk	29
Gambar 3.11. Diagram Aktivitas Administrator Menangani Pesanan	30
Gambar 3.12. Diagram Aktivitas Pengembang Melakukan Peluncuran Sistem...	30
Gambar 3.13 Diagram Kelas Utama	31
Gambar 3.14 Tabel UserProfile.....	31
Gambar 3.15 Tabel UserProfile.....	32
Gambar 3.16 Tabel <i>UserProfile</i> dengan Relasi.....	32
Gambar 3.17 Tabel <i>Role</i>	33
Gambar 3.18 Tabel <i>Role</i> dengan Relasi	33
Gambar 3.19 Tabel <i>UserSession</i>	34
Gambar 3.20 Tabel <i>StripeCustomer</i>	34
Gambar 3.21 Tabel <i>StripeCustomer</i> dengan Relasi	35
Gambar 3.22 Tabel <i>StripeSubscription</i>	35
Gambar 3.23 Tabel <i>StripeSubscription</i>	36
Gambar 3.24 Tabel <i>StripeOrder</i>	36
Gambar 3.25 Tabel <i>StripeOrder</i> dengan Relasi	37
Gambar 3.26 Tabel <i>StripeUser Subscriptions</i>	37

Gambar 3.27 Tabel <i>StripeUser Subscriptions</i> dengan Relasi	38
Gambar 3.28 Tabel <i>StripeUserOrders</i>	38
Gambar 3.28 Tabel <i>StripeUserOrders</i>	39
Gambar 3.29 Tabel <i>StripeUserOrders</i> dengan Relasi	39
Gambar 3.30 Tabel <i>ProductCategory</i>	39
Gambar 3.31 Tabel <i>ProductCategory</i> dengan Relasi.....	40
Gambar 3.32 Tabel <i>Product</i>	40
Gambar 3.33 Tabel <i>Product</i> dengan Relasi.....	41
Gambar 3.34 Tabel <i>InventoryMovement</i>	41
Gambar 3.35 Tabel <i>InventoryMovement</i> dengan Relasi.....	42
Gambar 3.36 Tabel <i>Order</i>	42
Gambar 3.37 Tabel <i>Order</i> dengan Relasi	43
Gambar 3.38 Tabel <i>OrderItem</i>	43
Gambar 3.39 Tabel <i>OrderItem</i> dengan Relasi.....	44
Gambar 3.40 Tabel <i>Review</i>	45
Gambar 3.41 Tabel <i>Review</i> dengan Relasi	45
Gambar 3.42 Tabel <i>ChatThread</i>	46
Gambar 3.43 Tabel <i>ChatThread</i> dengan Relasi	46
Gambar 3.44 Tabel <i>ChatMessage</i>	47
Gambar 3.45 Tabel <i>ChatMessage</i> dengan Relasi.....	47
Gambar 3.46 Tabel <i>ChatThreadParticipant</i>	48
Gambar 3.47 Tabel <i>ChatThreadParticipant</i> dengan Relasi	48
Gambar 3.48 Tabel <i>PriceWhitelist</i>	49
Gambar 3.47 Tabel <i>PriceWhitelist</i> dengan Relasi.....	49
Gambar 3.46 Tabel <i>WebhookLog</i>	50
Gambar 3.47 Tabel <i>WebhookLog</i> dengan Relasi.....	50
Gambar 3.46 Tabel <i>AuditLog</i>	51
Gambar 3.47 Tabel <i>AuditLog</i> dengan Relasi.....	51
Gambar 3.48 Struktur Navigasi Hirarki	76
Gambar 3.49 Struktur Navigasi Jaringan	77

Gambar 3.50 Struktur Navigasi Campuran.....	78
Gambar 3.51 Struktur Navigasi Linear	79
Gambar 3.52 Struktur Navigasi Utama Azka Garden.....	79
Gambar 3.53 Struktur Navigasi Responsif	80
Gambar 3.54 Alur Navigasi Perjalanan Pengguna	81
Gambar 3.55 Arsitektur Komponen Navigasi.....	82
Gambar 3.56 Perancangan Halaman Beranda (/).....	83
Gambar 3.57 Perancangan Halaman Tentang Kami (/about).....	83
Gambar 3.58 Perancangan Halaman Kontak (/contact).....	84
Gambar 3.59 Perancangan Halaman FAQ (/faq)	85
Gambar 3.60 Perancangan Halaman Katalog Produk (/products)	85
Gambar 3.61 Perancangan Halaman Koleksi Premium (/stripe-products)	86
Gambar 3.62 Perancangan Halaman Panduan Perawatan (/care-guide).....	86
Gambar 3.63 Perancangan Halaman Blog & Tips (/blog)	87
Gambar 3.64 Perancangan Halaman Login (/login)	87
Gambar 3.65 Perancangan Halaman Registrasi (/register)	88
Gambar 3.65 Perancangan Halaman Profil (/profile)	88
Gambar 3.66 Perancangan Halaman Pesanan (/orders)	89
Gambar 3.67 Perancangan Halaman Keranjang (/cart).....	89
Gambar 3.68 Perancangan Halaman Wishlist (/wishlist).....	90
Gambar 3.69 Perancangan Halaman Customer Service (/customer-service)	90
Gambar 3.70 Perancangan Halaman Ulasan dan Komentar (/reviews).....	91
Gambar 3.71 Perancangan Portal Administrator (/admin/login)	91
Gambar 3.72 Perancangan Portal Pengembang (/developer/login)	92
Gambar 3.73 Perancangan Halaman Kebijakan Privasi (/privacy).....	93
Gambar 3.74 Perancangan Halaman Syarat dan Ketentuan (/terms).....	93
Gambar 3.75 Perancangan Halaman Kebijakan Cookie (/cookies)	94
Gambar 3.76 Perancangan Halaman Kebijakan Pengembalian (/returns)	95
Gambar 3.77 Perancangan Halaman Informasi Pengiriman (/shipping)	95
Gambar 3.78 Perancangan Halaman Karir (/careers)	96

Gambar 3.79 Perancangan Halaman Pencarian (/search)	97
Gambar 3.80 Tampilan Basis Data.....	98
Gambar 3.81 Tampilan Setup dan Konfigurasi Basis Data.....	98
Gambar 3.82 Implementasi Tabel <i>Categories</i>	99
Gambar 3.83 Implementasi Tabel <i>Users</i>	99
Gambar 3.84 Implementasi Tabel <i>Products</i>	100
Gambar 3.85 Implementasi Tabel <i>Cart_Items</i>	100
Gambar 3.86 Implementasi Tabel <i>Orders</i>	101
Gambar 3.87 Implementasi Tabel <i>Order_Items</i>	101
Gambar 3.88 Implementasi Tabel <i>Product_Reviews</i>	102
Gambar 3.89 Implementasi Tabel <i>Wishlist</i>	102
Gambar 3.90 Implementasi Tabel <i>Roles</i>	103
Gambar 3.91 Implementasi Tabel <i>User Profiles</i>	103
Gambar 3.92 Implementasi Tabel <i>User_Sessions</i>	104
Gambar 3.93 Implementasi Tabel <i>Inventory_Movements</i>	104
Gambar 3.94 Implementasi Tabel <i>Stripe Customers</i>	105
Gambar 3.95 Implementasi Tabel <i>Stripe Orders</i>	105
Gambar 3.96 Implementasi Tabel <i>Audit_Logs</i>	105
Gambar 3.97 Batasan Pemeriksaan untuk Validasi Aturan Bisnis	106
Gambar 3.98 Ringkasan Statistik Basis Data Akhir.....	107
Gambar 3.99 Implementasi Halaman Beranda (/).....	107
Gambar 3.100 Implementasi Halaman Tentang Kami (/about)	108
Gambar 3.101 Implementasi Halaman Kontak (/contact)	108
Gambar 3.102 Implementasi Halaman Tanya Jawab (/faq)	109
Gambar 3.103 Implementasi Halaman Katalog Produk (/products).....	110
Gambar 3.104 Implementasi Halaman Koleksi Premium (/stripe-products).....	110
Gambar 3.105 Implementasi Halaman Panduan Perawatan (/care-guide).....	111
Gambar 3.106 Implementasi Halaman Blog & Tips (/blog)	111
Gambar 3.107 Implementasi Halaman Login (/login)	112
Gambar 3.108 Implementasi Halaman Registrasi (/register).....	112

Gambar 3.109 Implementasi Halaman Profil (/profile)	113
Gambar 3.110 Implementasi Halaman Pesanan (/orders).....	113
Gambar 3.111 Implementasi Halaman Keranjang (/cart)	114
Gambar 3.112 Perancangan Halaman Wishlist (/wishlist).....	114
Gambar 3.113 Implementasi Halaman Layanan Pelanggan (/customer-service)	115
Gambar 3.114 Implementasi Halaman Ulasan dan Komentar (/reviews).....	115
Gambar 3.115 Implementasi Portal Administrator (/admin/login)	116
Gambar 3.116 Implementasi Dasbor Administrator (/admin/dashboard)	116
Gambar 3.117 Implementasi Portal Pengembang (/developer/login)	117
Gambar 3.118 Implementasi Dasbor Pengembang (/developer/dashboard)	117
Gambar 3.119 Implementasi Halaman Kebijakan Privasi (/privacy)	118
Gambar 3.120 Implementasi Halaman Syarat dan Ketentuan (/terms)	118
Gambar 3.121 Implementasi Halaman Kebijakan Kuki (/cookies)	119
Gambar 3.122 Implementasi Halaman Kebijakan Pengembalian (/returns).....	119
Gambar 3.123 Implementasi Halaman Informasi Pengiriman (/shipping)	120
Gambar 3.124 Implementasi Halaman Karier (/careers)	120
Gambar 3.125 Implementasi Halaman Pencarian (/search).....	121
Gambar 3.126 Konfigurasi Proyek Netlify	126
Gambar 3.127 Integrasi Netlify dengan GitHub	127
Gambar 3.128 Pemilihan Repository Target	127
Gambar 3.129 Konfigurasi Build dan Direktori Publik	128
Gambar 3.130 Ringkasan Deploy	128
Gambar 3.131 Log dan File Hasil Deploy	129

DAFTAR LAMPIRAN

Lampiran 1. <i>Listing Program</i>	L-1
Lampiran 2. Tampilan Halaman Situs Web	L-109

1. PENDAHULUAN

1.1 Latar Belakang

Perdagangan daring telah menjadi komponen esensial dalam kehidupan modern, menawarkan efisiensi waktu dan kemudahan transaksi. Melalui koneksi internet, konsumen dapat melakukan pembelian tanpa harus mengunjungi toko fisik. Salah satu komoditas yang mengalami peningkatan permintaan di pasar daring adalah tanaman hias. Namun, keterbatasan sistem manajemen stok yang digunakan oleh pelaku usaha mikro sering kali menyebabkan ketidakseimbangan antara ketersediaan produk dan permintaan pelanggan.

Faktor-faktor seperti kesiapan teknologi, dukungan manajerial, serta persepsi terhadap risiko merupakan determinan utama dalam adopsi perdagangan daring, khususnya pada Usaha Mikro, Kecil, dan Menengah (UMKM). Pemanfaatan platform digital memungkinkan UMKM untuk memperluas pasar dan meningkatkan efisiensi operasional, meskipun penerapannya masih menghadapi berbagai tantangan teknis dan strategis.

Hasil observasi terhadap UMKM Azka Garden pada tahun 2024–2025 mengungkapkan adanya kendala dalam proses pemesanan, seperti validasi alamat yang kurang akurat dan konfirmasi pesanan yang lambat akibat proses manual. Situasi ini menunjukkan perlunya sistem perdagangan daring yang terotomatisasi, dengan fitur pengelolaan pesanan dan validasi informasi yang lebih andal.

Penelitian ini bertujuan untuk merancang dan menguji prototipe sistem perdagangan daring berbasis web dengan integrasi modul pengelolaan pesanan secara sistematis pada UMKM Azka Garden. Sistem ini dirancang sebagai dasar untuk pengembangan selanjutnya ke platform seluler, dengan mempertimbangkan aspek keamanan, kemudahan penggunaan, serta skalabilitas untuk mendukung pertumbuhan bisnis UMKM Azka Garden.

1.2 Batasan Masalah

Penelitian ini berfokus pada pengembangan dan pengujian situs web Azka Garden, dengan penekanan pada pengujian kotak hitam serta evaluasi penerimaan

sistem oleh pengguna sebagai metode untuk menilai fungsionalitas sistem. Penulisan ini tidak mencakup pengujian performa sistem, keamanan tingkat lanjut, atau pengembangan aplikasi berbasis seluler. Dengan demikian, ruang lingkup penelitian ini terbatas pada aspek-aspek yang berkaitan dengan fungsionalitas dasar sistem dan pengalaman pengguna dalam menggunakan situs web.

1.3 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk membuat situs web yang dapat digunakan untuk mengoptimalkan proses pemesanan tanaman hias secara daring dan memperluas jangkauan pasar UMKM Azka Garden. Selain itu, penelitian ini bertujuan untuk mengevaluasi kinerja sistem melalui tingkat penerimaan dan kenyamanan pengguna. Dengan mencapai tujuan ini, diharapkan sistem yang dikembangkan dapat memberikan solusi yang efektif bagi pelaku usaha mikro dalam mengelola penjualan dan pemesanan produk mereka.

1.4 Metode Penelitian

Penelitian ini menerapkan model Air Terjun (Waterfall) dalam kerangka Siklus Hidup Pengembangan Perangkat Lunak (SDLC) karena menyediakan alur kerja yang sistematis dan terstruktur. Tahapan penelitian meliputi:

1. Tahap Perencanaan, yang mencakup penyusunan rencana proyek serta persiapan infrastruktur seperti perangkat keras, perangkat lunak, dan jaringan internet.
2. Tahap Analisis, di mana kebutuhan sistem dikumpulkan melalui wawancara informal dengan pemilik usaha dan calon pengguna, dengan memperhatikan prinsip etika dan privasi. Sebanyak lima responden terlibat dalam proses ini.
3. Tahap Perancangan, yang melibatkan pembuatan model visual sistem, seperti diagram kasus penggunaan, diagram aktivitas, diagram kelas, serta rancangan awal antarmuka pengguna.
4. Tahap Implementasi, di mana sistem dikembangkan menggunakan React, Supabase, dan Stripe. Teknologi ini dipilih karena mendukung arsitektur yang responsif dan aman.

5. Tahap Pengujian, di mana sistem diuji menggunakan metode pengujian kotak hitam untuk memastikan setiap fungsi berjalan sesuai spesifikasi. Evaluasi pengguna dilakukan untuk menilai kenyamanan dan efektivitas antarmuka.

1.5 Sistematika Penulisan

Penulisan laporan penelitian ini disusun dalam empat bab, dengan sistematika penulisan sebagai berikut:

1. Bab 1 Pendahuluan. Bab ini mencakup latar belakang, batasan masalah, tujuan penelitian, metode penelitian, dan sistematika penulisan.
2. Bab 2 Tinjauan Pustaka. Bagian ini memuat kerangka teori dan kajian terdahulu yang relevan terkait perdagangan daring, React, Supabase, dan model SDLC Waterfall.
3. Bab 3 Pembahasan. Bab ini menjelaskan proses pengembangan sistem serta hasil pengujian yang telah dilakukan.
4. Bab 4 Penutup. Bagian ini menyajikan kesimpulan dan saran untuk pengembangan lebih lanjut.

Dengan sistematika tersebut, diharapkan pembaca dapat mengikuti proses pengembangan sistem secara menyeluruh, mulai dari landasan teori, perancangan teknis, hingga hasil evaluasi, yang dijelaskan secara berurutan dalam bab-bab selanjutnya.

2. TINJAUAN PUSTAKA

Bab ini menguraikan landasan teoretis yang mendukung perancangan dan pembangunan situs web perdagangan daring yang andal, aman, dan mudah digunakan. Pembahasan mencakup konsep web dan protokol, arsitektur antarmuka modern, keamanan dan kepatuhan, basis data relasional, sistem pembayaran daring, antarmuka dan pengalaman pengguna, struktur navigasi web, metodologi pengembangan, teknik pengujian, serta pemodelan dengan UML. Seluruh subbab merujuk pada literatur terpercaya dan praktik terbaik yang relevan.

2.1 Web, Situs Web, dan Perdagangan Daring

Web adalah infrastruktur jaringan yang memungkinkan pertukaran data menggunakan protokol HTTP dan HTTPS yang mengatur sintaks pesan serta pola komunikasi antara klien dan server (Fielding & Reschke, 2014). Situs web berperan sebagai medium dalam domain tertentu untuk menyajikan konten dan layanan interaktif kepada pengguna. Dalam konteks perdagangan daring, situs web lazimnya mengintegrasikan katalog produk, keranjang belanja, alur checkout, sistem pembayaran, serta pelacakan pesanan dan riwayat transaksi untuk mendukung layanan pelanggan dan akuntabilitas operasional (Turban et al., 2018). Perkembangan teknologi web modern mendorong pengalaman pengguna yang responsif sekaligus memungkinkan integrasi erat antara antarmuka, layanan data, dan penyedia layanan pihak ketiga.

2.2 Arsitektur Aplikasi Web Modern

Arsitektur antarmuka modern banyak mengadopsi konsep *Single Page Application* untuk menghadirkan interaksi yang cepat dan dinamis. React memanfaatkan mekanisme *virtual DOM* untuk mengoptimalkan proses perenderan serta pengelolaan keadaan antarmuka sehingga perubahan tampilan berlangsung efisien dari sisi kinerja dan terasa mulus bagi pengguna (Facebook, 2023). Konsep antarmuka tunggal telah dibahas luas dalam literatur pengembangan aplikasi satu halaman sebagai pendekatan yang efektif untuk meningkatkan pengalaman

pengguna dan efisiensi pemuatan konten (Fink & Flatow, 2014). Pada sisi layanan, pendekatan *Backend as a Service* seperti Supabase menyediakan PostgreSQL terkelola, autentikasi, penyimpanan berkas, dan kanal waktu nyata sehingga tim pengembang dapat berfokus pada logika bisnis tanpa terbebani kompleksitas pengelolaan infrastruktur (Supabase, 2024). Kombinasi antarmuka tunggal dan layanan siap pakai ini mendukung siklus pengembangan yang iteratif dan terukur, terutama bagi aplikasi transaksi yang membutuhkan sinkronisasi data yang andal.

2.3 Protokol Web, Keamanan Transport, dan URL

HTTP mendefinisikan sintaks pesan dan aturan pertukaran data antara klien dan server serta menjadi dasar komunikasi pada aplikasi web modern (Fielding & Reschke, 2014). Untuk menjaga kerahasiaan dan integritas data, HTTPS menerapkan TLS 1.3 sebagai lapisan pengamanan transport yang menyediakan enkripsi, integritas, dan autentikasi server sehingga data tidak mudah disadap atau dimodifikasi selama transmisi (Rescorla, 2018). Di sisi lain, URL berfungsi sebagai alamat sumber daya di web dengan struktur baku sesuai standar IETF yang memungkinkan penautan dan pengalaman konsisten di seluruh aplikasi.

2.4 Keamanan dan Kepatuhan

Keamanan sistem transaksi daring menuntut pendekatan berlapis yang meliputi pengamanan transport, kontrol akses aplikasi, serta kepatuhan terhadap standar industri. Pada lapisan transport, penerapan HTTPS dengan TLS 1.3 melindungi data dalam perjalanan dari intersepsi dan manipulasi yang tidak sah (Rescorla, 2018). Pada lapisan aplikasi dan data, kontrol akses yang kuat serta kebijakan pemisahan hak istimewa diperlukan agar hanya subjek berwenang yang dapat mengakses sumber daya tertentu. Implementasi kontrol akses tingkat baris atau Row Level Security pada PostgreSQL yang difasilitasi oleh Supabase memungkinkan pembatasan akses data secara granular per identitas pengguna sehingga mengurangi risiko kebocoran data antar pengguna dalam satu skema (Supabase, 2024). Untuk pemrosesan pembayaran, kepatuhan terhadap *Payment*

Card Industry Data Security Standard serta praktik tokenisasi memastikan data kartu tidak disimpan di sistem aplikasi melainkan diganti token yang aman dan tidak bernilai di luar konteks transaksi (PCI Security Standards Council, 2022). Manajemen sesi yang aman dapat memanfaatkan *JSON Web Token* untuk skenario tanpa status dengan tetap memperhatikan praktik pengamanan kunci, masa berlaku, serta pemisahan hak akses berdasarkan peran pengguna (Jones et al., 2015).

2.5 Basis Data dan Desain Skema

PostgreSQL dipilih karena dukungannya terhadap transaksi ACID, tipe data kaya termasuk JSON, indeks canggih, partisi, serta kebijakan keamanan seperti *Row Level Security* yang relevan untuk aplikasi dengan kebutuhan multi penyewa dan kontrol akses granular (PostgreSQL Global Development Group, 2024). Perancangan skema mengikuti prinsip normalisasi untuk meminimalkan redundansi dan menjaga integritas data pada relasi yang meliputi pengguna atau pelanggan, produk, stok, pesanan, detail pesanan, dan transaksi (Elmasri & Navathe, 2016). Rancangan yang baik mendukung konsistensi dan kinerja pada skala pertumbuhan yang wajar serta memudahkan perluasan fitur seperti ulasan dan percakapan waktu nyata.

2.6 Sistem Pembayaran Daring

Integrasi dengan penyedia pembayaran seperti Stripe menyediakan antarmuka pemrograman aplikasi yang komprehensif untuk pembuatan sesi checkout, penanganan metode pembayaran, dan verifikasi transaksi. Komponen Stripe Checkout dioptimalkan untuk tingkat konversi dan kepatuhan PCI, sementara mekanisme *webhook* memungkinkan sinkronisasi status pembayaran secara waktu nyata dari penyedia ke aplikasi sehingga status pesanan dapat diperbarui akurat dan tepat waktu tanpa intervensi manual (Stripe, 2024). Pada konteks lokal Indonesia, gateway seperti Midtrans menyediakan kanal pembayaran populer termasuk virtual account, QRIS, dan dompet elektronik yang dapat dipertimbangkan pada tahap integrasi berikutnya guna meningkatkan penerimaan

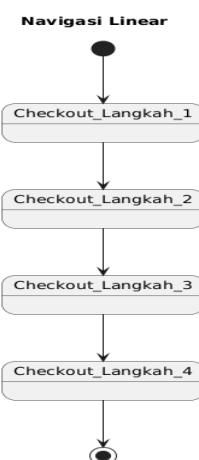
pengguna (Midtrans, 2024).

2.7 Struktur Navigasi Web

Struktur navigasi berperan krusial dalam kemudahan penemuan informasi dan keberhasilan tugas pengguna. Nielsen mengidentifikasi empat pola navigasi utama, yaitu linear (*linear*), hirarkis (*hierarchical*), jaringan (*network*), dan campuran (*composite*) yang masing-masing sesuai dengan konteks interaksi dan jenis konten yang berbeda (Nielsen, 2020).

2.7.1 Struktur Navigasi Linear

Struktur navigasi linear tepat untuk proses bertahap yang harus diikuti secara urut, seperti alur checkout. Dalam pola ini, setiap langkah dilalui berurutan untuk meminimalkan kesalahan dan memastikan kelengkapan data. Pengguna akan diarahkan dari satu langkah ke langkah berikutnya tanpa kebingungan, sehingga meningkatkan efisiensi dalam menyelesaikan tugas.

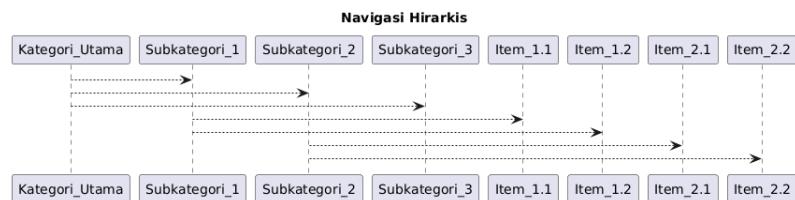


Gambar 2.1. Contoh Struktur Navigasi Linear

2.7.2 Struktur Navigasi Hirarkis

Struktur navigasi hirarkis memfasilitasi organisasi konten dari kategori umum menuju subkategori yang lebih spesifik. Misalnya, kategori Tanaman Indoor dapat bercabang ke Tanaman Meja dan Tanaman Gantung. Pendekatan ini

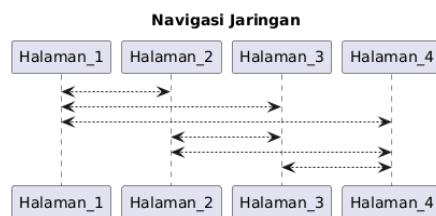
memudahkan penelusuran katalog berdasarkan pengelompokan yang sistematis, sehingga pengguna dapat dengan cepat menemukan informasi yang relevan.



Gambar 2.2. Struktur Navigasi Hirarkis

2.7.3 Struktur Navigasi Jaringan

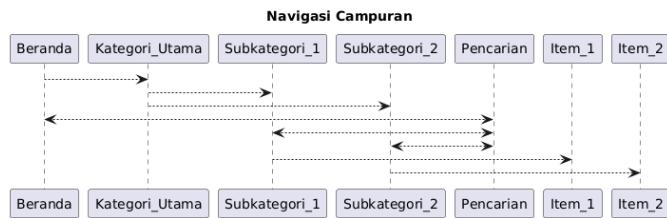
Struktur navigasi jaringan memungkinkan akses langsung antarhalaman tanpa perlu mengikuti urutan tetap. Melalui menu pintas, kolom pencarian, tautan kontekstual, dan rekomendasi, pengguna dapat dengan mudah menjelajahi konten yang tersedia. Pendekatan ini mengurangi waktu pencarian informasi dan memberikan fleksibilitas lebih bagi pengguna dalam menemukan apa yang mereka butuhkan.



Gambar 2.3. Struktur Navigasi Jaringan

2.7.4 Struktur Navigasi Campuran

Struktur navigasi campuran menggabungkan beberapa pola sekaligus untuk memberikan fleksibilitas yang lebih besar. Misalnya, katalog berbasis kategori yang dilengkapi fitur pencarian cepat, penyaringan, dan rekomendasi, serta dukungan jejak tautan atau breadcrumbs untuk orientasi pengguna. Pendekatan ini memungkinkan pengguna untuk memilih cara yang paling sesuai untuk menjelajahi konten, meningkatkan pengalaman pengguna secara keseluruhan.



Gambar 2.4. Struktur Navigasi Campuran

2.8 Metodologi Pengembangan Perangkat Lunak

Model Waterfall dalam kerangka Siklus Hidup Pengembangan Perangkat Lunak menekankan tahapan berurutan yang terdiri atas analisis kebutuhan, perancangan, implementasi, pengujian, dan pemeliharaan yang masing masing didokumentasikan dengan baik untuk memastikan keterlacakkan keputusan serta kemudahan perawatan di masa mendatang (Sommerville, 2016; Pressman & Maxim, 2020). Walaupun kurang luwes menghadapi perubahan kebutuhan yang cepat, model ini sesuai untuk proyek dengan ruang lingkup yang relatif stabil dan kebutuhan dokumentasi rapi seperti pengembangan prototipe terstruktur yang ditujukan untuk pembuktian konsep dan kesiapan integrasi.

2.9 PlantUML

PlantUML adalah alat berbasis teks untuk membuat diagram UML secara otomatis. Dengan integrasi ke *Visual Studio Code* dan sistem kontrol versi seperti Git, PlantUML memudahkan dokumentasi teknis secara *real time* dan terstruktur.

2.10 Bahasa Pemodelan Terpadu (UML)

Pemodelan dengan *Unified Modeling Language* (UML) digunakan untuk memvisualisasikan kebutuhan dan desain sistem melalui berbagai diagram standar. Use case diagram membantu memetakan interaksi aktor dengan fungsi sistem, activity diagram memodelkan alur proses serta logika kontrol, sedangkan class diagram merepresentasikan struktur data dan relasi antarkelas sebagai dasar perancangan skema dan lapisan domain aplikasi. Pemanfaatan UML memfasilitasi

komunikasi lintas pemangku kepentingan dan mempercepat transisi dari rancangan konseptual ke implementasi teknis yang konsisten serta terdokumentasi baik (Fowler, 2004; Rumbaugh et al., 2004).

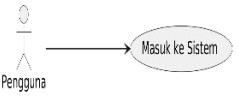
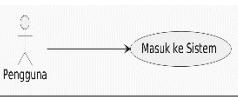
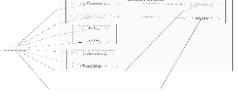
2.10.1 Notasi dalam Diagram Kasus Penggunaan

Diagram kasus penggunaan menggambarkan interaksi aktor (pengguna, administrator, dan pengembang) dengan fungsi sistem untuk memastikan bahwa seluruh kebutuhan fitur teridentifikasi dan terdokumentasi sebelum implementasi (Fawareh et al., 2024). Pendekatan ini memberikan gambaran yang jelas bagi tim pengembang dan pemangku kepentingan mengenai alur kerja sistem, sehingga memudahkan proses validasi dan koordinasi. Selain itu, diagram ini juga membantu dalam mengidentifikasi celah fungsional serta memprioritaskan kebutuhan pengembangan berdasarkan skenario penggunaan. Dengan demikian, penggunaan UML dalam perancangan sistem tidak hanya mendukung dokumentasi teknis yang baik, tetapi juga mempercepat komunikasi antar tim pengembang secara visual dan terstruktur. Tabel 2.1 menunjukkan notasi atau simbol dalam diagram kasus penggunaan. Simbol-simbol tersebut membantu visualisasi interaksi sistem dengan pengguna dan menyoroti fungsi utama secara terstruktur.

Tabel 2.1. Notasi dalam Diagram Kasus Penggunaan

Sumber: (Fawareh et al., 2024)

Notasi	Nama	Keterangan
 Pengguna	Aktor	Entitas eksternal yang berinteraksi langsung dengan sistem.
 Masuk ke Sistem	Kasus Penggunaan	Serangkaian aksi yang menghasilkan keluaran bernilai bagi aktor.

Notasi	Nama	Keterangan
	Asosiasi	Hubungan antara aktor dan kasus penggunaan.
	Menyertakan	Kasus tambahan yang selalu dipanggil oleh kasus utama.
	Memperluas	Kasus tambahan yang dipanggil secara opsional untuk menambahkan perilaku pada kasus utama.
	Batas Sistem	Pembatas grafis antara elemen sistem dan aktor.
	Paket	Pengelompokan elemen sub-diagram dalam satu konteks.

2.10.2 Notasi dalam Diagram Aktivitas

Diagram aktivitas memodelkan urutan aktivitas dan aliran kontrol dalam proses bisnis, sehingga logika dan alur data dapat diikuti dengan jelas (Fowler, 2004). Tabel 2.2 menunjukkan notasi atau simbol dalam diagram aktivitas. Tabel ini mempermudah pemahaman alur proses dan tanggung jawab peran dalam sistem.

Tabel 2.2. Notasi dalam Diagram Aktivitas

Sumber: (Fowler, 2004)

Notasi	Nama	Keterangan
	Inisialisasi	Simbol yang menandai permulaan dari alur aktivitas.
	Aktivitas	Langkah atau tindakan yang dilakukan dalam konteks proses bisnis.

Notasi	Nama	Keterangan
	Percabangan	Titik pengambilan keputusan untuk menentukan jalur alur berdasarkan kondisi tertentu.
	Finalisasi	Simbol yang menandai akhir dari alur aktivitas.
	Transisi	Aliran perpindahan dari satu aktivitas ke aktivitas lainnya.
	Jalur Peran	Pembagian aktivitas berdasarkan peran atau entitas pelaksana.
	Pengulangan	Konstruksi pengulangan (<i>loop/repeat</i>) yang bergantung pada kondisi tertentu.

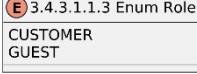
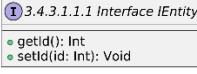
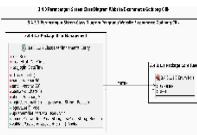
2.10.3 Notasi dalam Diagram Kelas

Diagram kelas menampilkan struktur statis sistem, termasuk atribut, metode, dan hubungan antar kelas, untuk merancang desain data dan arsitektur sistem (Rumbaugh dkk. 2004). Tabel 2.3 menunjukkan notasi dalam diagram kelas. Sajian notasi ini memperjelas hubungan antarelemen sistem dan membantu transisi dari desain konseptual ke implementasi teknis secara efektif.

Tabel 2.3. Notasi dalam Diagram Kelas

Sumber: (Rumbaugh dkk. 2004)

Notasi	Nama	Keterangan
	Kelas	Struktur objek yang memiliki atribut dan metode.

Notasi	Nama	Keterangan
	Enumerasi	Tipe data terbatas yang berisi sekumpulan nilai konstan.
	Antarmuka	Sekumpulan metode tanpa implementasi langsung.
	Paket	Kelompok logis dari kelas, antarmuka, dan subsistem terkait.
	Realisasi	Relasi yang menunjukkan bahwa suatu kelas mengimplementasikan antarmuka, digambarkan dengan garis putus-putus dan panah segitiga terbuka ke arah antarmuka.
	Asosiasi	Hubungan antar-kelas yang dapat berupa agregasi, komposisi, atau navigasi.

2.11 Pengujian Perangkat Lunak

Antarmuka yang baik memperhatikan hierarki visual, keterbacaan tipografi, kontras warna, dan *affordance* komponen, sedangkan pengalaman pengguna menekankan kemudahan navigasi, kecepatan, dan efektivitas pengguna dalam menyelesaikan tugas. Pedoman aksesibilitas *Web Content Accessibility Guidelines 2.1* dari W3C menjadi rujukan penting agar situs dapat diakses dan digunakan oleh sebanyak mungkin pengguna termasuk mereka yang memiliki keterbatasan (W3C, 2018). Desain responsif sebagaimana diperkenalkan oleh Marcotte memastikan tampilan adaptif lintas perangkat sehingga pengalaman penggunaan konsisten pada layar kecil maupun besar (Marcotte, 2011). Kerangka utilitas Tailwind CSS mendukung konsistensi komponen antarmuka dan percepatan penataan gaya sehingga pengembangan antarmuka lebih terstruktur dan mudah dipelihara seiring pertumbuhan fitur sistem (Tailwind Labs, 2024). Evaluasi pengalaman pengguna dapat menggunakan metrik seperti tingkat keberhasilan tugas, waktu penyelesaian

tugas, dan *System Usability Scale* untuk mengukur efektivitas serta efisiensi rancangan antarmuka (Sauro & Lewis, 2016).

Berdasarkan landasan teoritis tersebut, rancangan dan implementasi situs perdagangan daring diarahkan untuk memadukan arsitektur antarmuka modern, pengelolaan data yang andal dan aman, integrasi pembayaran yang patuh standar, serta pengalaman pengguna yang efektif. Bab selanjutnya menguraikan analisis kebutuhan dan perancangan sistem dengan mengacu pada prinsip-prinsip tersebut dan memisahkan contoh teori dari hasil rancangan khusus sistem.

3. PEMBAHASAN

3.1 Gambaran Umum Sistem

Situs web Azka Garden dikembangkan sebagai platform e-commerce yang mendukung transaksi jual beli tanaman hias secara daring. Sistem ini dirancang dengan arsitektur modular, yang memisahkan antarmuka publik yang dapat diakses oleh pengguna (seperti beranda, katalog produk, detail produk, keranjang belanja, proses pemesanan, dan halaman langganan) dari antarmuka administrator yang digunakan untuk pemeliharaan konten situs, data pengguna, transaksi pembayaran, serta pemeliharaan dan pencadangan basis data. Sistem ini dibangun menggunakan teknologi modern, termasuk *React 18* dan *TypeScript* untuk antarmuka pengguna, serta *Supabase* yang menyediakan basis data *Postgres*, autentikasi, Row Level Security (RLS), dan *Edge Functions*. *Stripe* digunakan untuk proses pembayaran, termasuk sesi checkout dan sinkronisasi order melalui webhook. Model pengembangan mengikuti pendekatan bertahap (iterative-incremental) dengan inspirasi dari tahapan *Waterfall*, yang mencakup perencanaan, analisis, perancangan, implementasi, dan pengujian, tetapi dieksekusi dalam sprint untuk mendukung ekspansi fitur secara progresif.

3.2 Tahap Perencanaan

Tahap perencanaan bertujuan untuk menetapkan ruang lingkup proyek, mendefinisikan tujuan, serta menentukan alokasi sumber daya yang diperlukan. Ruang lingkup awal mencakup integrasi pembayaran *Stripe* (one-time & subscription), struktur autentikasi pengguna menggunakan *Supabase Auth*, dasar keranjang belanja, dan checkout simulatif. Beberapa perangkat lunak yang digunakan meliputi *React 18*, *TypeScript*, *Vite*, *Tailwind CSS*, dan *React Router* untuk antarmuka pengguna, serta *Supabase* untuk layanan yang mencakup *Postgres*, *Auth*, dan *Edge Functions*. Rencana perluasan mencakup tabel produk, ulasan, chat, portal administrator, dan integrasi pembayaran lokal. Dengan perangkat keras yang memadai, pengembangan dan simulasi dilakukan pada perangkat komputer yang mendukung pengujian beban tinggi.

3.3 Tahap Analisis

Analisis sistem dilakukan untuk mengidentifikasi kebutuhan pengguna dan administrator. Dari perspektif pengguna, sistem harus mendukung berbagai aktivitas seperti registrasi akun, autentikasi, pencarian dan pemilihan produk, manajemen keranjang belanja, pemrosesan dan pelacakan pesanan, serta interaksi chat support di masa depan. Sementara itu, dari perspektif administrator, sistem diharapkan dapat memfasilitasi manajemen data produk, kategori, stok, verifikasi pembayaran, dan pemantauan kegagalan *webhook*.

3.3.1 Analisis Kebutuhan Fungsional

Kebutuhan fungsional menggambarkan kemampuan sistem yang harus ada untuk memenuhi tujuan bisnis dan operasional Azka Garden. Pengguna dapat melakukan pendaftaran akun, masuk ke akun, mengelola profil, menjelajahi katalog produk, melakukan pemesanan, dan menerima notifikasi status pesanan. Di sisi lain, administrator bertugas mengelola konten sistem serta proses-proses sisi server. Untuk mengatur akses berdasarkan tanggung jawab, digunakan model hak akses berbasis peran (role-based access control) sehingga setiap pengguna hanya dapat mengakses fitur yang sesuai dengan perannya. Terdapat tiga peran utama dalam sistem ini: pengguna (customer), administrator, dan pengembang. Fungsi saat ini mencakup autentikasi pengguna, katalog produk, keranjang belanja, dan integrasi pembayaran *Stripe*, sementara roadmap mencakup pengembangan produk CRUD, ulasan, chat support, dan portal administrator.

3.3.2 Analisis Kebutuhan Nonfungsional

Kebutuhan nonfungsional berperan penting dalam memastikan kualitas, keandalan, dan kenyamanan sistem secara keseluruhan. Beberapa aspek utama yang dianalisis dalam pengembangan situs web Azka Garden meliputi keamanan, performa, skalabilitas, serta pengalaman pengguna. Pada aspek keamanan, sistem wajib menggunakan autentikasi JWT dari *Supabase*, RLS untuk pembatasan akses,

dan *Edge Functions* untuk operasi sensitif. Untuk performa dan skalabilitas, pengoptimalan dilakukan melalui rendering cepat dengan *Vite*, caching, dan penggunaan *Supabase Realtime* untuk pembaruan tanpa polling. Pengalaman pengguna ditingkatkan dengan desain responsif menggunakan *Tailwind CSS*, komponen modular, dan fitur-fitur UX yang direncanakan seperti dark mode dan indikator status waktu nyata.

3.3.2.1 Spesifikasi Perangkat Keras

Sistem dikembangkan dan diuji pada konfigurasi berikut:

1. Prosesor: *Intel Core i7-11370H* (generasi Tiger Lake-U, fabrikasi 10 nm, TDP 35 watt).
2. Kecepatan Prosesor: Kecepatan dasar 3,30 GHz dengan kecepatan maksimum dinamis hingga 3,99 GHz.
3. Motherboard: *ASUSTeK FX516PC* dengan antarmuka bus berkecepatan tinggi.
4. Memori Utama: 8 GB jenis *DDR4*.
5. Layar Monitor: Berukuran 14 inci dengan resolusi tinggi.
6. Perangkat Masukan: Papan ketik (*keyboard*) dan tetikus (*mouse*) yang ergonomis.

3.3.2.2 Spesifikasi Perangkat Lunak

Perangkat lunak pendukung pengembangan dan pengujian mencakup:

1. Sistem Operasi: *Windows 10 64-bit* sebagai sistem operasi utama, dengan opsi alternatif *Linux Ubuntu LTS* dan *macOS* untuk pengujian lintas platform.
2. Peramban Utama: *Google Chrome* (versi 114+) dan *Microsoft Edge* (versi 113+) untuk pengujian antarmuka pengguna.
3. Kerangka Antarmuka Pengguna: *React 18* untuk pengembangan antarmuka pengguna berbasis komponen, dengan *TypeScript* untuk pengetikan statis.
4. Layanan Data: *Supabase* yang menyediakan *Postgres* sebagai sistem manajemen basis data, autentikasi, dan *Edge Functions*.

5. Pembayaran: *Stripe* untuk proses pembayaran, termasuk sesi checkout dan sinkronisasi *webhook*.
6. Manajemen Status: *React Context* dan *localStorage* untuk pengelolaan keranjang belanja dan sesi pengguna.
7. Ikon dan Antarmuka Pengguna: *lucide-react* untuk ikon dan komponen UI.
8. Alat Pengembangan dan Pembangunan: *Vite* sebagai *bundler* dan dev server untuk pengembangan yang cepat dan efisien.

3.3.2.3 Keamanan dan Keandalan

Langkah-langkah keamanan dirancang untuk menjaga integritas dan kerahasiaan data:

1. Penggunaan SSL/TLS untuk mengenkripsi koneksi antara pengguna dan server.
2. Enkripsi basis data untuk menyimpan informasi sensitif seperti sandi dan data transaksi.
3. Sistem direncanakan mengandalkan enkripsi at-rest dari penyedia (*Supabase*) + TLS in transit.
4. Pengawasan akses internal, audit keamanan rutin, serta pelatihan berkala kepada administrator.

Seluruh protokol keamanan dirancang merujuk pada praktik terbaik dari OWASP (*Open Web Application Security Project*) untuk mencegah serangan umum seperti SQL injection dan cross-site scripting.

3.3.2.4 Kinerja dan Skalabilitas

Sistem dirancang agar tetap efisien pada kondisi beban tinggi:

1. Sistem memanfaatkan prosesor multi-inti dan memori besar untuk mendukung pemrosesan paralel.
2. Perancangan arsitektur perangkat lunak mengikuti prinsip modular dan skalabel, sedangkan infrastruktur didukung oleh perangkat keras dengan antarmuka jaringan dan bus berkecepatan tinggi.

3. Optimalisasi kinerja dilakukan melalui pemanatan kode sumber (minifikasi), penyimpanan data sementara (*caching*), pemuatan lambat (*lazy loading*), serta penomoran halaman (*pagination*) untuk meningkatkan efisiensi waktu respons.

3.3.2.5 Pengalaman dan Antarmuka Pengguna (UX dan UI)

Desain visual mengacu pada prinsip Garrett (2022) dan Nielsen (2020), yang menekankan pentingnya konsistensi, efisiensi, serta kemudahan navigasi dalam antarmuka pengguna. Desain antarmuka dibuat agar mudah digunakan dan mendukung berbagai perangkat:

1. Desain responsif berbasis kerangka *Tailwind CSS* yang menyesuaikan tampilan dengan ukuran layar.
2. Elemen interaktif dengan waktu respons cepat melalui *React*.
3. Dukungan aksesibilitas pada mode ringan untuk koneksi lambat serta perangkat berspesifikasi rendah.
4. Struktur navigasi yang ringkas, termasuk proses pemesanan yang tidak rumit.
5. Desain visual konsisten, tampilan bersih, dan gambar produk berkualitas tinggi.

Dengan rancangan ini, situs web Azka Garden tidak hanya fungsional, tetapi juga mampu memberikan pengalaman belanja daring yang nyaman dan efisien.

3.3.3 Analisis Kebutuhan Infrastruktur

Analisis kebutuhan infrastruktur berfokus pada komponen teknis yang menjamin stabilitas sistem, kelancaran pengembangan, serta keberlanjutan operasional situs web perdagangan daring Azka Garden. Infrastruktur yang kuat mendukung skalabilitas, keamanan, dan efisiensi pemeliharaan.

3.3.3.1 Server Lokal dan Produksi

Penggunaan server dipisahkan antara lingkungan pengembangan dan produksi guna mencegah potensi gangguan operasional:

1. Server lokal menggunakan *Node.js* dan *Vite* untuk proses pengembangan dan pengujian awal secara tertutup.

2. Server produksi berbasis *Linux* dipilih untuk menjamin stabilitas, efisiensi, dan skalabilitas operasional ketika situs diakses melalui internet. Kebutuhan pengguna dan administrator yang telah diuraikan sebelumnya menjadi landasan dalam pemetaan fitur serta pengembangan antarmuka pada tahap perancangan dan implementasi berikutnya. Rancangan infrastruktur ini diharapkan menunjang pengembangan situs web Azka Garden secara stabil, aman, dan optimal untuk operasional skala kecil hingga menengah.

3.3.3.2 Manajemen Basis Data

Manajemen basis data dilakukan dengan efisien dan aman:

1. Sistem *PostgreSQL* digunakan untuk menyimpan informasi produk, pengguna, pesanan, dan pembayaran.
2. Akses basis data dilakukan melalui antarmuka *Supabase* untuk kemudahan pemeliharaan.

3.3.3.3 Pemeliharaan Versi dan Kolaborasi

Pemeliharaan kode dilakukan secara terpusat dengan pendekatan kolaboratif. *Git* digunakan sebagai sistem kendali versi terdistribusi yang mencatat setiap perubahan kode, memungkinkan kolaborasi tim yang efisien serta pemeliharaan riwayat versi secara terstruktur dan akurat. *GitHub* digunakan sebagai repositori daring yang menyediakan fitur pelacakan isu (issue tracking), penggabungan kode (pull request), serta mendukung proses integrasi dan penyebaran berkelanjutan (*Continuous Integration/Continuous Delivery* atau CI/CD). Dengan pendekatan ini, pengembangan sistem dapat berlangsung secara iteratif dan kolaboratif, serta memfasilitasi kontrol kualitas yang konsisten dan kesiapan sistem untuk skala produksi. Dalam implementasinya, CI/CD dikelola melalui *GitHub Actions* yang secara otomatis menjalankan pengujian unit dan integrasi setiap kali ada pembaruan kode, guna memastikan stabilitas sistem sebelum rilis ke server produksi. Repositori juga dilengkapi dengan dokumentasi pengembangan yang diperbarui secara berkala untuk menjaga konsistensi tim.

3.4 Tahap Perancangan

Tahap perancangan pada Azka Garden difokuskan pada arsitektur aplikasi berbasis komponen (*component based*) dan pemisahan tanggung jawab antar lapisan, mengantikan pendekatan *Model View Controller* (MVC) tradisional karena logika server dan *controller* eksplisit tidak dihosting di dalam repositori. Rancangan sistem diformulasikan melalui diagram UML, rancangan tampilan, dan struktur navigasi antarmuka pengguna. Arsitektur dibagi menjadi empat lapisan:

1. Lapisan Presentasi: Komponen *React* (katalog, keranjang, halaman produk *Stripe*, form checkout, manajer langganan).
2. Lapisan Layanan Klien: *StripeService*, *Supabase client*, helper format harga, *context* autentikasi dan keranjang.
3. Lapisan Integrasi Edge: *Supabase Edge Functions* (stripe-checkout, stripe-webhook) sebagai jembatan operasi sensitif (dengan fallback demo jika fungsi gagal).
4. Lapisan Data Terkelola: Tabel *stripe_customers*, *stripe_subscriptions*, *stripe_orders*, view *stripe_user_subscriptions* dan *stripe_user_orders* (jika belum ada migrasinya akan ditambahkan), serta tabel domain produk, orders internal, reviews, chat (direncanakan).

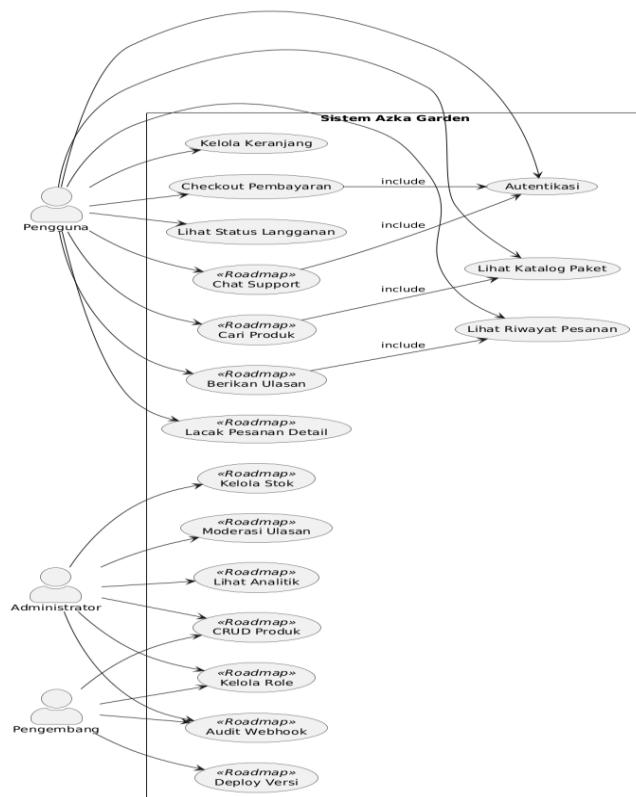
Tujuan perancangan adalah agar seluruh komponen sistem dapat dikembangkan secara modular, terdokumentasi dengan baik, serta mudah dipelihara dan diperluas di masa mendatang dengan pola modular yang memungkinkan penambahan fitur tanpa memodifikasi bagian inti pembayaran, dengan isolasi logika antara klien dan fungsi Edge.

3.4.1 Perancangan Diagram Kasus Penggunaan

Diagram kasus penggunaan (*Use Case*) digunakan untuk memodelkan interaksi antara aktor (pengguna, administrator, dan pengembang) dengan sistem. Dalam pengembangan sistem Azka Garden, relasi penyertaan (*include*) digunakan

untuk menggambarkan proses yang selalu dipanggil oleh proses utama, sedangkan relasi perluasan (*extend*) menunjukkan proses tambahan yang bersifat opsional tergantung pada kondisi tertentu.

Aktor utama yang direncanakan meliputi pengguna (*customer*), administrator (direncanakan), dan pengembang (direncanakan). Kasus penggunaan saat ini mencakup autentikasi, melihat katalog, mengelola keranjang, checkout pembayaran, dan melihat status langganan/pesanan (sebagian status langganan dapat menggunakan data mock sebelum sinkronisasi selesai). Roadmap mencakup CRUD produk, kelola stok, moderasi ulasan, chat support, analitik, *deploy* versi, audit *webhook*, dan kelola *role*.



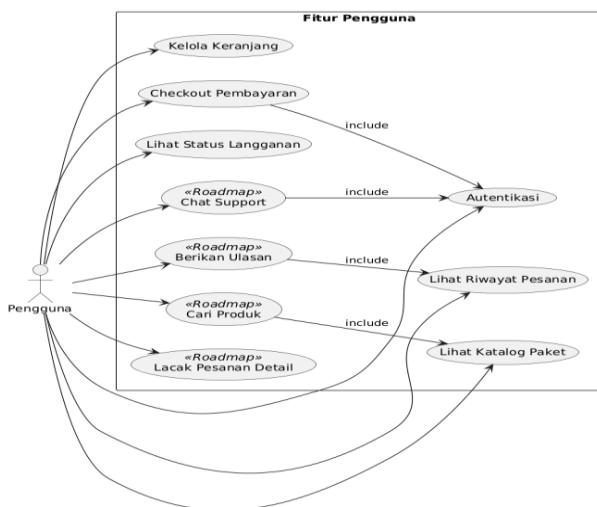
Gambar 3.1 Diagram Kasus Penggunaan Utama

Diagram kasus penggunaan utama web ini menggambarkan diagram induk yang melibatkan semua aktor utama dan fungsi sistem. Hubungan antara proses

utama dan proses tambahan digambarkan dengan garis asosiasi serta tanda *include* dan *extend*, sebagai kerangka kerja awal sistem sebelum perancangan teknis lebih lanjut.

3.4.1.1 Diagram Kasus Penggunaan Pengguna

Diagram kasus penggunaan untuk aktor pengguna menggambarkan semua interaksi dari perspektif pengguna, mulai dari login atau daftar, melihat katalog paket (*Stripe*), menambahkan ke keranjang, melakukan checkout, hingga melihat status langganan dan pesanan. Beberapa proses seperti "checkout pembayaran" bergantung pada proses autentikasi (*include*), sedangkan "ulasan" dan "chat support" merupakan fitur tambahan yang muncul setelah modul lanjutan tersedia (*extend*). Diagram ini menekankan bagaimana fitur-fitur utama seperti pemesanan dan pelacakan pesanan saling terhubung dan dapat diperluas dengan fitur tambahan seperti pencarian produk dan ulasan pengguna.

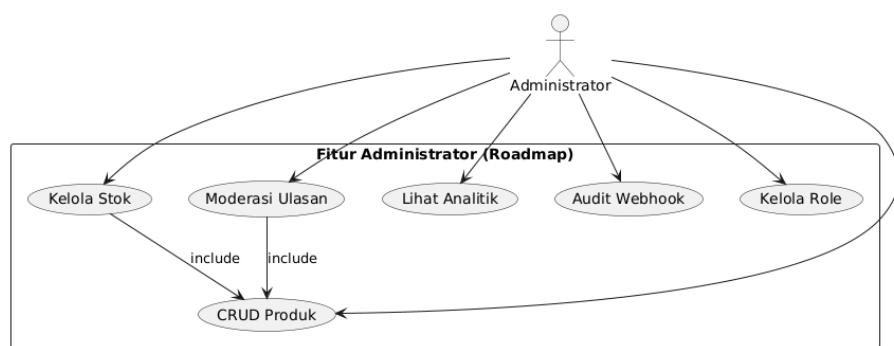


Gambar 3.2 Diagram Kasus Penggunaan untuk Aktor Pengguna

3.4.1.2 Diagram Kasus Penggunaan Administrator

Diagram kasus penggunaan untuk aktor administrator (direncanakan) mencakup aktivitas seperti mengelola data produk, memantau pesanan dan subscription, memeriksa log kegagalan *webhook*, mengelola stok dan kategori, serta

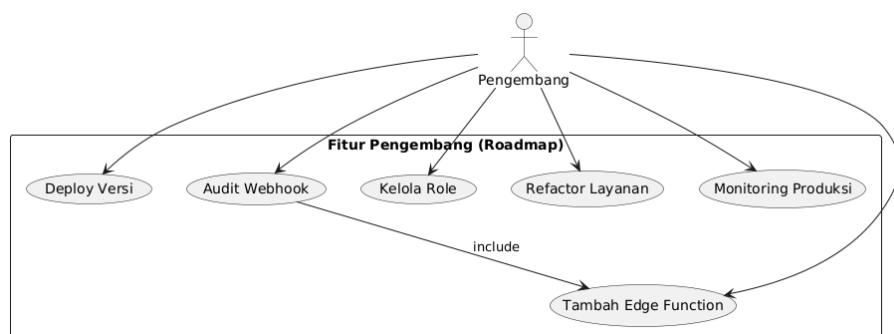
memoderasi ulasan. Beberapa proses menggunakan relasi include untuk ketergantungan logika sistem, seperti pengecekan status transaksi sebelum pengelolaan pesanan. Ilustrasi ini menjelaskan bahwa tanggung jawab administratif memiliki jalur proses yang saling terhubung dan memerlukan validasi kondisi tertentu sebelum aktivitas dapat dilanjutkan.



Gambar 3.3 Diagram Kasus Penggunaan untuk Aktor Administrator

3.4.1.3 Diagram Kasus Penggunaan Pengembang

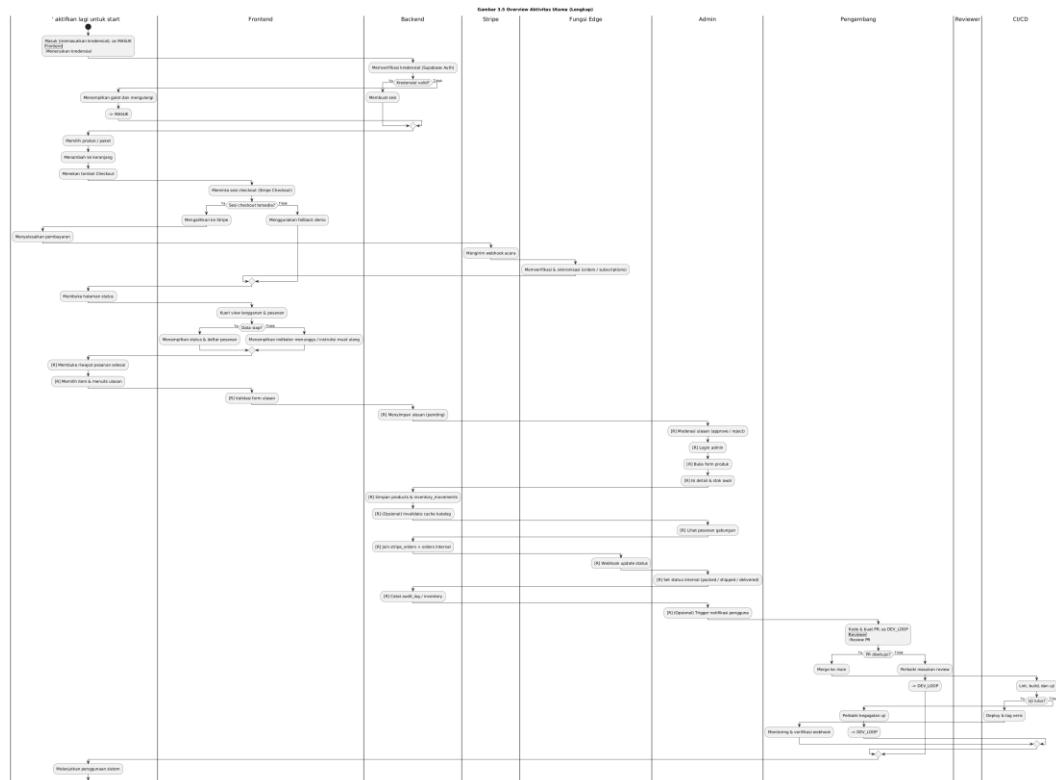
Diagram kasus penggunaan untuk aktor pengembang (direncanakan) menggambarkan peran pengembang dalam siklus teknis pengembangan, meliputi integrasi fitur baru, pemantauan log teknis, penambahan fungsi *Edge* baru, pengelolaan *sandbox Stripe*, audit keamanan, dan orkestrasi rilis (CI atau CD ketika pipeline telah dibuat). Proses ini menggambarkan kontribusi pengembang dalam menjaga kestabilan sistem serta menerapkan mekanisme pemulihan jika terjadi kegagalan implementasi.



Gambar 3.4. Diagram Kasus Penggunaan untuk Aktor Pengembang

3.4.2 Perancangan Diagram Aktivitas

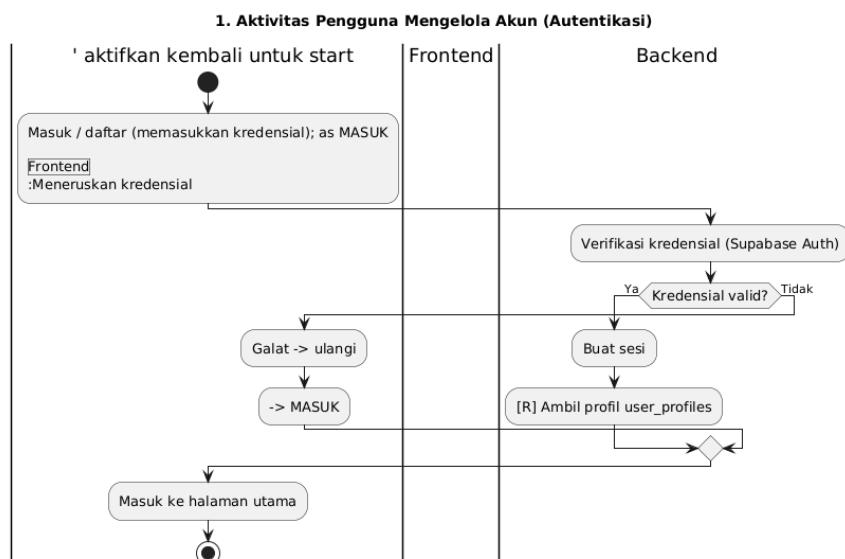
Diagram aktivitas digunakan untuk menggambarkan alur kerja dinamis dari setiap aktor dan memetakan alur proses lintas lapisan: interaksi pengguna di antarmuka, panggilan layanan klien, pemanggilan Edge Function, hingga penyimpanan data di tabel stripe_*. Diagram disusun berdasarkan jalur aktivitas (swimlane) untuk membedakan tanggung jawab pengguna, administrator, dan pengembang. Setiap proses digambarkan secara logis untuk mendukung pengujian serta perancangan teknis yang mendalam. Alur lanjutan (seperti ulasan dan chat) disiapkan sebagai perluasan dengan catatan khusus bahwa fallback demo pembayaran masih aktif, role dan RLS granular diterapkan penuh setelah user_profiles dibuat, dan pelacakan pesanan terbatas pada data *Stripe* yang tersinkronisasi.



Gambar 3.5. Diagram Aktivitas Utama

3.4.2.1 Diagram Aktivitas Pengguna Mengelola Akun

Diagram aktivitas pengguna mengelola akun mencakup masuk ke akun, verifikasi kredensial melalui *Supabase Auth*, dan pemeliharaan profil. Sistem memverifikasi validitas data sebelum memberikan akses ke halaman utama. Alur dimulai dari pengguna memasukkan kredensial, *Supabase Auth* melakukan verifikasi, jika gagal kembali ke form, jika sukses membuat sesi dan memuat profil (di masa depan dari tabel `user_profiles`). Diagram ini menunjukkan alur interaksi pengguna dengan modul akun, dimulai dari autentikasi hingga pembaruan data profil.

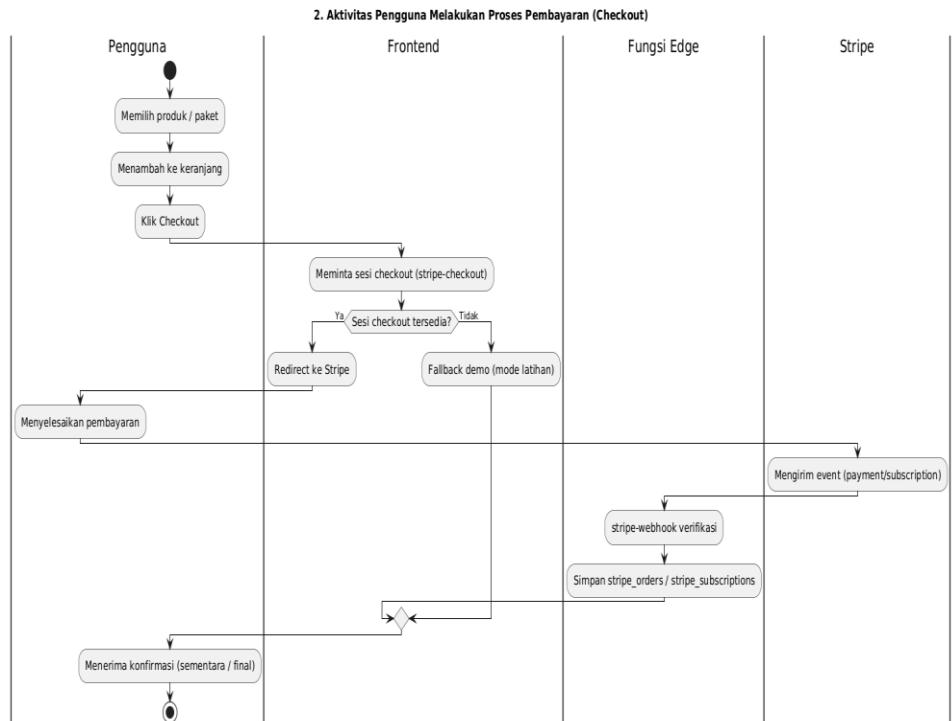


Gambar 3.6. Diagram Aktivitas Pengguna Mengelola Akun

3.4.2.2 Diagram Aktivitas Pengguna Melakukan Proses Pembayaran

Diagram aktivitas pengguna melakukan proses pembayaran menggambarkan pengguna memilih produk paket (*Stripe*), menambahkan ke keranjang, dan menyelesaikan pembayaran melalui Edge Function `stripe-checkout`. Jika Edge Function gagal, sistem menggunakan fallback demo. Setelah pembayaran sukses, *Stripe* mengirim event ke Edge Function `stripe-webhook` untuk sinkronisasi ke `stripe_orders` atau `stripe_subscriptions`. Diagram ini menekankan pentingnya

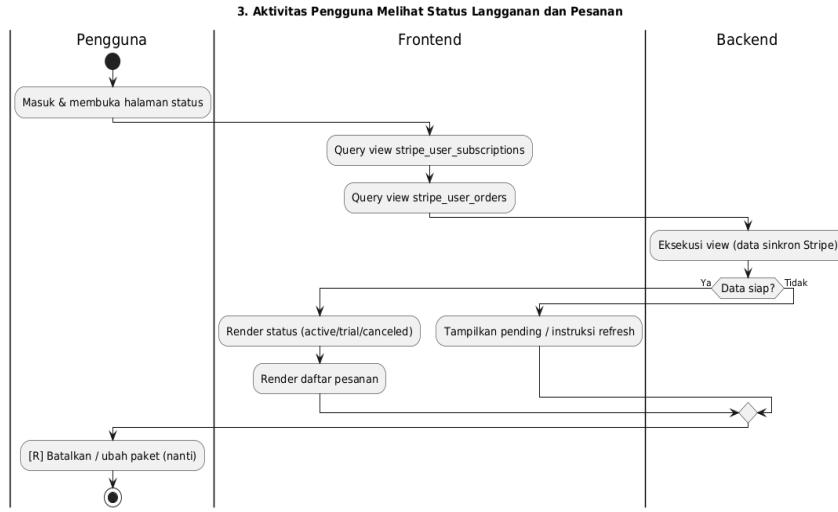
validasi akun sebagai syarat keamanan dan bagaimana sistem mengelola status pesanan secara otomatis setelah pembayaran berhasil.



Gambar 3.7. Diagram Aktivitas Pengguna Melakukan Proses Pembayaran

3.4.2.3 Diagram Aktivitas Pengguna Melihat Status Langganan dan Pesanan

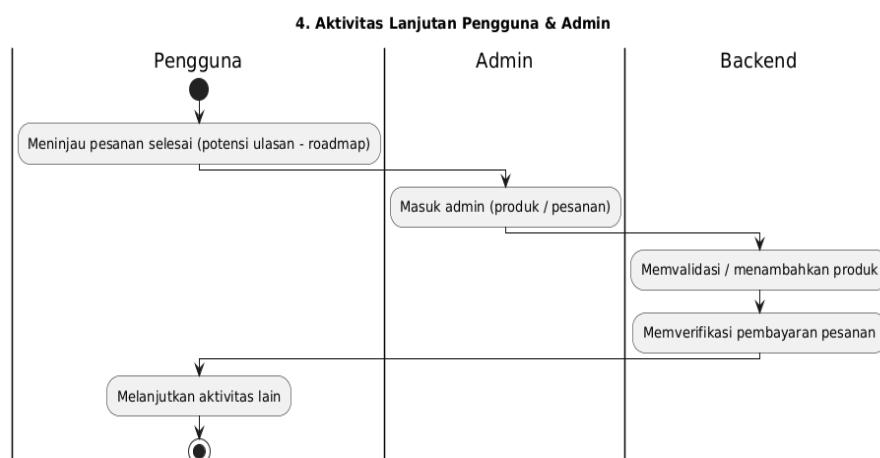
Diagram aktivitas pengguna melihat status langganan dan pesanan dimulai dari autentikasi, query view stripe_user_subscriptions dan stripe_user_orders. Jika data belum siap (webhook belum selesai), tampilkan pending atau instruksi refresh. Jika tersedia, tampilkan status (active, canceled, trial) dan daftar pesanan. (Roadmap) mencakup kemampuan untuk membatalkan langganan, mengubah paket, dan pelacakan pengiriman detail (memerlukan tabel orders internal dan kolom status logistik). Diagram ini mendemonstrasikan bagaimana sistem memberikan transparansi status dan memungkinkan pengguna memberikan umpan balik melalui fitur ulasan yang direncanakan.



Gambar 3.8. Diagram Aktivitas Pengguna Melihat Status Langganan dan Pesanan

3.4.2.4 Diagram Aktivitas Pengguna Memberi Ulasan (Direncanakan)

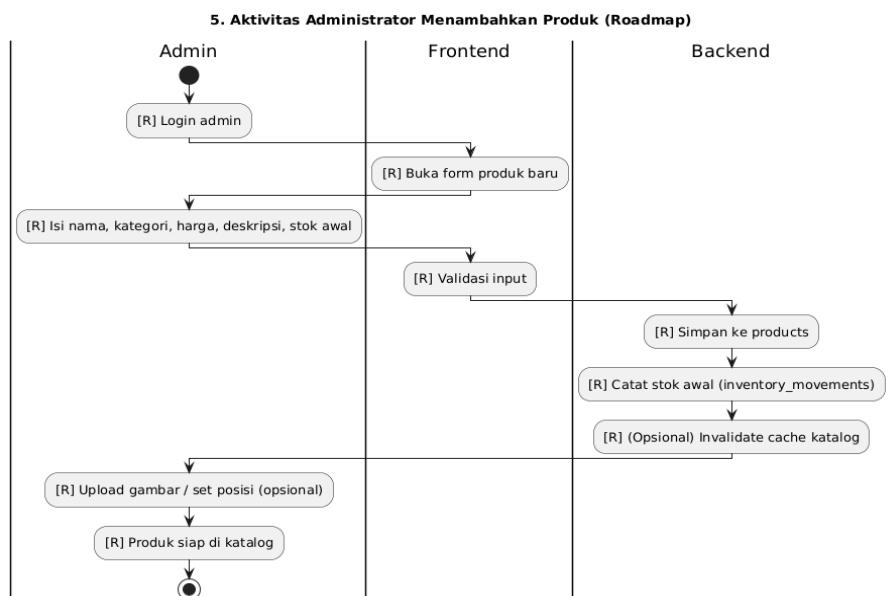
Diagram aktivitas pengguna memberi ulasan mencakup autentikasi, melihat daftar pesanan selesai, memilih pesanan, mengisi ulasan, dan menyimpan ke tabel reviews. (*Roadmap*) mencakup moderasi administrator dan tampilan di halaman produk. Proses ini memungkinkan pengguna memberikan *feedback* terhadap produk yang telah dibeli dan membantu pengguna lain dalam pengambilan keputusan pembelian.



Gambar 3.9. Diagram Aktivitas Pengguna Memberi Ulasan

3.4.2.5 Diagram Aktivitas Administrator Menambahkan Produk (Direncanakan)

Diagram aktivitas administrator menambahkan produk menunjukkan proses administratif yang sistematis dan terstruktur dalam menambah produk baru ke sistem. Alur mencakup login admin, pengisian form produk, validasi, penyimpanan ke tabel products, pengaturan stok awal, dan roadmap untuk invalidasi cache atau daftar katalog di klien. Proses ini memastikan bahwa hanya administrator yang berwenang dapat menambahkan produk dan semua data produk tervalidasi dengan benar.

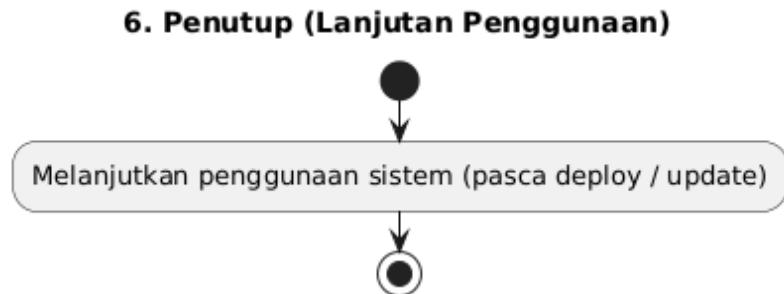


Gambar 3.10. Diagram Aktivitas Administrator Menambahkan Produk

3.4.2.6 Diagram Aktivitas Administrator Menangani Pesanan (Direncanakan)

Diagram aktivitas administrator menangani pesanan menampilkan keterkaitan antara aktivitas administratif dengan sistem otomatisasi validasi transaksi. Alur mencakup login admin, melihat gabungan pesanan (stripe_orders dan orders internal), menandai status internal (*packed*, *shipped*), dan roadmap untuk memperbarui notifikasi pengguna. Proses ini memungkinkan administrator untuk

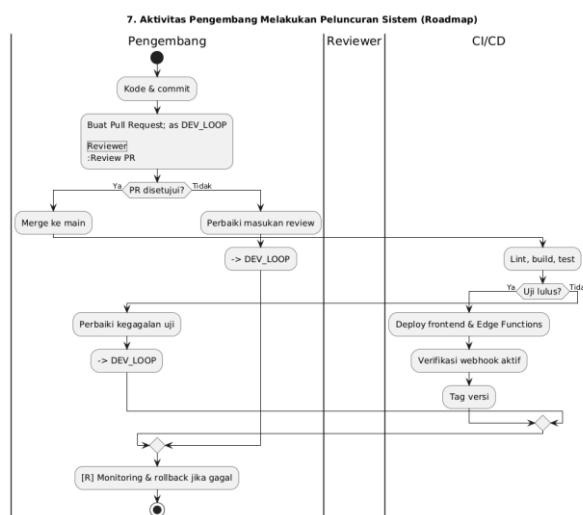
mengelola fulfillment pesanan dan memberikan update status kepada pelanggan.



Gambar 3.11. Diagram Aktivitas Administrator Menangani Pesanan

3.4.2.7 Diagram Aktivitas Pengembang Melakukan Peluncuran Sistem

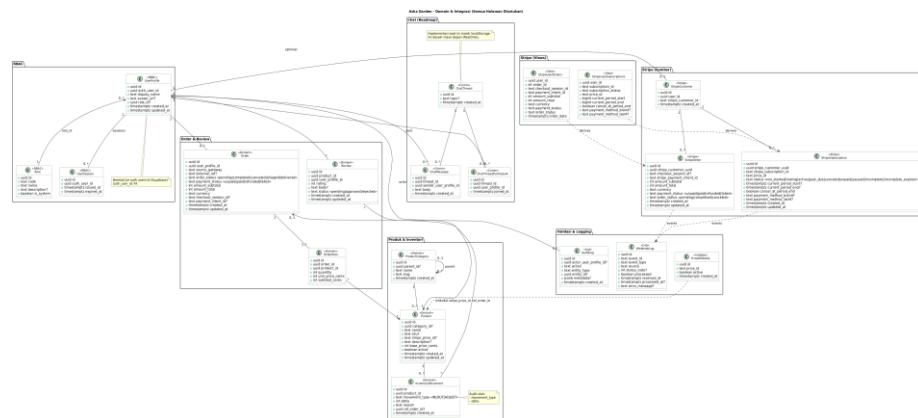
Diagram aktivitas pengembang melakukan peluncuran sistem menggambarkan seluruh rangkaian aktivitas tim pengembang dalam peluncuran sistem, dimulai dari koding, commit, pull request, review, merge, pipeline CI/CD (lint, build, test), deploy sisi antarmuka pengguna ke hosting statis, verifikasi webhook berfungsi, hingga tag versi. Proses ini dilakukan secara berkelanjutan melalui pendekatan CI/CD, guna memastikan bahwa setiap pembaruan kode dapat langsung diuji dan diimplementasikan tanpa mengganggu kestabilan sistem yang sedang berjalan.



Gambar 3.12. Diagram Aktivitas Pengembang Melakukan Peluncuran Sistem

3.4.3 Perancangan Diagram Kelas

Perancangan kelas menekankan pemisahan antara entitas bisnis (domain), entitas integrasi (*Stripe*), serta entitas pendukung (audit, keamanan, komunikasi). Karena saat ini sebagian data masih bersandar pada tabel *Stripe* (*stripe_**), diagram kelas memuat dua kategori: kelas TERSEDIA (existing) yang telah didukung oleh skema saat ini dan kelas RENCANA (roadmap) yang akan diimplementasikan setelah kebutuhan domain internal masuk tahap pembangunan.



Gambar 3.13 Diagram Kelas Utama

3.4.3.1 Tabel *UserProfile*

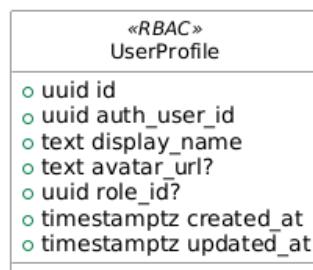
Tabel *UserProfile* menyimpan informasi pengguna yang terautentikasi dalam sistem. Tabel ini memiliki beberapa atribut penting, termasuk ID pengguna, ID pengguna autentikasi, nama tampilan, URL avatar, ID peran, serta timestamp untuk penciptaan dan pembaruan data. Struktur tabel ini dirancang untuk mendukung sistem kontrol akses berbasis peran (RBAC).

«RBAC»	
UserProfile	
o	uuid id
o	uuid auth_user_id
o	text display_name
o	text avatar_url?
o	uuid role_id?
o	timestamptz created_at
o	timestamptz updated_at

Gambar 3.14 Tabel *UserProfile*

3.4.3.2 Tabel *UserProfile* dengan Relasi

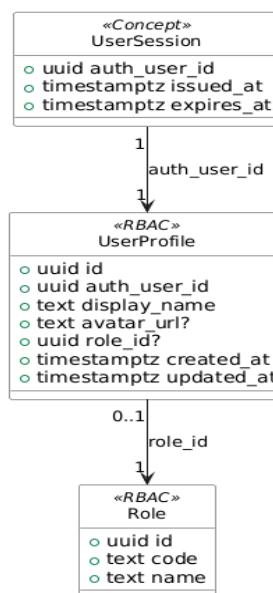
Tabel *UserProfile* memiliki relasi dengan tabel *Role* dan *UserSession*. Relasi ini menunjukkan bahwa setiap pengguna dapat memiliki satu peran dan satu sesi autentikasi. Hal ini mendukung pengelolaan hak akses pengguna dalam sistem.



Gambar 3.15 Tabel *UserProfile*

3.4.3.3 Tabel *UserProfile* dengan Relasi

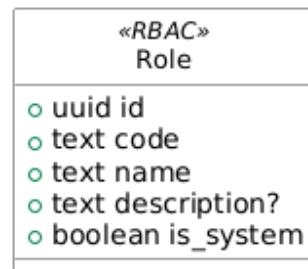
Tabel *UserProfile* memiliki relasi dengan tabel *Role* dan *UserSession*. Relasi ini menunjukkan bahwa setiap pengguna dapat memiliki satu peran dan satu sesi autentikasi. Hal ini mendukung pengelolaan hak akses pengguna dalam sistem.



Gambar 3.16 Tabel *UserProfile* dengan Relasi

3.4.3.4 Tabel *Role*

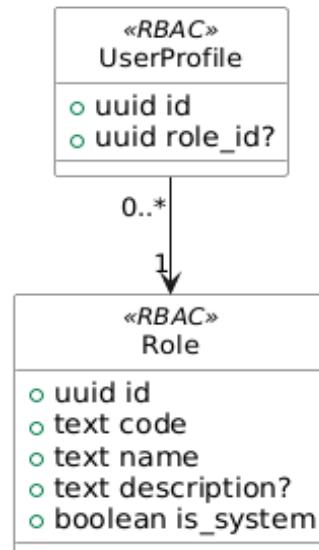
Tabel *Role* menyimpan informasi tentang peran yang dapat diberikan kepada pengguna. Setiap peran memiliki ID, kode, nama, deskripsi opsional, dan status sistem. Tabel ini penting untuk implementasi kontrol akses berbasis peran (RBAC).



Gambar 3.17 Tabel *Role*

3.4.3.5 Tabel *Role* dengan Relasi

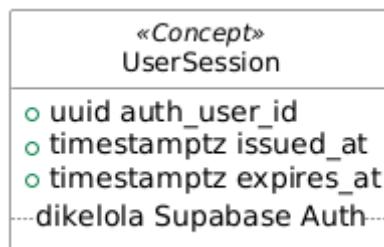
Tabel *Role* memiliki relasi dengan tabel *UserProfile*. Relasi ini menunjukkan bahwa setiap pengguna dapat memiliki satu peran yang ditetapkan, yang memungkinkan pengelolaan hak akses yang lebih terstruktur.



Gambar 3.18 Tabel *Role* dengan Relasi

3.4.3.6 Tabel *UserSession*

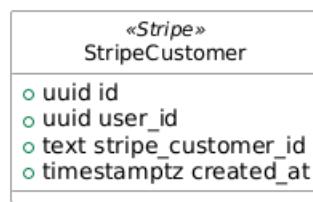
Tabel *UserSession* menyimpan informasi sesi autentikasi pengguna. Tabel ini mencakup ID pengguna, waktu penerbitan token, dan waktu kedaluwarsa token. Tabel ini dikelola oleh *Supabase Auth* untuk memastikan keamanan sesi pengguna.



Gambar 3.19 Tabel *UserSession*

3.4.3.7 Tabel *StripeCustomer*

Tabel *StripeCustomer* menyimpan informasi tentang pelanggan yang terdaftar di *Stripe*. Tabel ini mencakup ID unik untuk pelanggan, ID pengguna yang terhubung, ID pelanggan *Stripe*, dan timestamp untuk penciptaan data. Tabel ini penting untuk mengelola informasi pelanggan yang berinteraksi dengan sistem pembayaran.

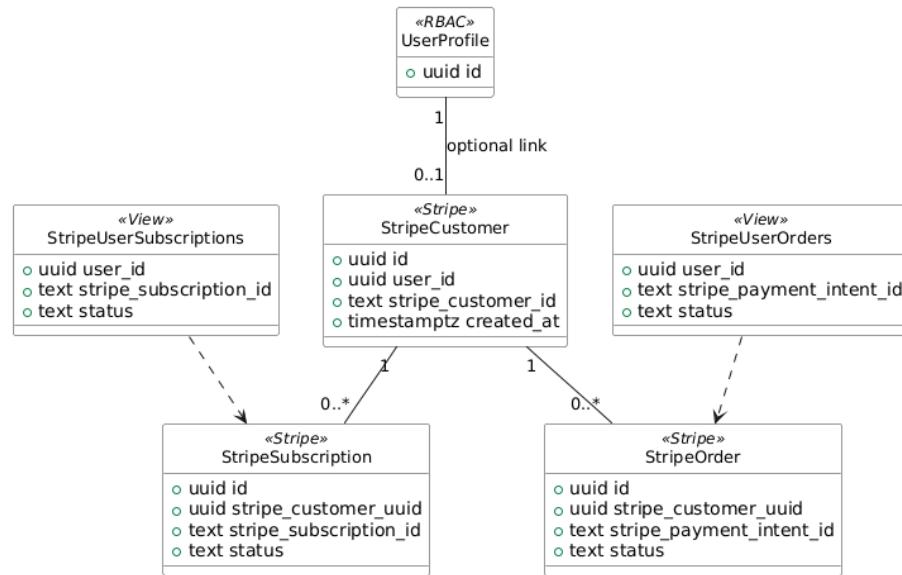


Gambar 3.20 Tabel *StripeCustomer*

3.4.3.8 Tabel *StripeCustomer* dengan Relasi

Tabel *StripeCustomer* memiliki relasi dengan tabel *StripeSubscription* dan *StripeOrder*. Relasi ini menunjukkan bahwa setiap pelanggan dapat memiliki beberapa langganan dan pesanan. Selain itu, terdapat

view *StripeUser Subscriptions* dan *StripeUser Orders* yang menyajikan informasi langganan dan pesanan berdasarkan pengguna.



Gambar 3.21 Tabel *StripeCustomer* dengan Relasi

3.4.3.9 Tabel *StripeSubscription*

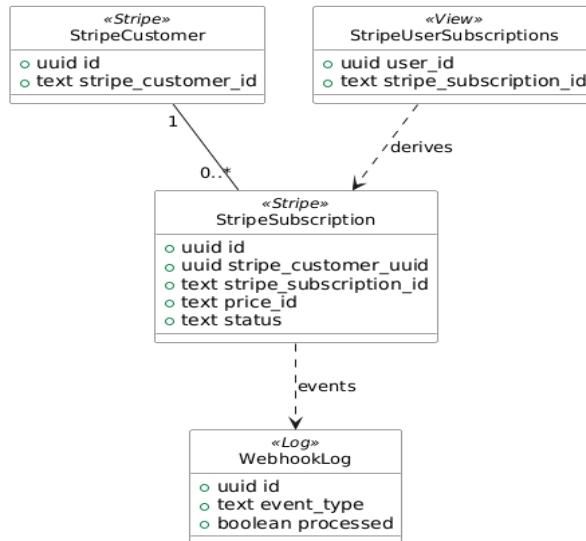
Tabel *StripeCustomer* memiliki relasi dengan tabel *StripeSubscription* dan *StripeOrder*. Relasi ini menunjukkan bahwa setiap pelanggan dapat memiliki beberapa langganan dan pesanan. Selain itu, terdapat view *StripeUser Subscriptions* dan *StripeUser Orders* yang menyajikan informasi langganan dan pesanan berdasarkan pengguna.

«Stripe»
StripeSubscription
o uuid id
o uuid stripe_customer_uuid
o text stripe_subscription_id
o text price_id
o text status
o date current_period_start?
o date current_period_end?
o boolean cancel_at_period_end
o timestamp created_at
o timestamp updated_at

Gambar 3.22 Tabel StripeSubscription

3.4.3.10 Tabel *StripeSubscription* dengan Relasi

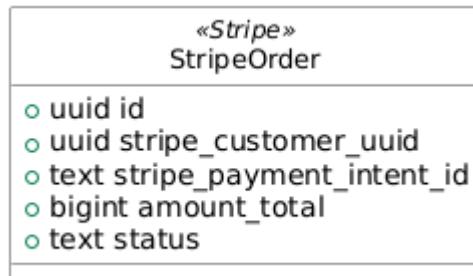
Tabel *StripeSubscription* memiliki relasi dengan tabel *StripeCustomer* dan view *StripeUser Subscriptions*. Relasi ini menunjukkan bahwa setiap langganan terkait dengan satu pelanggan dan dapat ditampilkan dalam view berdasarkan pengguna.



Gambar 3.23 Tabel *StripeSubscription*

3.4.3.11 Tabel *StripeOrder*

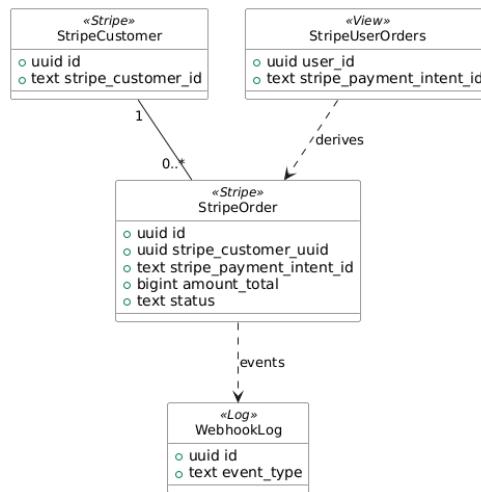
Tabel *StripeOrder* menyimpan informasi tentang pesanan yang dibuat oleh pelanggan di *Stripe*. Tabel ini mencakup ID unik untuk pesanan, ID pelanggan *Stripe*, ID intent pembayaran, dan status pesanan. Tabel ini penting untuk mengelola transaksi yang dilakukan oleh pelanggan.



Gambar 3.24 Tabel *StripeOrder*

3.4.3.12 Tabel *StripeOrder* dengan Relasi

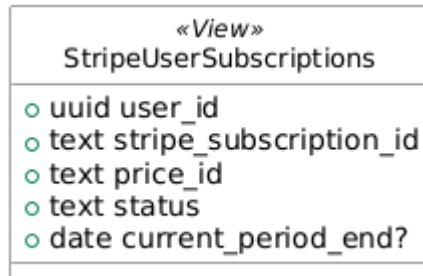
Tabel *StripeOrder* memiliki relasi dengan tabel *StripeCustomer* dan view *StripeUserOrders*. Relasi ini menunjukkan bahwa setiap pesanan terkait dengan satu pelanggan dan dapat ditampilkan dalam view berdasarkan pengguna.



Gambar 3.25 Tabel *StripeOrder* dengan Relasi

3.4.3.13 Tabel *StripeUser Subscriptions*

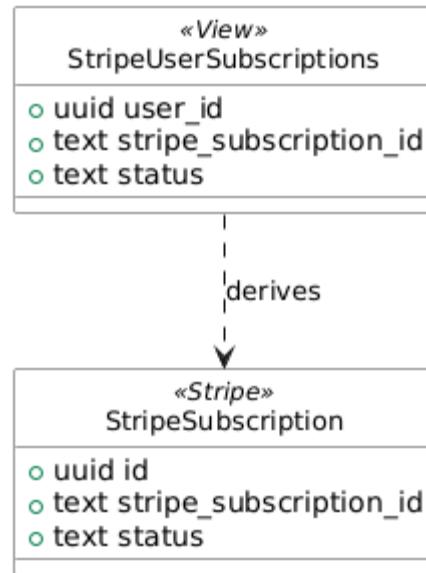
Tabel *StripeUser Subscriptions* adalah view yang menyajikan informasi langganan berdasarkan pengguna. Tabel ini mencakup ID pengguna, ID langganan *Stripe*, status langganan, dan tanggal akhir periode saat ini.



Gambar 3.26 Tabel *StripeUser Subscriptions*

3.4.3.14 Tabel *StripeUser Subscriptions* dengan Relasi

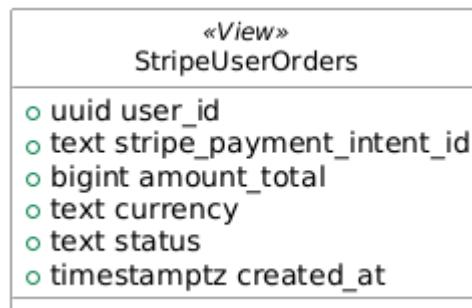
Tabel *StripeUser Subscriptions* memiliki relasi dengan tabel *StripeSubscription*. Relasi ini menunjukkan bahwa setiap langganan pengguna berasal dari langganan yang terdaftar di *Stripe*.



Gambar 3.27 Tabel *StripeUser Subscriptions* dengan Relasi

3.4.3.15 Tabel *StripeUserOrders*

Tabel *StripeUserOrders* adalah view yang menyajikan informasi pesanan berdasarkan pengguna. Tabel ini mencakup ID pengguna, ID intent pembayaran *Stripe*, total jumlah, mata uang, status pesanan, dan timestamp untuk penciptaan data.



Gambar 3.28 Tabel *StripeUserOrders*

3.4.3.16 Tabel *StripeUserOrders* dengan Relasi

Tabel *StripeUser Orders* memiliki relasi dengan tabel *StripeOrder*. Relasi ini menunjukkan bahwa setiap pesanan pengguna berasal dari pesanan yang terdaftar di *Stripe*.

«View» StripeUserOrders	
o	uuid user_id
o	text stripe_payment_intent_id
o	bigint amount_total
o	text currency
o	text status
o	timestamp created_at

Gambar 3.29 Tabel *StripeUserOrders* dengan Relasi

3.4.3.17 Tabel *ProductCategory*

Tabel *ProductCategory* menyimpan informasi tentang kategori produk yang ada dalam sistem. Tabel ini mencakup ID kategori, ID kategori induk (jika ada), nama kategori, slug untuk URL, dan timestamp untuk penciptaan data. Tabel ini penting untuk mengelompokkan produk berdasarkan kategori.

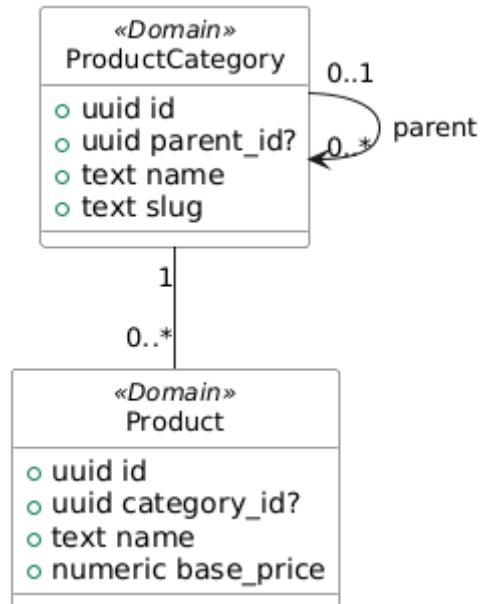
«Domain» ProductCategory	
o	uuid id
o	uuid parent_id?
o	text name
o	text slug
o	timestamp created_at

Gambar 3.30 Tabel *ProductCategory*

3.4.3.18 Tabel *ProductCategory* dengan Relasi

Tabel *ProductCategory* memiliki relasi dengan tabel *Product*. Relasi ini

menunjukkan bahwa setiap kategori dapat memiliki banyak produk, dan juga dapat memiliki kategori induk untuk membentuk hierarki kategori.



Gambar 3.31 Tabel *ProductCategory* dengan Relasi

3.4.3.19 Tabel *Product*

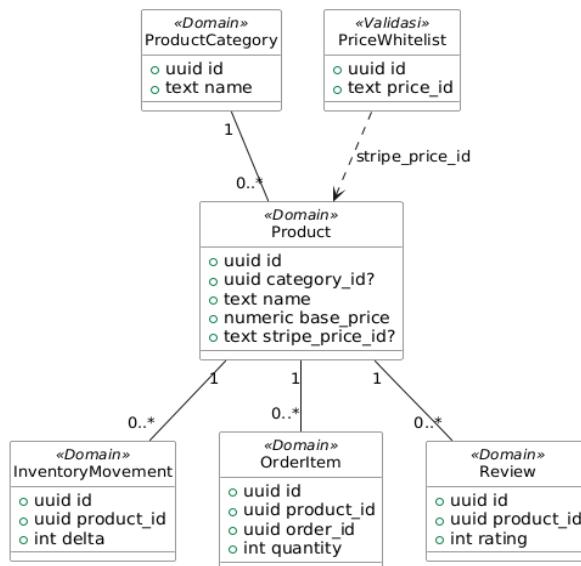
Tabel *Product* menyimpan informasi tentang produk yang tersedia dalam sistem. Tabel ini mencakup ID produk, ID kategori, nama produk, SKU, ID harga *Stripe*, deskripsi, harga dasar, status aktif, serta timestamp untuk penciptaan dan pembaruan data. Tabel ini penting untuk mengelola informasi produk yang ditawarkan kepada pelanggan.



Gambar 3.32 Tabel *Product*

3.4.3.20 Tabel *Product* dengan Relasi

Tabel *Product* memiliki relasi dengan tabel *ProductCategory*, *InventoryMovement*, *OrderItem*, dan *Review*. Relasi ini menunjukkan bahwa setiap produk dapat terkait dengan satu kategori, memiliki banyak pergerakan stok, item pesanan, dan ulasan.



Gambar 3.33 Tabel *Product* dengan Relasi

3.4.3.21 Tabel *InventoryMovement*

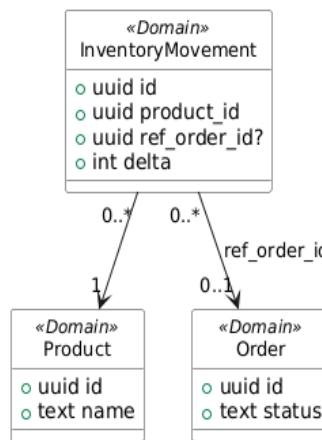
Tabel *InventoryMovement* menyimpan informasi tentang pergerakan stok produk. Tabel ini mencakup ID pergerakan, ID produk, perubahan jumlah stok (delta), alasan pergerakan, ID referensi pesanan (jika ada), dan timestamp untuk penciptaan data. Tabel ini penting untuk melacak perubahan stok produk.

«Domain» InventoryMovement	
o	uuid id
o	uuid product_id
o	int delta
o	text reason
o	uuid ref_order_id?
o	timestamp created_at

Gambar 3.34 Tabel *InventoryMovement*

3.4.3.22 Tabel *InventoryMovement* dengan Relasi

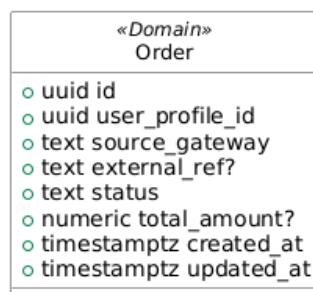
Tabel *InventoryMovement* memiliki relasi dengan tabel *Product* dan *Order*. Relasi ini menunjukkan bahwa setiap pergerakan stok terkait dengan satu produk dan dapat merujuk ke satu pesanan jika pergerakan tersebut disebabkan oleh transaksi.



Gambar 3.35 Tabel *InventoryMovement* dengan Relasi

3.4.3.23 Tabel *Order*

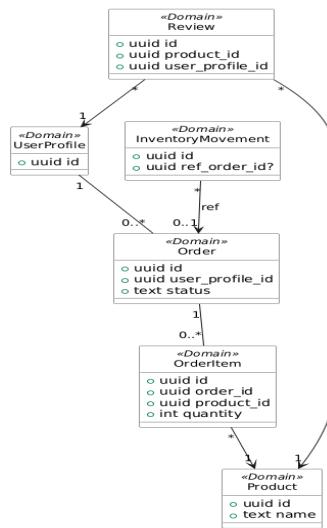
Tabel *Order* menyimpan informasi tentang pesanan yang dibuat oleh pengguna. Tabel ini mencakup ID pesanan, ID pengguna, sumber gateway pembayaran, referensi eksternal, status pesanan, total jumlah, serta timestamp untuk penciptaan dan pembaruan data. Tabel ini penting untuk mengelola transaksi yang dilakukan oleh pengguna.



Gambar 3.36 Tabel *Order*

3.4.3.24 Tabel *Order* dengan Relasi

Tabel *Order* memiliki relasi dengan tabel *User Profile*, *OrderItem*, *InventoryMovement*, dan *Review*. Relasi ini menunjukkan bahwa setiap pesanan terkait dengan satu pengguna, dapat memiliki banyak item pesanan, dan dapat memiliki pergerakan stok serta ulasan terkait.



Gambar 3.37 Tabel *Order* dengan Relasi

3.4.3.25 Tabel *OrderItem*

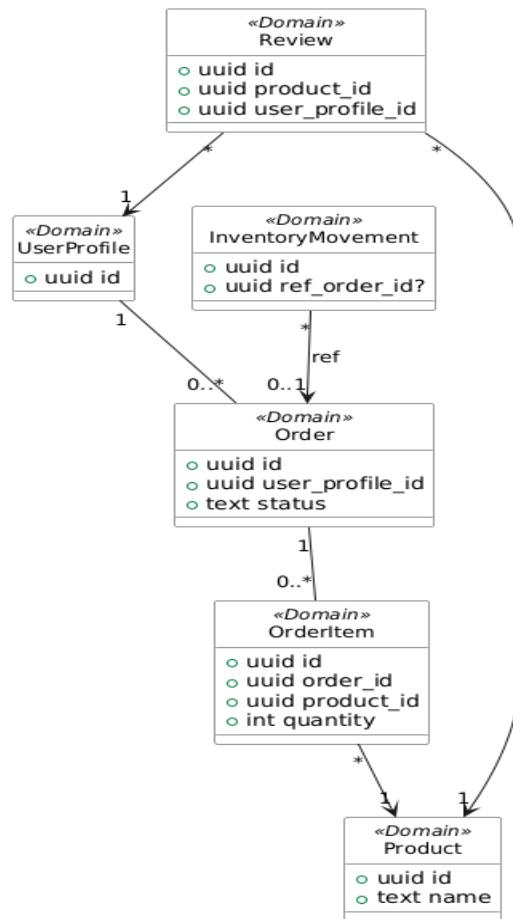
Tabel *OrderItem* menyimpan informasi tentang item yang termasuk dalam pesanan. Tabel ini mencakup ID item, ID pesanan, ID produk, jumlah item, harga satuan, dan subtotal. Tabel ini penting untuk mengelola rincian setiap item dalam pesanan.

«Domain» OrderItem	
o	uuid id
o	uuid order_id
o	uuid product_id
o	int quantity
o	numeric unit_price
o	numeric subtotal

Gambar 3.38 Tabel *OrderItem*

3.4.3.26 Tabel *OrderItem* dengan Relasi

Tabel *OrderItem* memiliki relasi dengan tabel *Order*, *Product*, dan *Review*. Relasi ini menunjukkan bahwa setiap item pesanan terkait dengan satu pesanan dan satu produk, serta dapat memiliki ulasan terkait.



Gambar 3.39 Tabel *OrderItem* dengan Relasi

3.4.3.27 Tabel *Review*

Tabel *Review* menyimpan informasi tentang ulasan yang diberikan oleh pengguna terhadap produk. Tabel ini mencakup ID ulasan, ID produk, ID pengguna, rating, isi ulasan, status, serta timestamp untuk penciptaan dan pembaruan data. Tabel ini penting untuk mengelola umpan balik dari pengguna

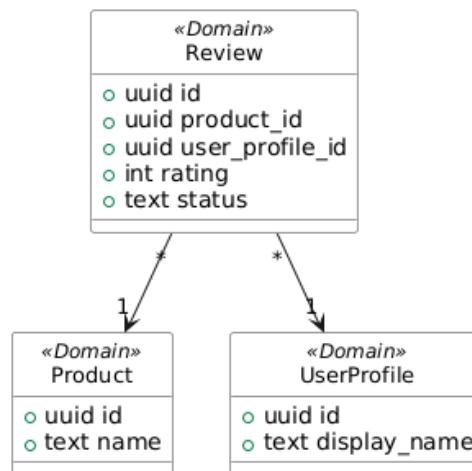
mengenai produk.

«Domain»	
Review	
o	uuid id
o	uuid product_id
o	uuid user_profile_id
o	int rating
o	text body?
o	text status
o	timestamptz created_at
o	timestamptz updated_at

Gambar 3.40 Tabel *Review*

3.4.3.28 Tabel *Review* dengan Relasi

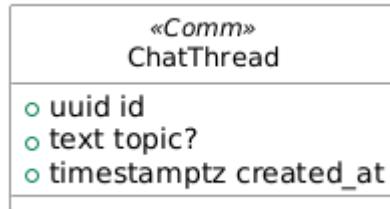
Tabel *Review* memiliki relasi dengan tabel *Product* dan *User Profile*. Relasi ini menunjukkan bahwa setiap ulasan terkait dengan satu produk dan satu pengguna yang memberikan ulasan.



Gambar 3.41 Tabel *Review* dengan Relasi

3.4.3.29 Tabel *ChatThread*

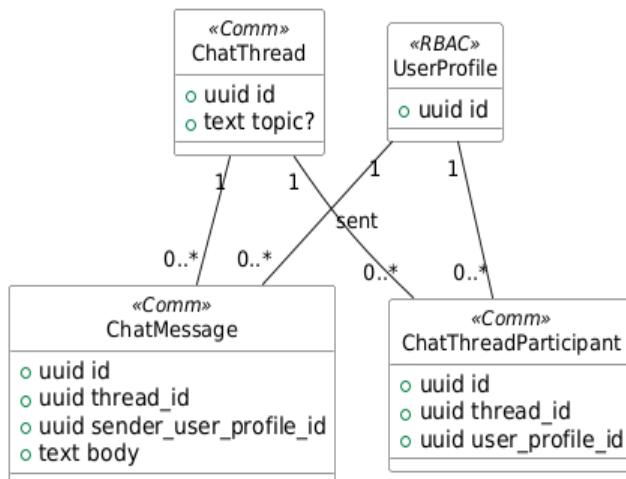
Tabel *ChatThread* menyimpan informasi tentang thread percakapan yang dibuat oleh pengguna. Tabel ini mencakup ID thread, topik percakapan, dan timestamp untuk penciptaan data. Tabel ini penting untuk mengelola diskusi antara pengguna.



Gambar 3.42 Tabel *ChatThread*

3.4.3.30 Tabel *ChatThread* dengan Relasi

Tabel *ChatThread* memiliki relasi dengan tabel *ChatMessage* dan *ChatThreadParticipant*. Relasi ini menunjukkan bahwa setiap thread dapat memiliki banyak pesan dan peserta yang terlibat dalam percakapan.



Gambar 3.43 Tabel *ChatThread* dengan Relasi

3.4.3.31 Tabel *ChatMessage*

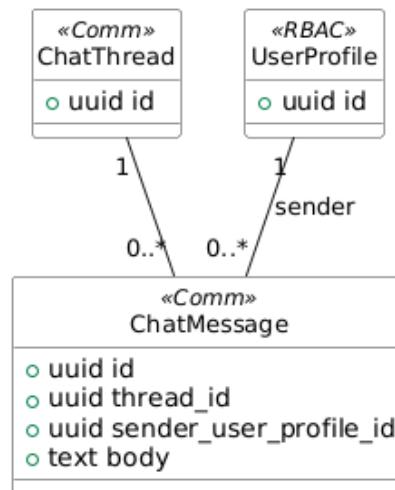
Tabel *ChatMessage* menyimpan informasi tentang pesan yang dikirim dalam thread percakapan. Tabel ini mencakup ID pesan, ID thread, ID pengguna pengirim, isi pesan, dan timestamp untuk penciptaan data. Tabel ini penting untuk mengelola komunikasi antar pengguna.

«Comm»
ChatMessage
o uuid id
o uuid thread_id
o uuid sender_user_profile_id
o text body
o timestamptz created_at

Gambar 3.44 Tabel *ChatMessage*

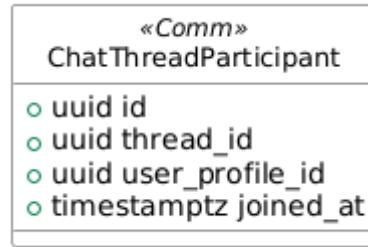
3.4.3.32 Tabel *ChatMessage* dengan Relasi

Tabel *ChatMessage* memiliki relasi dengan tabel *ChatThread* dan *User Profile*. Relasi ini menunjukkan bahwa setiap pesan terkait dengan satu thread dan satu pengguna yang mengirim pesan.

Gambar 3.45 Tabel *ChatMessage* dengan Relasi

3.4.3.33 Tabel *ChatThreadParticipant*

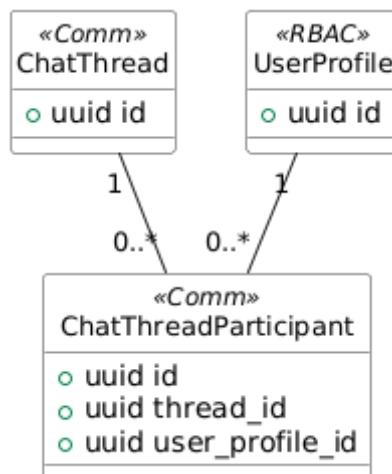
Tabel *ChatThreadParticipant* menyimpan informasi tentang peserta yang terlibat dalam thread percakapan. Tabel ini mencakup ID peserta, ID thread, ID pengguna, dan timestamp untuk saat bergabung. Tabel ini penting untuk mengelola siapa saja yang terlibat dalam diskusi.



Gambar 3.46 Tabel *ChatThreadParticipant*

3.4.3.34 Tabel *ChatThreadParticipant* dengan Relasi

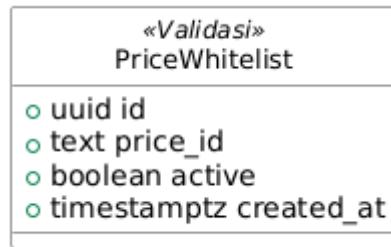
Tabel *ChatThreadParticipant* memiliki relasi dengan tabel *ChatThread* dan *User Profile*. Relasi ini menunjukkan bahwa setiap peserta terkait dengan satu thread dan satu pengguna yang berpartisipasi dalam diskusi.



Gambar 3.47 Tabel *ChatThreadParticipant* dengan Relasi

3.4.3.35 Tabel *PriceWhitelist*

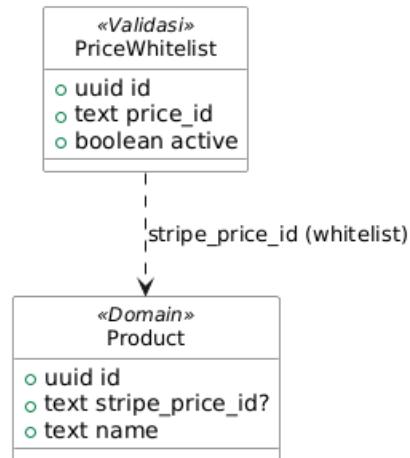
Tabel *PriceWhitelist* menyimpan informasi tentang harga yang diizinkan untuk produk tertentu. Tabel ini mencakup ID whitelist, ID harga, status aktif, dan timestamp untuk penciptaan data. Tabel ini penting untuk memastikan bahwa hanya harga yang valid yang dapat digunakan dalam transaksi.



Gambar 3.48 Tabel *PriceWhitelist*

3.4.3.36 Tabel *PriceWhitelist* dengan Relasi

Tabel *PriceWhitelist* memiliki relasi dengan tabel *Product*. Relasi ini menunjukkan bahwa setiap harga whitelist dapat terkait dengan satu produk melalui ID harga *Stripe*.



Gambar 3.47 Tabel *PriceWhitelist* dengan Relasi

3.4.3.37 Tabel *WebhookLog*

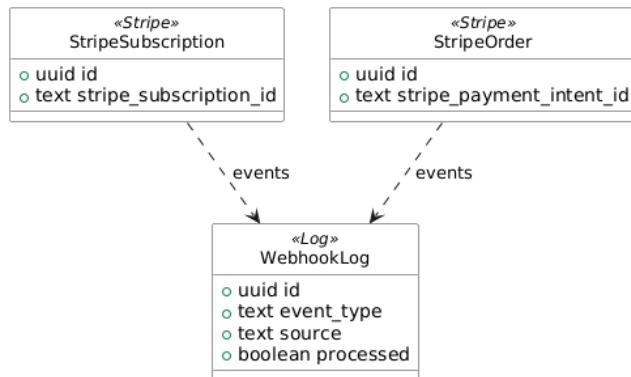
Tabel *WebhookLog* menyimpan informasi tentang log webhook yang diterima dari *Stripe*. Tabel ini mencakup ID log, ID event, jenis event, sumber, kode status, status pemrosesan, timestamp untuk penerimaan dan pemrosesan, serta pesan kesalahan jika ada. Tabel ini penting untuk melacak dan memverifikasi event yang diterima dari *Stripe*.

«Log» WebhookLog	
o	uuid id
o	text event_id
o	text event_type
o	text source
o	int status_code?
o	boolean processed
o	timestamptz received_at
o	timestamptz processed_at?
o	text error_message?

Gambar 3.46 Tabel *WebhookLog*

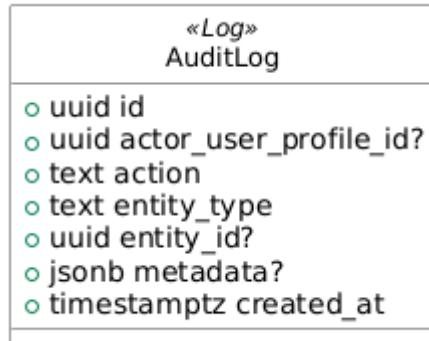
3.4.3.38 Tabel *WebhookLog* dengan Relasi

Tabel *WebhookLog* memiliki relasi dengan tabel *StripeSubscription* dan *StripeOrder*. Relasi ini menunjukkan bahwa setiap log *webhook* dapat terkait dengan *event* yang berasal dari langganan atau pesanan yang dibuat di *Stripe*.

Gambar 3.47 Tabel *WebhookLog* dengan Relasi

3.4.3.39 Tabel *AuditLog*

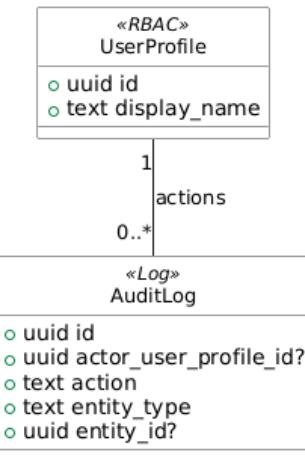
Tabel *AuditLog* menyimpan informasi tentang tindakan yang dilakukan oleh pengguna dalam sistem. Tabel ini mencakup ID log, ID pengguna yang melakukan tindakan, jenis tindakan, jenis entitas yang terlibat, ID entitas, metadata dalam format JSON, dan timestamp untuk penciptaan data. Tabel ini penting untuk melacak aktivitas pengguna dan audit keamanan.



Gambar 3.46 Tabel *AuditLog*

3.4.3.40 Tabel *AuditLog* dengan Relasi

Tabel *AuditLog* memiliki relasi dengan tabel *UserProfile*. Relasi ini menunjukkan bahwa setiap log audit terkait dengan satu pengguna yang melakukan tindakan dalam sistem.



Gambar 3.47 Tabel *AuditLog* dengan Relasi

3.4.4 Perancangan Struktur Tabel Nilai Basis Data

Dalam pengembangan sistem situs web perdagangan daring Azka Garden, perancangan struktur basis data merupakan aspek fundamental yang menentukan efisiensi, integritas, dan skalabilitas sistem. Basis data dirancang menggunakan

PostgreSQL dengan implementasi *cloud* melalui Supabase, yang menyediakan fitur waktu nyata dan keamanan tingkat *enterprise*.

3.4.4.1 Arsitektur Basis Data

Sistem Azka Garden menggunakan arsitektur basis data relasional dengan 15 (lima belas) tabel yang terdiri dari 8 tabel implementasi saat ini dan 7 tabel enhancement yang dirancang untuk mendukung fungsionalitas situs web perdagangan daring tanaman hias tingkat enterprise. Sistem ini mengintegrasikan multiple schema yaitu schema public untuk aplikasi utama, schema auth untuk manajemen autentikasi Supabase, dan schema pendukung lainnya (storage, realtime, vault).

Semua tabel menggunakan UUID (*Universally Unique Identifier*) sebagai primary key untuk memastikan keunikan data secara global dan mendukung sistem terdistribusi. Arsitektur ini mengakomodasi integrasi Stripe untuk payment processing, RBAC (*Role-Based Access Control*) untuk manajemen hak akses, dan audit logging untuk tracking aktivitas sistem.

3.4.4.2 Struktur Tabel Sistem

Struktur tabel sistem Azka Garden dirancang untuk mendukung operasional perdagangan daring tanaman hias yang komprehensif. Setiap tabel memiliki fungsi spesifik dalam mendukung proses bisnis mulai dari manajemen produk, pengelolaan pelanggan, hingga pemrosesan transaksi. Implementasi menggunakan pendekatan bertahap dengan prioritas pada tabel inti yang mendukung fungsi dasar sistem.

Tabel 3.1 Struktur Tabel Categories (Kategori Tanaman)

No.	Nama Field	Tipe Data	Ukuran	Constraint
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generate_v4()

2	name	varchar	100 chars	NOT NULL, UNIQUE
3	description	text	~65535 chars	NULL
4	image_url	varchar	500 chars	NULL
5	created_at	timestamptz	8 bytes	DEFAULT now()

Tabel categories berfungsi untuk mengklasifikasikan jenis-jenis tanaman hias berdasarkan karakteristik tertentu seperti tanaman indoor, outdoor, sukulen, dan tanaman berbunga.

Tabel 3.2 Struktur Tabel Users (Pelanggan)

No.	Nama Field	Tipe Data	Ukuran	Constraint
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generate_v4()
2	username	varchar	50 chars	NOT NULL, UNIQUE
3	email	varchar	320 chars	NOT NULL, UNIQUE
4	full_name	varchar	150 chars	NOT NULL
5	phone	varchar	20 chars	NULL
6	address	text	~1000 chars	NULL
7	city	varchar	100 chars	NULL
8	postal_code	varchar	10 chars	NULL
9	role_id	uuid	36 chars	FOREIGN KEY → roles(id)
10	password_hash	varchar	255 chars	NULL

11	email_verified	boolean	1 byte	DEFAULT false
12	phone_verified	boolean	1 byte	DEFAULT false
13	is_active	boolean	1 byte	DEFAULT true
14	created_at	timestamptz	8 bytes	DEFAULT now()
15	updated_at	timestamptz	8 bytes	DEFAULT now()

Tabel users menyimpan informasi lengkap pelanggan yang melakukan registrasi pada sistem Azka Garden, termasuk data kontak dan alamat pengiriman dengan fitur verifikasi email dan telepon.

Tabel 3.3 Struktur Tabel Products (Produk Tanaman)

No.	Nama Field	Tipe Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid genera te_v4()	Identifier unik produk
2	name	varchar	200 chars	NOT NULL	Nama produk tanaman
3	description	text	~65535 chars	NULL	Deskripsi lengkap produk
4	price	bigint	8 bytes	NOT NULL, CHECK (price >= 0)	Harga produk dalam rupiah
5	discount_price	bigint	8 bytes	DEFAULT 0, CHECK (discount_pr	Harga diskon (jika ada)

				$\text{ice} \geq 0$)	
6	stock	integer	4 bytes	DEFAULT 0, CHECK ($\text{stock} \geq 0$)	Jumlah stok tersedia
7	category_id	uuid	36 chars	NOT NULL, FOREIGN KEY → categories(i d)	Referensi ke tabel categories
8	image_url	varchar	500 chars	NULL	URL gambar produk
9	care_level	varchar	20 chars	CHECK (care_level IN ('Mudah', 'Sedang', 'Sulit'))	Tingkat perawatan tanaman
10	light_requir ement	varchar	100 chars	NULL	Kebutuhan cahaya
11	watering_fr equency	varchar	100 chars	NULL	Frekuensi penyiraman
12	plant_size	varchar	50 chars	NULL	Ukuran tanaman
13	pot_includ e_d	boolean	1 byte	DEFAULT false	Apakah termasuk pot
14	weight_gra ms	integer	4 bytes	NULL	Berat dalam gram
15	sku	varchar	50 chars	UNIQUE	Stock

					Keeping Unit
16	tags	text[]	Variable	DEFAULT '{}'	Tag pencarian produk
17	is_active	boolean	1 byte	DEFAULT true	Status aktif produk
18	created_at	timestamptz	8 bytes	DEFAULT now()	Waktu pembuatan record
19	updated_at	timestamptz	8 bytes	DEFAULT now()	Waktu update terakhir

Tabel products menyimpan informasi lengkap tentang produk tanaman hias yang dijual, termasuk spesifikasi perawatan yang diperlukan untuk setiap jenis tanaman dengan fitur SKU dan tag untuk pencarian yang lebih efektif.

Tabel 3.4 Struktur Tabel Products (Produk Tanaman)

No.	Nama Field	Tipe Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generat e_v4()	Identifier unik item cart
2	user_id	uuid	36 chars	NOT NULL, FOREIGN KEY → users(id)	Referensi ke tabel users
3	product_id	uuid	36 chars	NOT NULL, FOREIGN KEY → products(id)	Referensi ke tabel products

4	quantity	smallint	2 bytes	DEFAULT 1, CHECK (quantity > 0 AND quantity <= 999)	Jumlah item
5	added_at	timestamptz	8 bytes	DEFAULT now()	Waktu item ditambahkan

UNIQUE(user_id, product_id) yang memastikan satu *user* tidak dapat menambahkan produk yang sama lebih dari sekali. Tabel cart_items menyimpan item-item yang ditambahkan pelanggan ke keranjang belanja sebelum melakukan checkout.

Tabel 3.5 Struktur Tabel Orders (Pesanan)

No.	Nama Field	Tipe Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generate_v4()	Identifier unik pesanan
2	order_number	varchar	50 chars	NOT NULL, UNIQUE	Nomor pesanan unik
3	user_id	uuid	36 chars	NOT NULL, FOREIGN KEY → users(id)	Referensi ke tabel users
4	total_amount	bigint	8 bytes	NOT NULL, CHECK (total_amount >= 0)	Total harga produk
5	shipping_cost	bigint	8 bytes	DEFAULT 15000, CHECK (shipping_cost >= 0)	Biaya pengiriman
6	tax_amount	bigint	8 bytes	DEFAULT 0, CHECK	Jumlah pajak

				(tax_amount >= 0)	
7	discount_amount	bigint	8 bytes	DEFAULT 0, CHECK (discount_amount >= 0)	Jumlah diskon
8	final_amount	bigint	8 bytes	NOT NULL, CHECK (final_amount >= 0)	Total akhir yang dibayar
9	status	varchar	20 chars	DEFAULT 'pending', CHECK (status IN ('pending', 'confirmed', 'processing', 'shipped', 'delivered', 'cancelled'))	Status pesanan
10	payment_method	varchar	50 chars	NULL	Metode pembayaran
11	payment_status	varchar	20 chars	DEFAULT 'unpaid', CHECK (payment_status IN ('unpaid', 'paid', 'refunded', 'partially_refunded'))	Status pembayaran
12	shipping_address	text	~1000 chars	NOT NULL	Alamat pengiriman
13	billing_address	text	~1000 chars	NULL	Alamat tagihan
14	customer_notes	text	~1000 chars	NULL	Catatan dari pelanggan
15	admin_notes	text	~1000 chars	NULL	Catatan admin internal
16	tracking_number	varchar	100 chars	NULL	Nomor resi pengiriman

17	shipped_at	timestamptz	8 bytes	NULL	Waktu pengiriman
18	order_date	timestamptz	8 bytes	DEFAULT now()	Tanggal pemesanan
19	created_at	timestamptz	8 bytes	DEFAULT now()	Waktu pembuatan record
20	updated_at	timestamptz	8 bytes	DEFAULT now()	Waktu update terakhir

Tabel orders menyimpan informasi header pesanan yang dilakukan pelanggan, termasuk detail pembayaran, pengiriman, dan tracking dengan fitur catatan pelanggan dan admin.

Tabel 3.6 Struktur Tabel Order_Items (Detail Pesanan)

No.	Nama Field	Tipe Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generat e_v4()	Identifier unik item pesanan
2	order_id	uuid	36 chars	NOT NULL, FOREIGN KEY → orders(id)	Referensi ke tabel orders
3	product_id	uuid	36 chars	NOT NULL, FOREIGN KEY → products(id)	Referensi ke tabel products
4	product_na me	varchar	200 chars	NOT NULL	Nama produk saat pemesanan
5	quantity	smallint	2 bytes	NOT NULL, CHECK (quantity > 0 AND quantity <= 999)	Jumlah item dipesan

6	unit_price	bigint	8 bytes	NOT NULL, CHECK (unit_price >= 0)	Harga satuan saat pemesanan
7	total_price	bigint	8 bytes	NOT NULL, CHECK (total_price >= 0)	Total harga item

Tabel order_items menyimpan detail item-item yang dipesan dalam setiap transaksi, berfungsi sebagai snapshot harga pada saat pemesanan.

Tabel 3.7 Struktur Tabel Product_Reviews (Review Produk)

No.	Nama Field	Tipe Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generate_v4()	Identifier unik review
2	product_id	uuid	36 chars	NOT NULL, FOREIGN KEY → products(id)	Referensi ke tabel products
3	user_id	uuid	36 chars	NOT NULL, FOREIGN KEY → users(id)	Referensi ke tabel users
4	rating	smallint	2 bytes	NOT NULL, CHECK (rating BETWEEN 1 AND 5)	Rating 1-5 bintang
5	review_text	text	~65535 chars	NULL	Teks review pelanggan
6	status	varchar	20 chars	DEFAULT 'pending', CHECK	Status persetujuan review

				(status IN ('pending', 'approved', 'rejected'))	
7	created_at	timestamptz	8 bytes	DEFAULT now()	Waktu review dibuat

Tabel product_reviews menyimpan ulasan dan rating yang diberikan pelanggan terhadap produk yang telah dibeli, termasuk status persetujuan review.

Tabel 3.8 Struktur Tabel Wishlist (Daftar Keinginan)

No.	Nama Field	Tipe Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generate_v4()	Identifier unik wishlist
2	user_id	uuid	36 chars	NOT NULL, FOREIGN KEY → users(id)	Referensi ke tabel users
3	product_id	uuid	36 chars	NOT NULL, FOREIGN KEY → products(id)	Referensi ke tabel products
4	added_at	timestamptz	8 bytes	DEFAULT now()	Waktu ditambahkan

UNIQUE(user_id, product_id) yang memastikan satu user tidak dapat menambahkan produk yang sama ke wishlist lebih dari sekali. Tabel wishlist menyimpan daftar produk yang diinginkan pelanggan untuk dibeli di kemudian hari. Berdasarkan analisis diagram kelas PlantUML dan kebutuhan sistem enterprise, diperlukan tabel tambahan untuk mendukung fitur *advanced*:

Tabel 3.9 Struktur Tabel Roles (Manajemen Role)

No.	Nama Field	Tipe Data	Ukuran	Constraint	Keterangan
-----	------------	-----------	--------	------------	------------

1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generate_v4()	Identifier unik role
2	code	varchar	50 chars	NOT NULL, UNIQUE, CHECK (char_length (code) >= 3)	Kode role (admin, customer, dll.)
3	name	varchar	100 chars	NOT NULL, CHECK (char_length (name) >= 3)	Nama role
4	description	text	~1000 chars	NULL	Deskripsi role
5	is_system	boolean	1 byte	DEFAULT false	Role sistem tidak dapat dihapus
6	permissions	jsonb	Variable	DEFAULT '[]':jsonb	Daftar permission dalam format JSON
7	created_at	timestamptz	8 bytes	DEFAULT now()	Waktu pembuatan
8	updated_at	timestamptz	8 bytes	DEFAULT now()	Waktu update terakhir

Tabel roles mengelola sistem peran dan hak akses pengguna dalam aplikasi Azka Garden.

Tabel 3.10 Struktur Tabel User_Profiles (Profil Extended)

No.	Nama Field	Tipe Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generate_v4()	Identifier unik profil

2	auth_user_id	uuid	36 chars	NOT NULL, UNIQUE, FOREIGN KEY → auth.users(id)	Link ke Supabase Auth
3	user_id	uuid	36 chars	NOT NULL, UNIQUE, FOREIGN KEY → users(id)	Link ke tabel users
4	role_id	uuid	36 chars	FOREIGN KEY → roles(id)	Role pengguna
5	display_name	varchar	100 chars	NULL	Nama tampilan
6	avatar_url	varchar	500 chars	NULL	URL foto profil
7	is_active	boolean	1 byte	DEFAULT true	Status aktif user
8	last_login_at	timestamptz	8 bytes	NULL	Waktu login terakhir
9	created_at	timestamptz	8 bytes	DEFAULT now()	Waktu pembuatan
10	updated_at	timestamptz	8 bytes	DEFAULT now()	Waktu update

Tabel user_profiles menghubungkan user aplikasi dengan sistem autentikasi Supabase dan mengelola manajemen hak akses berbasis peran.

Tabel 3.11 Struktur Tabel User_Sessions (Session Tracking)

No.	Nama Field	Tipe Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_gen erate_v4()	Identifier unik session
2	auth_user_id	uuid	36 chars	NOT NULL, FOREIGN	Link ke Supabase Auth

				KEY → auth.users(id)	
3	user_profile_id	uuid	36 chars	NOT NULL, FOREIGN KEY → user_profiles(id)	Link ke user profile
4	session_token	varchar	255 chars	NOT NULL, UNIQUE	Token session unik
5	user_agent	text	~500 chars	NULL	Informasi browser/device
6	ip_address	inet	16 bytes	NULL	Alamat IP pengguna
7	is_active	boolean	1 byte	DEFAULT true	Status aktif session
8	issued_at	timestamptz	8 bytes	DEFAULT now()	Waktu session dibuat
9	expires_at	timestamptz	8 bytes	NOT NULL	Waktu kadaluwarsa
10	last_activity_at	timestamptz	8 bytes	DEFAULT now()	Aktivitas terakhir

Tabel user_sessions melacak session aktif pengguna untuk keamanan dan *monitoring* aktivitas.

Tabel 3.12 Struktur Tabel Inventory_Movements (Pergerakan Stok)

No.	Nama Field	Tipe Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generate_v4()	Identifier unik movement
2	product_id	uuid	36 chars	NOT NULL, FOREIGN KEY → products(id)	Produk yang diubah stoknya

3	movement_type	varchar	10 chars	CHECK (movement_type IN ('IN','OUT','ADJUST'))	Jenis pergerakan stok
4	delta	integer	4 bytes	NOT NULL	Perubahan stok (+/-)
5	stock_before	integer	4 bytes	CHECK (stock_before >= 0)	Stok sebelum perubahan
6	stock_after	integer	4 bytes	CHECK (stock_after >= 0)	Stok setelah perubahan
7	reason	varchar	255 chars	NOT NULL	Alasan pergerakan
8	ref_order_id	uuid	36 chars	FOREIGN KEY → orders(id)	Referensi order (optional)
9	notes	text	~1000 chars	NULL	Catatan tambahan
10	created_by	uuid	36 chars	FOREIGN KEY → user_profiles(id)	User yang mencatat
11	created_at	timestamptz	8 bytes	DEFAULT now()	Waktu pencatatan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generate_v4()	Identifier unik session

Tabel inventory_movements melacak semua pergerakan stok produk untuk audit dan analisis inventory.

Tabel 3.13 Struktur Tabel Stripe_Customers (Customer Stripe)

No.	Nama Field	Tipe Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generate_v4()	Identifier unik

2	user_id	uuid	36 chars	NOT NULL, UNIQUE, FOREIGN KEY → users(id)	Link ke user
3	stripe_customer_id	varchar	255 chars	NOT NULL, UNIQUE	Stripe Customer ID
4	default_payment_method	varchar	255 chars	NULL	Metode pembayaran default
5	currency	varchar	3 chars	DEFAULT 'IDR', CHECK (char_length (currency) = 3)	Preferensi mata uang
6	is_active	boolean	1 byte	DEFAULT true	Status aktif customer
7	metadata	jsonb	Variable	DEFAULT '{}'::jsonb	Metadata tambahan dari Stripe
8	created_at	timestamptz	8 bytes	DEFAULT now()	Waktu pembuatan
9	updated_at	timestamptz	8 bytes	DEFAULT now()	Waktu update terakhir

Tabel stripe_customers mengelola integrasi customer dengan sistem pembayaran Stripe.

Tabel 3.14 Struktur Tabel Stripe_Orders (Order Tracking Stripe)

No.	Nama Field	Tipe Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_genera te_v4()	Identifier unik
2	order_id	uuid	36 chars	NOT NULL, UNIQUE, FOREIGN	Link ke order

				KEY → orders(id)	
3	stripe_customer_id	uuid	36 chars	NOT NULL, FOREIGN KEY → stripe_customers(id)	Link ke customer Stripe
4	checkout_session_id	varchar	255 chars	UNIQUE	Stripe Checkout Session ID
5	payment_intent_id	varchar	255 chars	UNIQUE	Stripe Payment Intent ID
6	amount_subtotal	bigint	8 bytes	NOT NULL, CHECK (amount_subtotal >= 0)	Subtotal dalam cents
7	amount_total	bigint	8 bytes	NOT NULL, CHECK (amount_total >= 0)	Total dalam cents
8	currency	varchar	3 chars	DEFAULT 'IDR', CHECK (char_length(currency) = 3)	Mata uang
9	payment_status	varchar	20 chars	DEFAULT 'unpaid', CHECK (payment_status IN ('unpaid','paid','refunded','failed'))	Status pembayaran
10	order_status	varchar	20 chars	DEFAULT 'pending', CHECK (order_status IN ('pending','c'))	Status order

				ompleted','c anceled'))	
11	stripe_meta data	jsonb	Variable	DEFAULT '{}':jsonb	Metadata dari Stripe
12	created_at	timestamptz	8 bytes	DEFAULT now()	Waktu pembuatan
13	updated_at	timestamptz	8 bytes	DEFAULT now()	Waktu update

Tabel stripe_orders melacak semua transaksi yang diproses melalui sistem pembayaran Stripe.

Tabel 3.15 Struktur Tabel Audit_Logs (Sistem Audit)

No.	Nama Field	Tipe Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_genera te_v4()	Identifier unik log audit
2	actor_user_i d	uuid	36 chars	FOREIGN KEY → user_profile s(id)	User yang melakukan aksi
3	action	varchar	100 chars	NOT NULL	Jenis aksi (CREATE, UPDATE, DELETE)
4	entity_type	varchar	100 chars	NOT NULL	Tipe entitas (users, products, orders)
5	entity_id	uuid	36 chars	NULL	ID entitas yang diakses
6	old_values	jsonb	Variable	NULL	Nilai sebelum perubahan
7	new_values	jsonb	Variable	NULL	Nilai setelah perubahan
8	metadata	jsonb	Variable	DEFAULT '{}':jsonb	Data tambahan
9	ip_address	inet	16 bytes	NULL	Alamat IP pengguna

10	user_agent	text	~500 chars	NULL	Informasi browser/device
11	success	boolean	1 byte	DEFAULT true	Status keberhasilan aksi
12	error_message	text	~1000 chars	NULL	Pesan error (jika gagal)
13	created_at	timestamptz	8 bytes	DEFAULT now()	Waktu aksi

Tabel audit_logs mencatat semua aktivitas penting dalam sistem untuk keperluan audit, keamanan, dan kepatuhan.

3.4.4.3 Relasi Antar Tabel Sistem Azka Garden

Struktur basis data Azka Garden mengimplementasikan relasi antar tabel dengan menggunakan foreign key constraints untuk memastikan integritas referensial. Setiap relasi dirancang untuk mendukung fungsionalitas perdagangan daring yang optimal dengan total 19 relasi yang mencakup semua tabel inti dalam sistem.

Tabel 3.16 Struktur Tabel Relasi Antar Tabel Sistem Azka Garden

No.	Tabel Child	Foreign Key	Tabel Parent	R	Kardinalitas	C	Keterangan
1	audit_logs	actor_user_id	user_profiles	id	1 : N	ON DELETE SET NULL	Tracking aktivitas user
2	cart_items	product_id	products	id	1 : N	ON DELETE CASCADE	Satu produk dapat ditambahkan ke banyak keranjang
3	cart_items	user_id	users	id	1 : N	ON DELETE CASCADE	Satu user dapat memiliki banyak item di keranjang

4	inventory_movements	created_by	user_profiles	id	1 : N	ON DELETE SET NULL	User yang mencatat movement
5	inventory_movements	product_id	products	id	1 : N	ON DELETE RESTRICT	Tracking pergerakan stok produk
6	inventory_movements	ref_order_id	orders	id	1 : N	ON DELETE SET NULL	Referensi order untuk movement
7	order_items	order_id	orders	id	1 : N	ON DELETE TE CASCADE	Satu pesanan dapat berisi banyak item produk
8	order_items	product_id	products	id	1 : N	ON DELETE RESTRICT	Satu produk dapat dipesan berkali-kali
9	orders	user_id	users	id	1 : N	ON DELETE RESTRICT	Satu user dapat memiliki banyak riwayat pesanan
10	product_reviews	product_id	products	id	1 : N	ON DELETE TE CASCADE	Satu produk dapat memiliki banyak review
11	product_reviews	user_id	users	id	1 : N	ON DELETE TE CASCADE	Satu user dapat memberikan review untuk banyak produk
12	products	category_id	categories	id	1 : N	ON DELETE	Satu kategori dapat memiliki

						REST RICT	banyak produk tanaman
13	stripe_customers	user_id	users	id	1 : 1	ON DELETE TE CASCADE	Integrasi customer dengan Stripe
14	stripe_orders	order_id	orders	id	1 : 1	ON DELETE TE CASCADE	Tracking order Stripe
15	stripe_orders	stripe_customer_id	stripe_customers	id	1 : N	ON DELETE TE RESTRICT	Link ke Stripe customer
16	user_profiles	role_id	roles	id	1 : N	ON DELETE TE SET NULL	Satu role dapat dimiliki banyak user profile
17	user_profiles	user_id	users	id	1 : 1	ON DELETE TE CASCADE	Satu user memiliki satu profil extended
18	user_sessions	user_profile_id	user_profiles	id	1 : N	ON DELETE TE CASCADE	Tracking session user profile
19	users	role_id	roles	id	1 : N	ON DELETE TE RESTRICT	Satu role dapat dimiliki banyak user
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_gen	Identifier unique log		

				generate_v4()	audit		
--	--	--	--	---------------	-------	--	--

3.4.4.4 Unique Constraints dan Composite Keys

Unique constraints dan composite keys digunakan untuk memastikan integritas data dan mencegah duplikasi yang tidak diinginkan dalam sistem. Implementasi ini mencakup 17 constraint unik yang melindungi data kritis dari duplikasi.

Tabel 3.17 Struktur Tabel Unique Constraints dan Composite Keys

No.	Tabel	Kolom	Tipe Constraint	Tujuan
1	categories	name	UNIQUE	Mencegah duplikasi nama kategori
2	users	username	UNIQUE	Memastikan username unik untuk login
3	users	email	UNIQUE	Mencegah registrasi email ganda
4	products	sku	UNIQUE	Memastikan SKU produk unik
5	orders	order_number	UNIQUE	Memastikan nomor pesanan unik
6	cart_items	(user_id, product_id)	UNIQUE COMPOSITE	Mencegah produk duplikat dalam keranjang user
7	product_reviews	(product_id, user_id)	UNIQUE COMPOSITE	Membatasi satu review per user per produk
8	wishlist	(user_id, product_id)	UNIQUE COMPOSITE	Mencegah produk duplikat dalam wishlist
9	roles	code	UNIQUE	Memastikan kode role unik
10	user_profiles	auth_user_id	UNIQUE	Satu auth user satu profile
11	user_profiles	user_id	UNIQUE	Satu user satu profile
12	user_sessions	session_token	UNIQUE	Memastikan token session unik
13	stripe_customers	stripe_customer_id	UNIQUE	Memastikan Stripe Customer ID unik

14	stripe_customers	user_id	UNIQUE	Satu user satu Stripe customer
15	stripe_orders	order_id	UNIQUE	Satu order satu Stripe tracking
16	stripe_orders	checkout_session_id	UNIQUE	Memastikan Stripe session ID unik
17	stripe_orders	payment_intent_id	UNIQUE	Memastikan payment intent unik

3.4.4.5 Check Constraints untuk Validasi Business Rules

Check constraints digunakan untuk memvalidasi data sesuai dengan aturan bisnis yang telah ditetapkan. Sistem mengimplementasikan 25 check constraint untuk memastikan integritas data dan kepatuhan terhadap aturan bisnis.

Tabel 3.18 Struktur Tabel Check Constraints untuk Validasi Business Rules

No.	Tabel	Check Constraint	Validasi Rule
1	products	price >= 0	Harga produk tidak boleh negatif
2	products	discount_price >= 0	Harga diskon tidak boleh negatif
3	products	stock >= 0	Stok tidak boleh negatif
4	products	care_level IN ('Mudah', 'Sedang', 'Sulit')	Level perawatan harus sesuai standar
5	cart_items	quantity > 0 AND quantity <= 999	Jumlah item harus positif dan realistik
6	orders	total_amount >= 0	Total pembayaran tidak boleh negatif
7	orders	shipping_cost >= 0	Ongkos kirim tidak boleh negatif
8	orders	final_amount >= 0	Total final tidak boleh negatif
9	orders	status IN ('pending', 'confirmed', 'processing', 'shipped', 'delivered', 'cancelled')	Status pesanan harus sesuai workflow
10	orders	payment_status IN ('unpaid', 'paid', 'refunded', 'partially_refunded')	Status pembayaran harus valid

11	product_reviews	rating BETWEEN 1 AND 5	Rating harus dalam range 1-5
12	product_reviews	status IN ('pending', 'approved', 'rejected')	Status review harus valid
13	order_items	quantity > 0 AND quantity <= 999	Jumlah item pesanan harus positif dan realistik
14	order_items	unit_price >= 0	Harga satuan tidak boleh negatif
15	order_items	total_price >= 0	Total harga item tidak boleh negatif
16	roles	char_length(code) >= 3	Kode role minimal 3 karakter
17	roles	char_length(name) >= 3	Nama role minimal 3 karakter
18	inventory_movements	movement_type IN ('IN','OUT','ADJUST')	Tipe pergerakan harus valid
19	inventory_movements	stock_before >= 0	Stok sebelum tidak boleh negatif
20	inventory_movements	stock_after >= 0	Stok setelah tidak boleh negatif
21	stripe_customers	char_length(currency) = 3	Currency code harus 3 karakter
22	stripe_orders	amount_subtotal >= 0	Subtotal tidak boleh negatif
23	stripe_orders	amount_total >= 0	Total amount tidak boleh negatif
24	stripe_orders	payment_status IN ('unpaid','paid','refunded','failed')	Status payment harus valid
25	stripe_orders	order_status IN ('pending','completed','cancelled')	Status order harus valid

3.4.4.6 Ringkasan Statistik Basis Data Azka Garden

Berdasarkan analisis struktur basis data yang telah dilakukan, berikut adalah ringkasan statistik lengkap sistem basis data Azka Garden yang mencerminkan implementasi komprehensif untuk mendukung operasional perdagangan daring tanaman hias.

Tabel 3.19 Struktur Tabel Ringkasan Statistik Basis Data Azka Garden

No.	Aspek Database	Jumlah	Keterangan
1	Total Tabel	15	8 Tabel Implementasi + 7 Tabel Enhancement
2	Total Kolom	151	Mencakup semua field dalam 15 tabel
3	Current Implementation Tables	8	Tabel yang sudah diimplementasikan
4	Enhancement Tables	7	Tabel untuk fitur advanced
5	Total Foreign Key Relationships	19	Relasi antar tabel untuk integritas data
6	Total Primary Keys	15	Setiap tabel memiliki UUID primary key
7	Total Unique Constraints	17	Mencegah duplikasi data kritis
8	Total Check Constraints	25	Validasi business rules
9	Total Composite Keys	3	Unique constraints gabungan
10	Stripe Integration Tables	2	Tabel untuk integrasi pembayaran
11	Audit & Security Tables	3	Tabel untuk audit, session, dan keamanan
12	Business Domain	Situs Web Perdagangan Daring Tanaman Hias	Sistem perdagangan tanaman hias online
13	Technology Stack	PostgreSQL + Supabase	Database cloud dengan fitur waktu nyata
14	System Features	Production Ready	RBAC, Integrasi Pembayaran, Audit Logging

3.4.5 Perancangan Struktur Navigasi Situs Web Perdagangan Dinamis Azka Garden

Perancangan struktur navigasi dalam sistem situs web perdagangan dinamis Azka Garden bertujuan untuk memberikan pengalaman pengguna yang intuitif dan

efisien. Struktur ini dirancang dengan mempertimbangkan berbagai tipe navigasi yang sesuai dengan kebutuhan pengguna, mulai dari browsing produk hingga proses checkout. Berikut adalah rincian dari struktur navigasi yang telah dirancang.

3.4.5.1 Arsitektur Navigasi Sistem

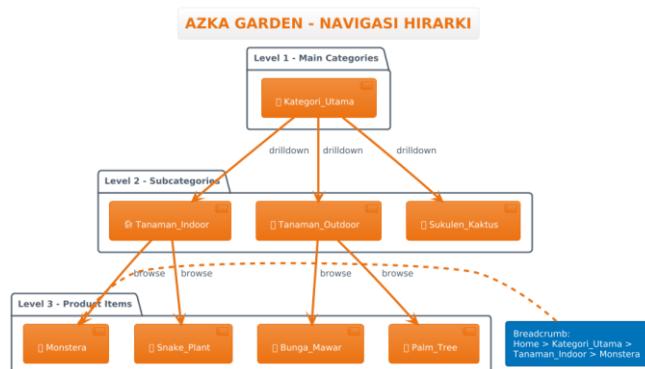
Sistem Azka Garden mengimplementasikan arsitektur navigasi yang terdiri dari beberapa tipe navigasi, termasuk hirarki, jaringan, campuran, linear, dan responsif. Setiap tipe navigasi memiliki karakteristik dan tujuan yang berbeda, yang dirancang untuk meningkatkan interaksi pengguna dengan sistem.

3.4.5.2 Struktur Navigasi Berdasarkan Tipe

Struktur navigasi dalam sistem Azka Garden dirancang untuk memberikan pengalaman pengguna yang optimal. Terdapat beberapa tipe navigasi yang digunakan, yaitu navigasi hirarki, jaringan, campuran, linear, responsif, dan alur perjalanan pengguna. Setiap tipe navigasi memiliki fungsi dan karakteristik yang berbeda untuk memenuhi kebutuhan pengguna.

3.4.5.3 Struktur Navigasi Hirarki (*Hierarchical Navigation*)

Struktur Navigasi Hirarki menggambarkan bagaimana pengguna dapat menavigasi melalui kategori utama, subkategori, dan item produk. Struktur ini memungkinkan pengguna untuk dengan mudah menemukan produk yang mereka cari dengan mengikuti jalur yang terorganisir.

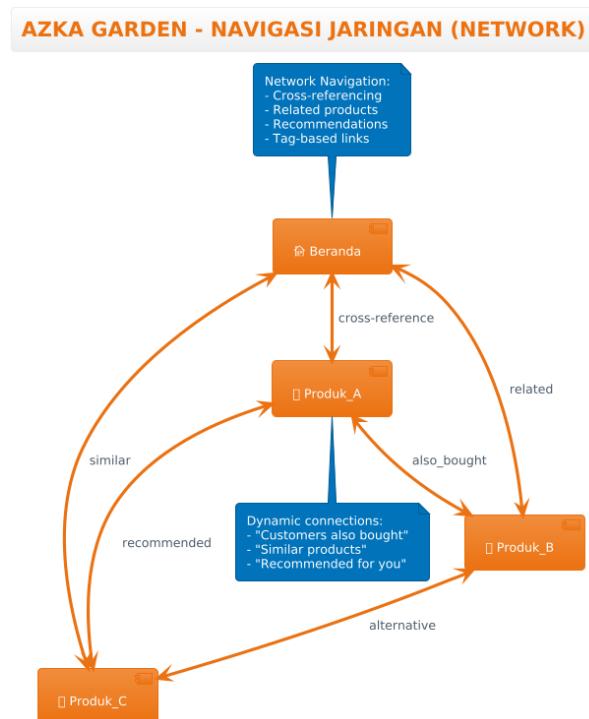


Gambar 3.48 Struktur Navigasi Hirarki

Struktur navigasi hirarki menyediakan struktur yang jelas dan terorganisir untuk menjelajahi produk berdasarkan kategori dan subkategori, memudahkan pengguna dalam menemukan produk yang diinginkan.

3.4.5.4 Struktur Navigasi Jaringan (*Network Navigation*)

Struktur navigasi jaringan menunjukkan bagaimana pengguna dapat menjelajahi produk melalui koneksi dinamis antar produk. Ini mencakup rekomendasi produk berdasarkan perilaku pengguna dan hubungan antar produk.



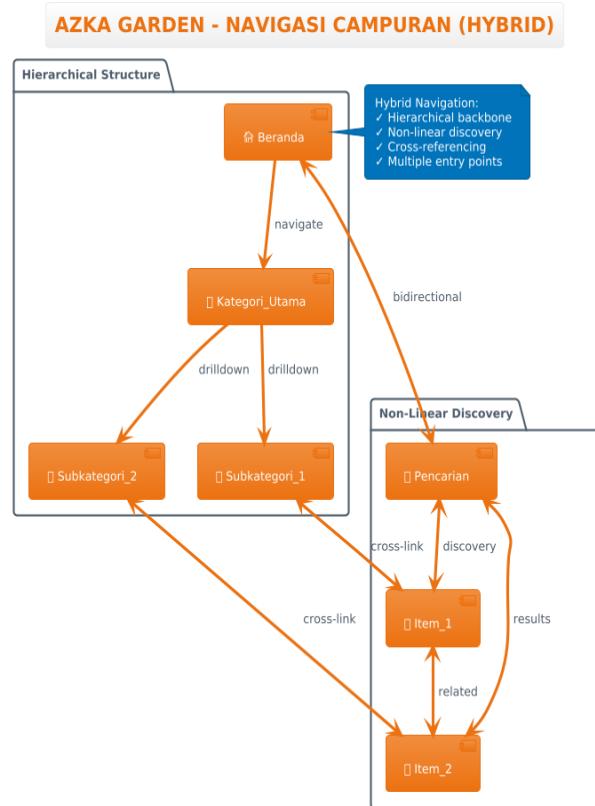
Gambar 3.49 Struktur Navigasi Jaringan

Struktur navigasi jaringan memungkinkan pengguna untuk menemukan produk baru melalui rekomendasi dan koneksi antar produk, meningkatkan peluang penjualan tambahan.

3.4.5.5 Struktur Navigasi Campuran (*Hybrid Navigation*)

Struktur navigasi campuran menggabungkan elemen dari navigasi hirarki

dan non-linear, memberikan fleksibilitas kepada pengguna untuk menjelajahi produk dengan cara yang lebih dinamis.

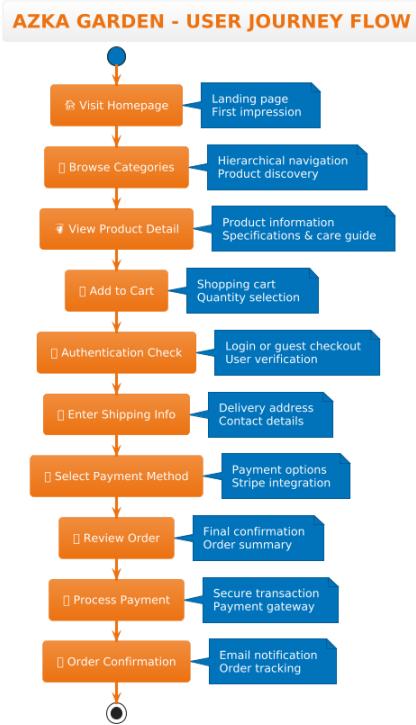


Gambar 3.50 Struktur Navigasi Campuran

Struktur navigasi campuran memberikan pengalaman yang lebih kaya dengan menggabungkan struktur hierarki dan penemuan non-linear, memungkinkan pengguna untuk menjelajahi produk dengan cara yang lebih fleksibel.

3.4.5.6 Struktur Navigasi Linear (*Linear Navigation*)

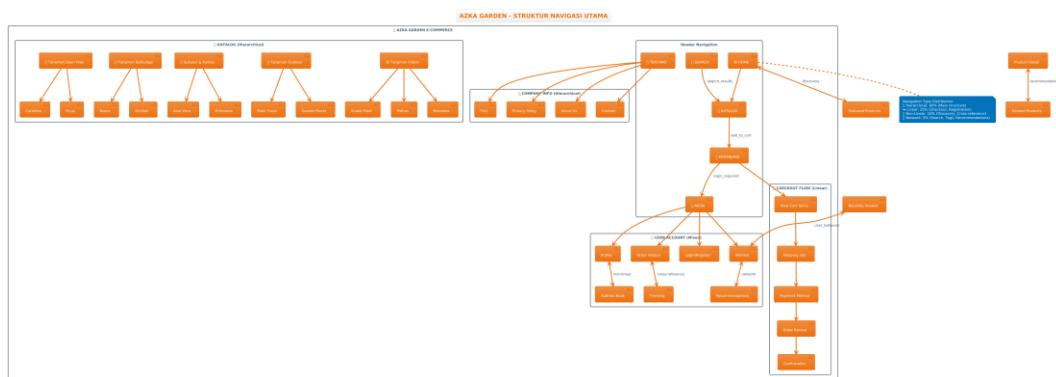
Struktur navigasi linear menggambarkan proses checkout yang terstruktur, di mana pengguna mengikuti langkah-langkah yang jelas dan terurut untuk menyelesaikan pembelian.



Gambar 3.51 Struktur Navigasi Linear

3.4.5.7 Struktur Navigasi Utama Azka Garden

Struktur navigasi utama menggambarkan keseluruhan struktur navigasi yang digunakan dalam sistem Azka Garden, termasuk header, kategori, checkout, dan informasi perusahaan.

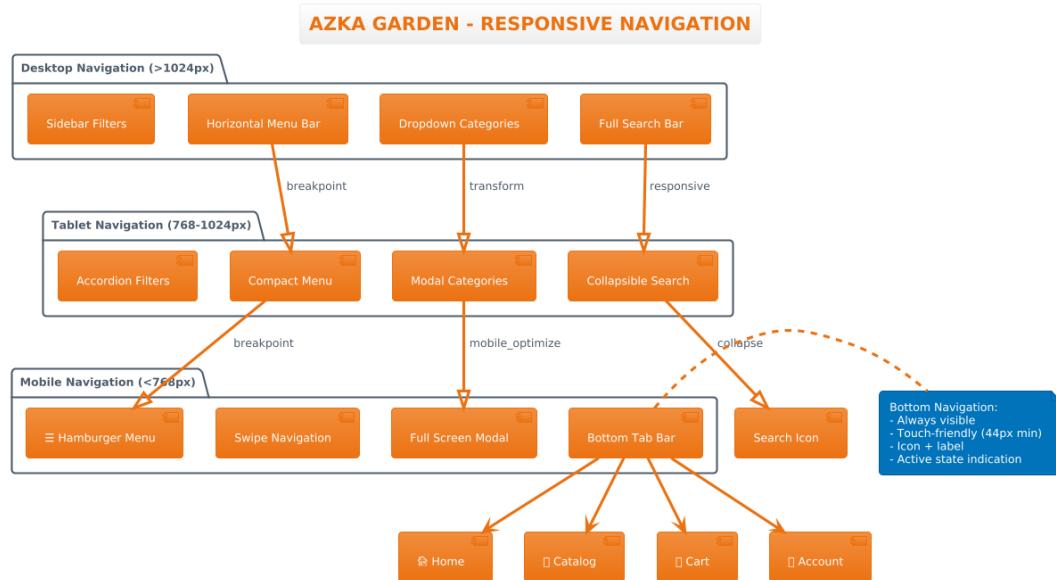


Gambar 3.52 Struktur Navigasi Utama Azka Garden

Struktur navigasi utama memberikan panduan yang jelas bagi pengguna untuk menjelajahi berbagai bagian dari situs web perdagangan dinamis, termasuk kategori produk, proses checkout, dan informasi perusahaan.

3.4.5.8 Struktur Navigasi Responsif

Struktur navigasi responsif menunjukkan bagaimana navigasi dioptimalkan untuk berbagai perangkat, termasuk desktop, tablet, dan mobile, untuk memastikan pengalaman pengguna yang konsisten di semua platform.

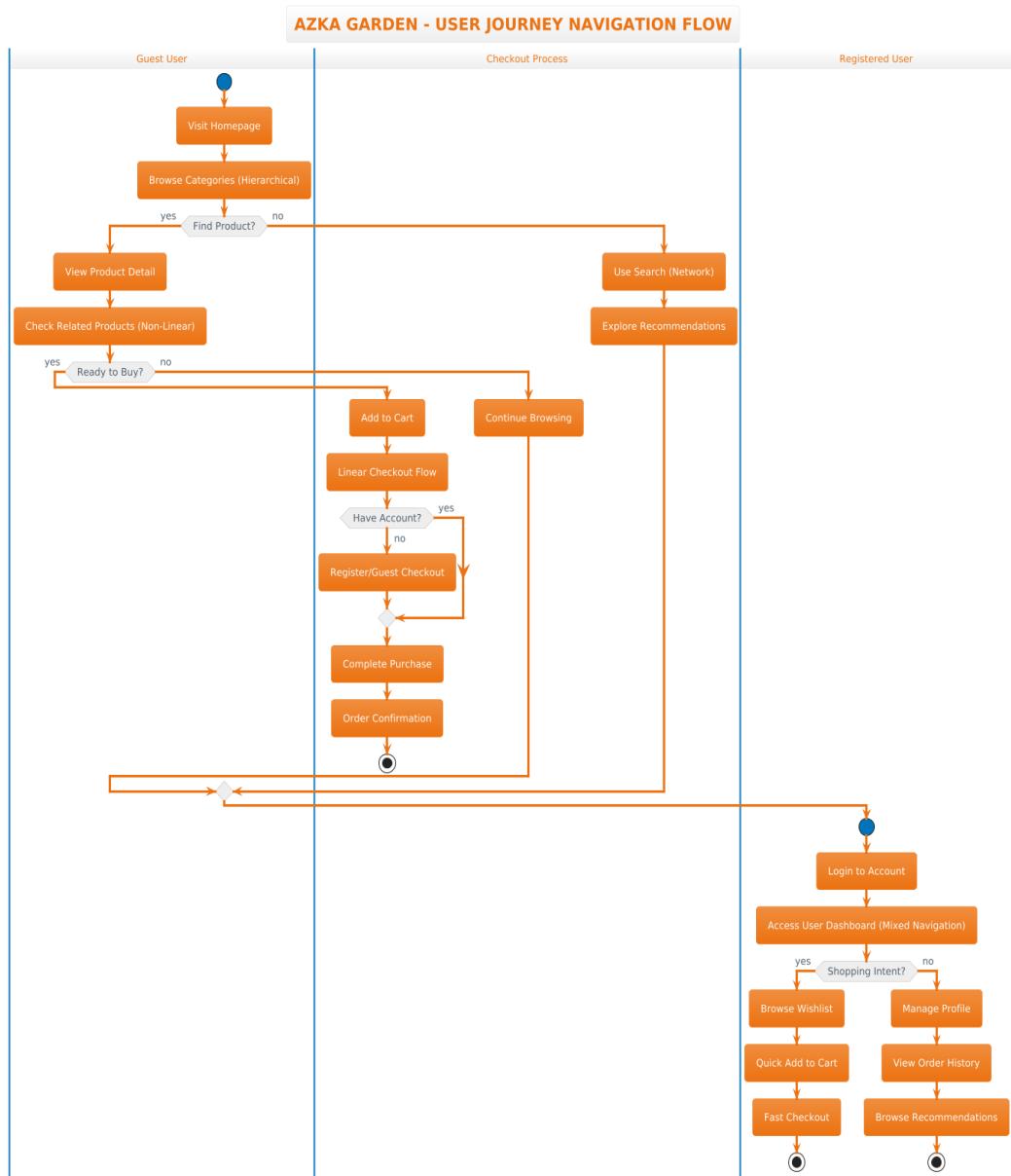


Gambar 3.53 Struktur Navigasi Responsif

Struktur navigasi responsif memastikan bahwa pengguna dapat dengan mudah mengakses semua fitur dan informasi di situs, terlepas dari perangkat yang mereka gunakan, dengan penyesuaian tampilan yang sesuai.

3.4.5.9 Alur Navigasi Perjalanan Pengguna

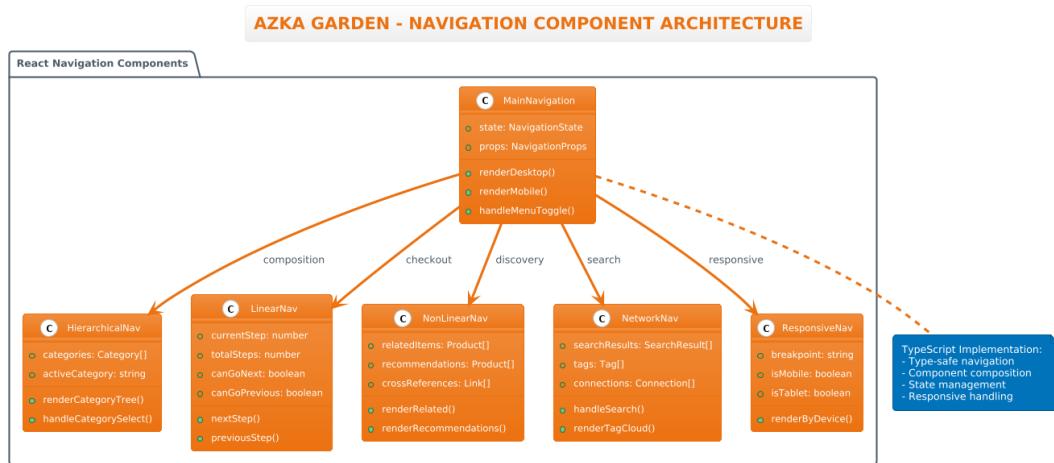
Alur navigasi perjalanan pengguna menggambarkan perjalanan pengguna dari pengunjung tamu hingga pengguna terdaftar, menunjukkan langkah-langkah yang diambil dalam proses interaksi dengan sistem.



Gambar 3.54 Alur Navigasi Perjalanan Pengguna

3.4.5.10 Arsitektur Komponen Navigasi

Arsitektur komponen navigasi menunjukkan struktur komponen navigasi yang digunakan dalam aplikasi, termasuk bagaimana komponen tersebut saling berinteraksi.



Gambar 3.55 Arsitektur Komponen Navigasi

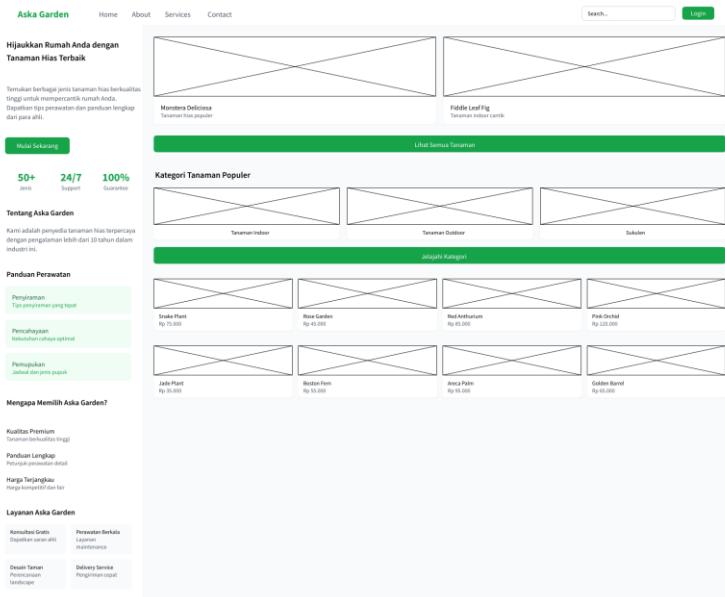
Arsitektur komponen navigasi menggambarkan bagaimana komponen navigasi dibangun dan diorganisir dalam aplikasi, memastikan bahwa navigasi dapat dikelola dengan baik dan responsif terhadap kebutuhan pengguna.

3.4.6 Perancangan Antarmuka Pengguna

Perancangan struktur antarmuka pengguna Azka Garden dirancang dengan pendekatan yang berpusat pada pengguna yang mengutamakan kemudahan navigasi, konsistensi visual, dan pengalaman pengguna yang optimal. Setiap halaman dirancang dengan mempertimbangkan perjalanan pengguna dan optimalisasi konversi untuk situs web perdagangan daring tanaman hias.

3.4.6.1 Perancangan Halaman Beranda (/)

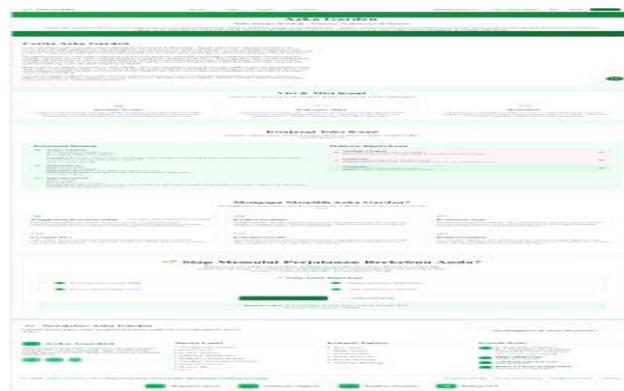
Perancangan halaman beranda merupakan pintu gerbang utama situs web Azka Garden yang dirancang untuk memberikan kesan pertama yang menarik dan memandu pengguna untuk menjelajahi produk-produk tanaman hias. Rancangan halaman ini menampilkan bagian utama dengan spanduk menarik, kisi kategori tanaman, produk unggulan, bagian testimoni, dan pendaftaran buletin untuk keterlibatan maksimal.



Gambar 3.56 Perancangan Halaman Beranda (/)

3.4.6.2 Perancangan Halaman Tentang Kami (/about)

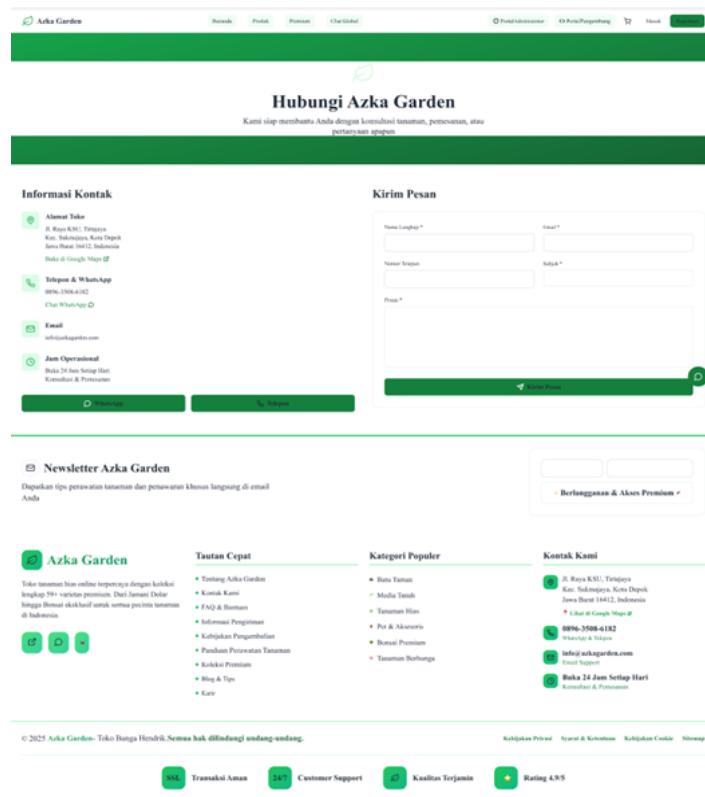
Perancangan halaman tentang kami menyajikan informasi lengkap mengenai Azka Garden sebagai perusahaan, visi misi, dan komitmen terhadap kualitas tanaman hias. Rancangan halaman ini menampilkan cerita perusahaan, bagian tim, pernyataan misi dan visi, statistik pencapaian, dan garis waktu perusahaan untuk membangun kepercayaan dan kredibilitas dengan pelanggan.



Gambar 3.57 Perancangan Halaman Tentang Kami (/about)

3.4.6.3 Perancangan Halaman Kontak (/contact)

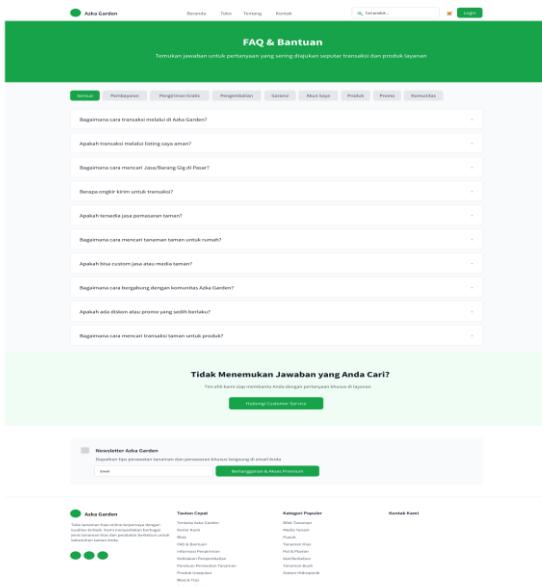
Perancangan halaman kontak menyediakan berbagai cara untuk menghubungi Azka Garden, termasuk form kontak, informasi alamat, dan jam operasional. Implementasi mencakup forum kontak dengan validasi waktu nyata, integrasi Google Maps, informasi kontak lengkap, dan link sosial media untuk memudahkan komunikasi pelanggan.



Gambar 3.58 Perancangan Halaman Kontak (/contact)

3.4.6.4 Perancangan Halaman FAQ (/faq)

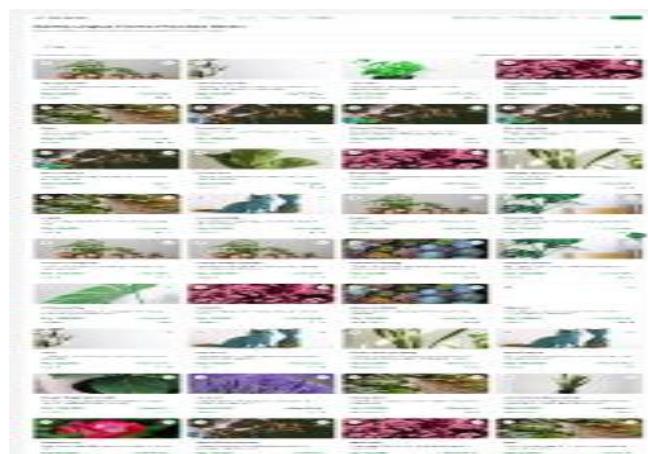
Perancangan halaman FAQ menyediakan jawaban untuk pertanyaan yang sering diajukan pelanggan mengenai produk, pengiriman, dan layanan Azka Garden. Halaman ini diorganisir dengan search functionality, kategorisasi pertanyaan, accordion interface, dan popular FAQ section untuk memudahkan pengguna menemukan informasi yang dibutuhkan.



Gambar 3.59 Perancangan Halaman FAQ (/faq)

3.4.6.5 Perancangan Halaman Katalog Produk (/products)

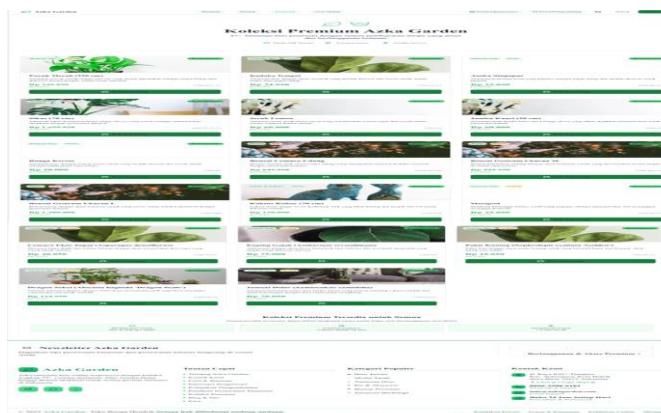
Perancangan halaman katalog produk merupakan jantung dari situs web perdagangan daring Azka Garden yang menampilkan seluruh koleksi tanaman hias dengan sistem filtering dan sorting yang canggih. Implementasi mencakup product grid yang responsif, filter sidebar, search functionality, dan pagination untuk handling large dataset dengan performa optimal.



Gambar 3.60 Perancangan Halaman Katalog Produk (/products)

3.4.6.6 Perancangan Halaman Koleksi Premium (/stripe-products)

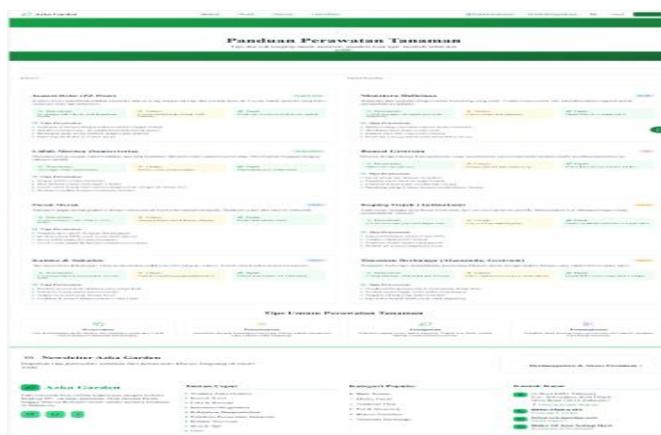
Perancangan halaman koleksi premium menampilkan produk-produk eksklusif dan paket berlangganan dengan integrasi Stripe untuk payment processing. Implementasi mencakup premium product grid, subscription plans, payment methods, dan billing history untuk memberikan pengalaman premium yang seamless.



Gambar 3.61 Perancangan Halaman Koleksi Premium (/stripe-products)

3.4.6.7 Perancangan Halaman Panduan Perawatan (/care-guide)

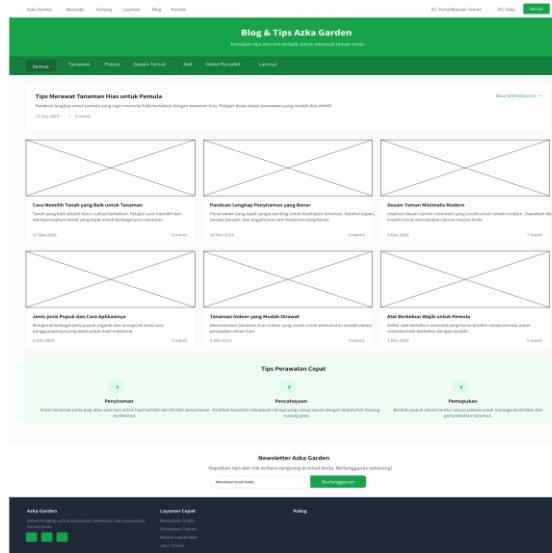
Perancangan halaman panduan perawatan menyediakan edukasi komprehensif untuk merawat tanaman hias dengan baik. Implementasi mencakup care guide grid, plant care calculator, seasonal tips, troubleshooting guide, and video tutorials untuk memberikan value added kepada pelanggan.



Gambar 3.62 Perancangan Halaman Panduan Perawatan (/care-guide)

3.4.6.8 Perancangan Halaman Blog & Tips (/blog)

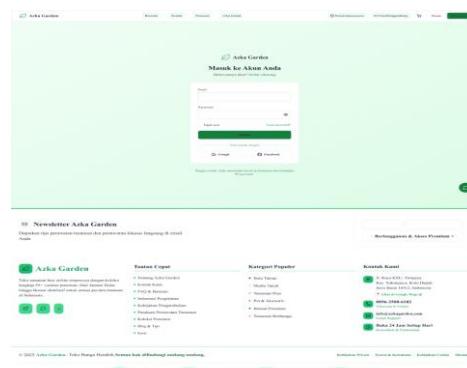
Perancangan halaman blog menyajikan artikel dan tutorial terkait tanaman hias untuk edukasi dan engagement pelanggan. Implementasi mencakup blog post grid, categories, featured posts, search functionality, dan related posts untuk meningkatkan SEO dan user engagement.



Gambar 3.63 Perancangan Halaman Blog & Tips (/blog)

3.4.6.9 Perancangan Halaman Login (/login)

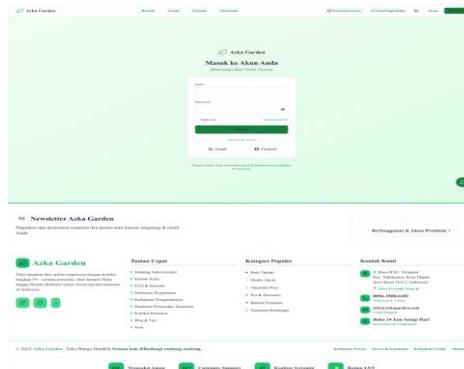
Perancangan halaman login menyediakan akses masuk ke akun pengguna dengan sistem autentikasi yang aman menggunakan Supabase Auth. Implementasi mencakup login form dengan validasi, social login options, forgot password functionality, dan redirect handling untuk user experience yang smooth.



Gambar 3.64 Perancangan Halaman Login (/login)

3.4.6.10 Perancangan Halaman Registrasi (/register)

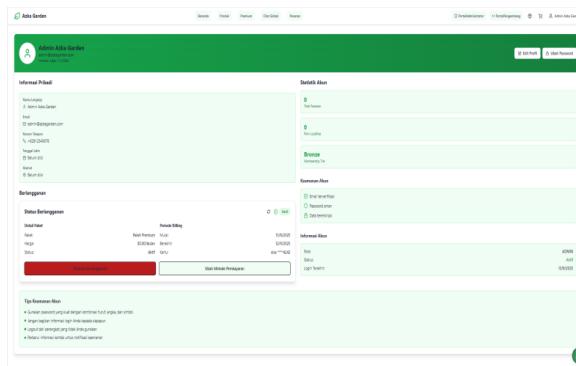
Perancangan halaman registrasi memungkinkan pengguna baru untuk membuat akun dengan proses yang mudah dan aman. Implementasi mencakup registration form dengan validasi waktu nyata, verifikasi email, persetujuan kebijakan, dan alur pembukaan untuk onboarding yang efektif.



Gambar 3.65 Perancangan Halaman Registrasi (/register)

3.4.6.11 Perancangan Halaman Profil (/profile)

Perancangan halaman profil memungkinkan pengguna untuk mengelola informasi personal dan preferensi akun. Implementasi mencakup formulir profil, avatar upload, manajemen alamat, notifikasi pengaturan, dan akun keamanan untuk memberikan kontrol penuh kepada pengguna.

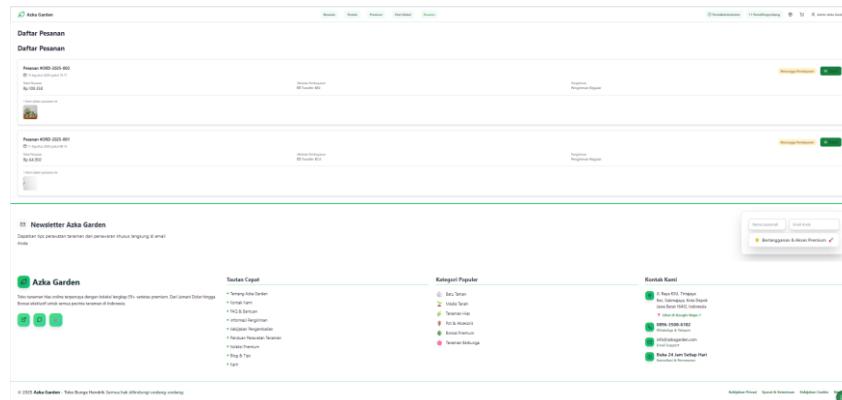


Gambar 3.65 Perancangan Halaman Profil (/profile)

3.4.6.12 Perancangan Halaman Pesanan (/orders)

Perancangan halaman pesanan menampilkan riwayat pembelian dan status

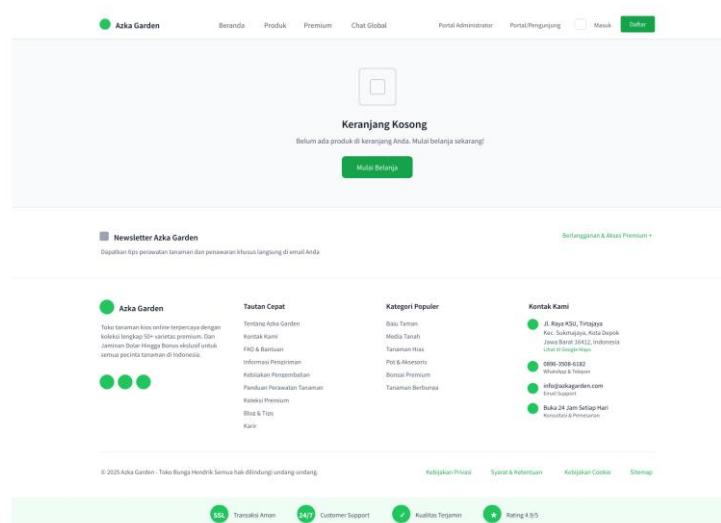
pengiriman dengan pelacakan yang waktunya nyata. Implementasi mencakup daftar riwayat pesanan , tinjauan detail pesanan, pelacakan informasi, dan fungsionalitas pemesanan kembali untuk kemudahan pengguna dalam monitoring transaksi.



Gambar 3.66 Perancangan Halaman Pesanan (/orders)

3.4.6.13 Perancangan Halaman Keranjang (/cart)

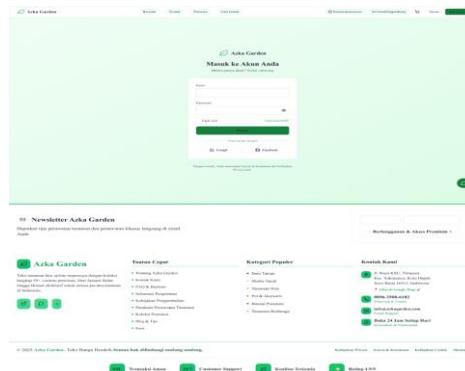
Perancangan halaman keranjang belanja memungkinkan pengguna untuk mengelola item sebelum checkout dengan interface yang intuitif. Implementasi mencakup daftar keranjang belanja, penyesuaian kuantitas, perhitungan harga, estimasi pengiriman, dan tombol pemrosesan pembayaran untuk optimisasi konversi.



Gambar 3.67 Perancangan Halaman Keranjang (/cart)

3.4.6.14 Perancangan Halaman Wishlist (/wishlist)

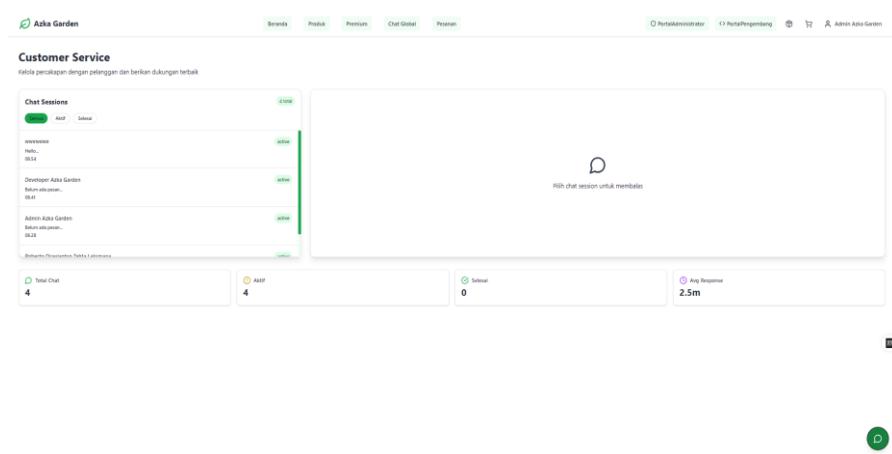
Perancangan halaman wishlist memungkinkan pengguna untuk menyimpan produk favorit untuk pembelian di masa mendatang. Implementasi mencakup grid keinginan barang, fungsionalitas untuk menambahkan keranjang belanja, opsi pembagian, dan mesin rekomendasi untuk meningkatkan engagement dan sales.



Gambar 3.68 Perancangan Halaman Wishlist (/wishlist)

3.4.6.15 Perancangan Halaman Customer Service (/customer-service)

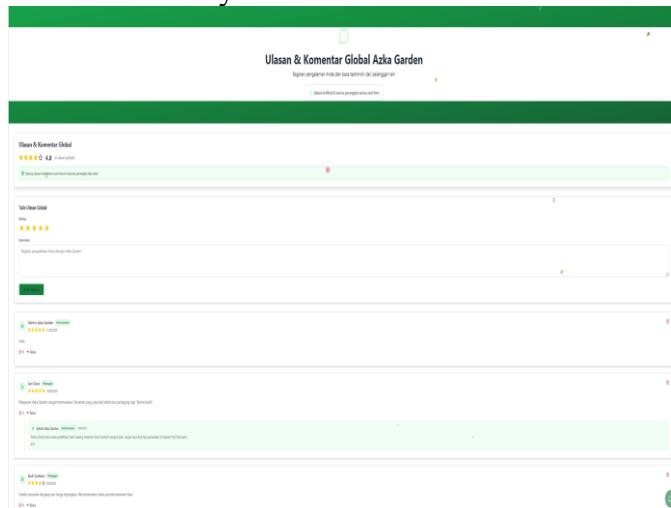
Halaman customer service menyediakan platform komunikasi waktu nyata untuk komunitas pecinta tanaman hias. Implementasi mencakup waktu nyata pengiriman pesan, indikator presensi pengguna, riwayat pesan, dan alat moderasi untuk menciptakan komunitas yang positif dan selaras.



Gambar 3.69 Perancangan Halaman Customer Service (/customer-service)

3.4.6.16 Perancangan Halaman Ulasan dan Komentar (/reviews)

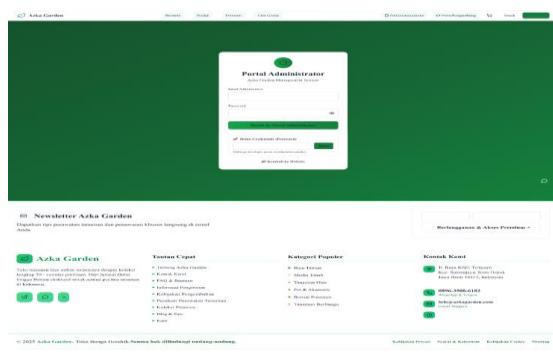
Perancangan halaman ulasan menampilkan feedback pelanggan terhadap produk dan layanan Azka Garden. Implementasi mencakup daftar ulasan, sistem penilaian, unggah foto, bantuan penyuaran, dan balasan dari penjual untuk membangun trust dan credibility.



Gambar 3.70 Perancangan Halaman Ulasan dan Komentar (/reviews)

3.4.6.17 Perancangan Portal Administrator (/admin/login)

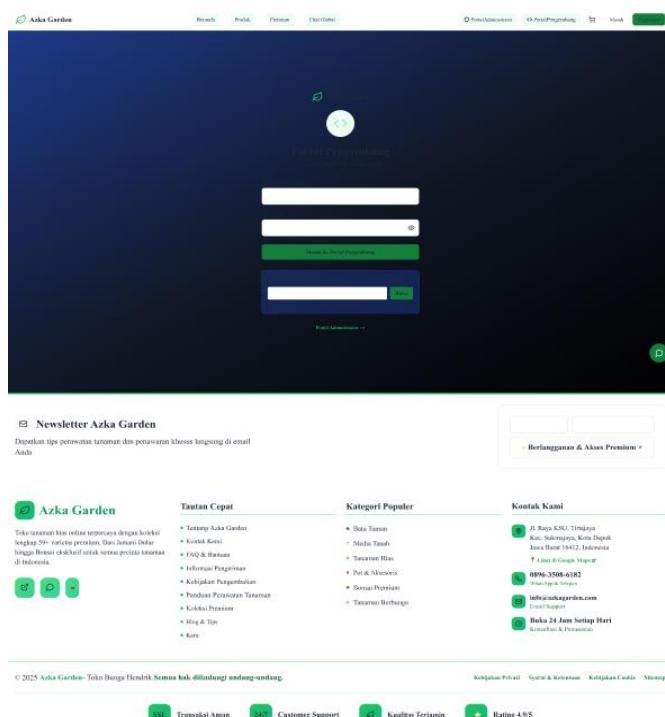
Perancangan portal administrator menyediakan akses khusus untuk mengelola sistem Azka Garden secara keseluruhan. Implementasi mencakup dasbor administrator, manajemen pengguna, manajemen produk, manajemen pesanan, dan analisis untuk efisiensi operasional.



Gambar 3.71 Perancangan Portal Administrator (/admin/login)

3.4.6.18 Perancangan Portal Pengembang (/developer/login)

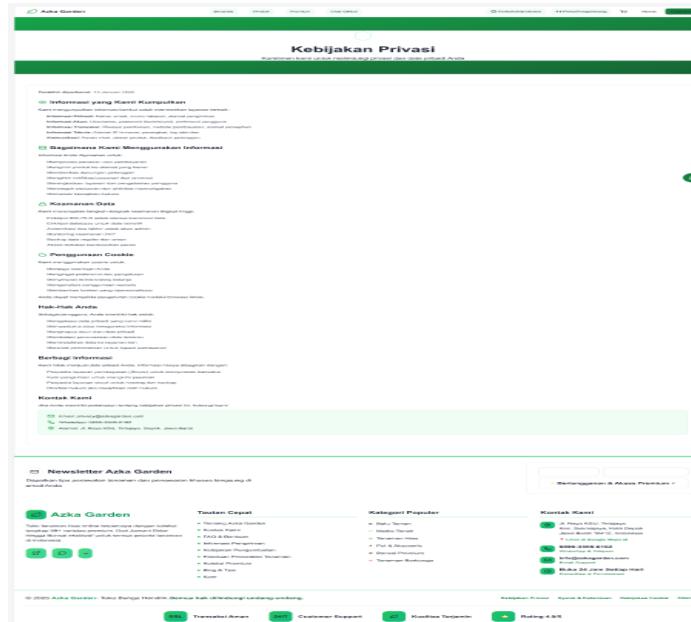
Perancangan portal pengembang menyediakan akses untuk developer dalam monitoring dan debugging sistem. Implementasi mencakup dokumentasi API, log sistem, metrik performa, dan alat pengembangan untuk pemeliharaan dan perbaikan yang berkelanjutan.



Gambar 3.72 Perancangan Portal Pengembang (/developer/login)

3.4.6.19 Perancangan Halaman Kebijakan Privasi (/privacy)

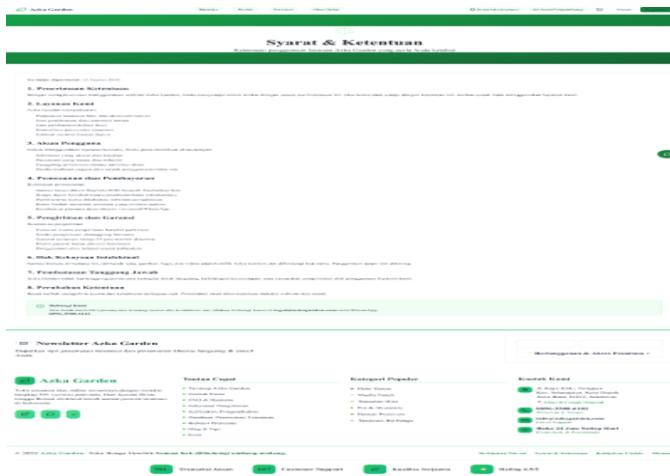
Halaman kebijakan privasi menjelaskan bagaimana Azka Garden mengumpulkan, menggunakan, dan melindungi data pribadi pengguna sesuai dengan regulasi perlindungan data. Implementasi mencakup konten kebijakan privasi, penjelasan koleksi data, informasi hak cipta, dan kontak untuk permintaan privasi untuk membangun kepercayaan dan compliance dengan GDPR dan regulasi lokal.



Gambar 3.73 Perancangan Halaman Kebijakan Privasi (/privacy)

3.4.6.20 Perancangan Halaman Syarat dan Ketentuan (/terms)

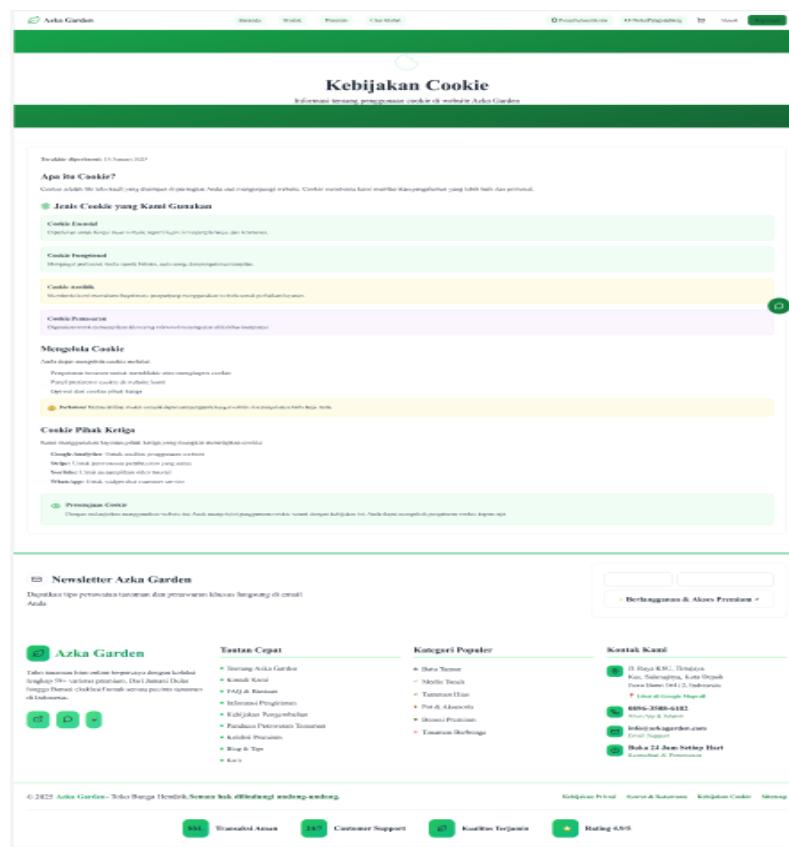
Halaman syarat dan ketentuan menetapkan aturan penggunaan platform Azka Garden dan hak serta kewajiban pengguna. Implementasi mencakup ketentuan dari konten layanan, obligasi pengguna, aturan platform, pembatasan tanggung jawab, dan penyelesaian sengketa untuk melindungi kepentingan perusahaan dan pengguna.



Gambar 3.74 Perancangan Halaman Syarat dan Ketentuan (/terms)

3.4.6.21 Perancangan Halaman Kebijakan Cookie (/cookies)

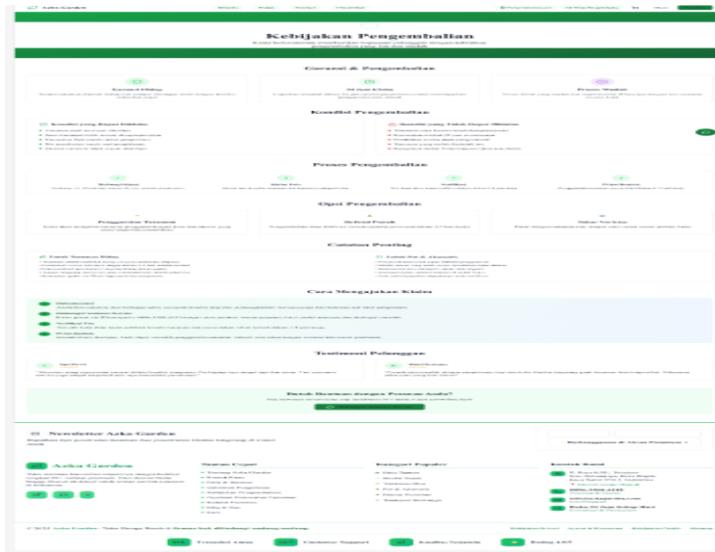
Perancangan halaman kebijakan cookie menjelaskan penggunaan cookie dan teknologi tracking lainnya di situs web perdagangan daring Azka Garden. Implementasi mencakup kebijakan cookie, penggunaan jenis cookie, opsi manajemen cookie, dan informasi cookie dari orang ketiga untuk transparansi dan kontrol pengguna.



Gambar 3.75 Perancangan Halaman Kebijakan Cookie (/cookies)

3.4.6.22 Perancangan Halaman Kebijakan Pengembalian (/returns)

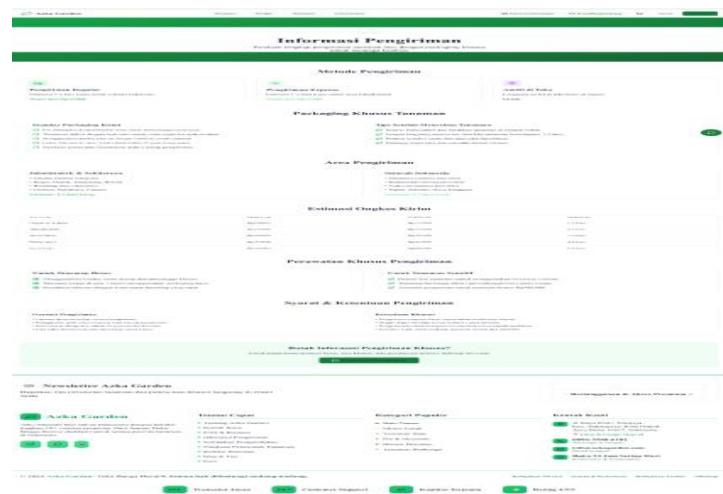
Perancangan halaman kebijakan pengembalian menjelaskan prosedur dan syarat untuk pengembalian produk tanaman hias. Implementasi mencakup return policy details, return process steps, refund timeline, and special conditions for live plants untuk memberikan clarity dan confidence kepada pembeli.



Gambar 3.76 Perancangan Halaman Kebijakan Pengembalian (/returns)

3.4.6.23 Perancangan Halaman Informasi Pengiriman (/shipping)

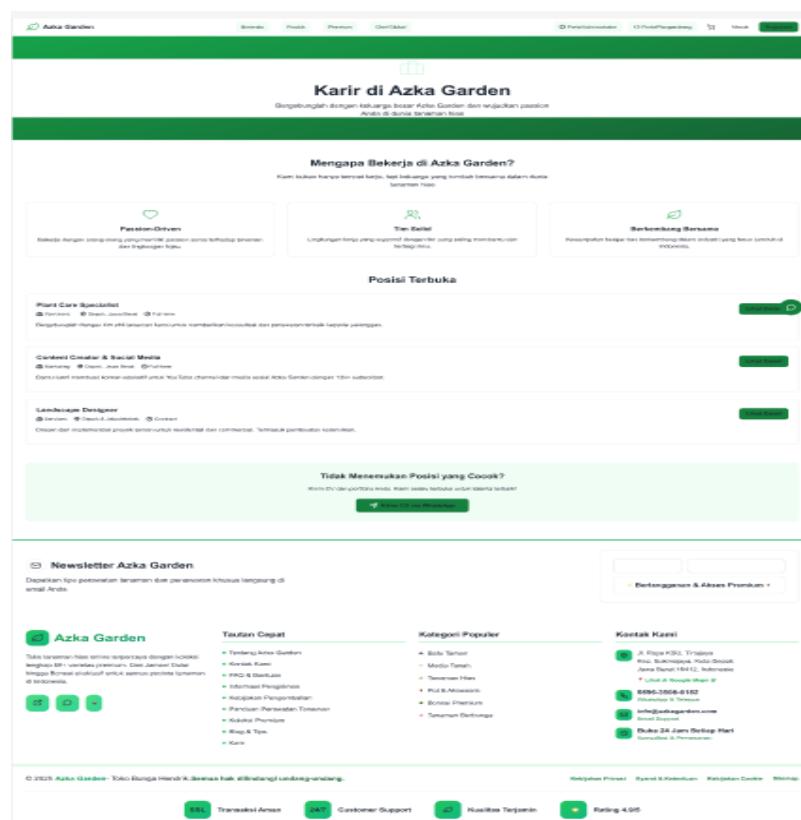
Perancangan halaman informasi pengiriman menyediakan detail lengkap mengenai opsi pengiriman, biaya, dan estimasi waktu. Implementasi mencakup opsi pengiriman, daerah pengiriman, kalkulator biaya pengiriman, informasi pengemasan, dan care instruksi dari rencana pengiriman untuk memastikan produk sampai dalam kondisi optimal.



Gambar 3.77 Perancangan Halaman Informasi Pengiriman (/shipping)

3.4.6.24 Perancangan Halaman Karir (/careers)

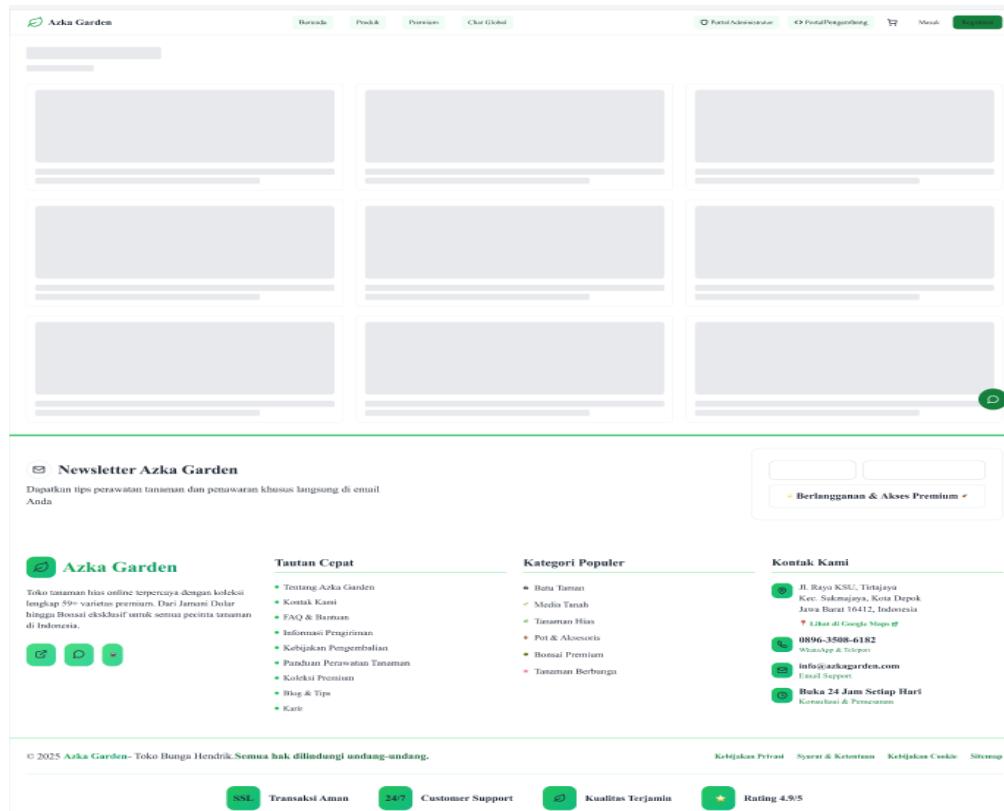
Perancangan halaman karir menampilkan lowongan pekerjaan dan informasi untuk bergabung dengan tim Azka Garden. Implementasi mencakup pendaftaran pekerjaan, informasi budaya perusahaan, proses aplikasi, keuntungan ketenagakerjaan, dan kesempatan pemberian karir untuk menarik talent terbaik dan membangun pengelolaan karyawan.



Gambar 3.78 Perancangan Halaman Karir (/careers)

3.4.6.25 Perancangan Halaman Pencarian (/search)

Perancangan halaman pencarian menampilkan hasil pencarian produk dan konten berdasarkan query pengguna. Implementasi mencakup search hasil pencarian, filter yang ditangguhkan, saran pencarian, penangan hasil pencarian tidak ditemukan, dan analisis pencarian untuk meningkatkan penemuan dan pengalaman pengguna.



Gambar 3.79 Perancangan Halaman Pencarian (/search)

3.5 Tahap Implementasi

Implementasi sistem Azka Garden mencakup dua aspek utama: implementasi basis data yang telah dirancang sebelumnya dan pengembangan antarmuka pengguna yang responsif. Bagian ini menjelaskan secara detail proses implementasi dari struktur basis data hingga pengembangan halaman-halaman web yang sesuai dengan sitemap yang telah ditentukan. Implementasi sistem Azka Garden mencakup dua aspek utama: implementasi basis data yang telah dirancang sebelumnya dan pengembangan antarmuka pengguna yang responsif. Bagian ini menjelaskan secara detail proses implementasi dari struktur basis data hingga pengembangan halaman-halaman web yang sesuai dengan sitemap yang telah ditentukan.

3.5.1 Implementasi Basis Data PostgreSQL dengan Supabase

Implementasi basis data Azka Garden menggunakan PostgreSQL melalui platform Supabase yang menyediakan fitur waktu nyata, autentikasi, dan API otomatis. Proses implementasi dilakukan secara bertahap dengan mempertimbangkan integritas data dan performa sistem. Database diimplementasikan dengan 15 tabel utama yang terdiri dari 8 tabel implementasi saat ini dan 7 tabel enhancement untuk mendukung fitur lanjutan.

The screenshot shows the AWS Glue Data Catalog interface with the following details:

- Project:** Amazon_ECommerce_Dev
- Tables:** Amazon_ECommerce_Tbl
- Schema:**

No	Name	Type	Data	Constraints	Relationships
1	id	int	26 chars	NOT NULL, DEFAULT 0, IDENTITY(1,1)	Identifier with support
2	name	text	unlimited	NOT NULL	Kecepatan kumpulan
3	desc_option	text	unlimited	NULL	Descript kumpulan
4	category_id	int	unlimited	NULL	laki, gender支撑
5	created_at	timestamp	8 bytes	DEFAULT 0000-00-00 00:00:00	Waktu pendaftaran record
6	updated_at	timestamp	8 bytes	DEFAULT 0000-00-00 00:00:00	Identifier with support
7	city	text	unlimited	NULL	Kota tempat tinggal
8	pincode_code	text	unlimited	NULL	Kode pos
9	created_at	timestamp	8 bytes	DEFAULT 0000-00-00 00:00:00	Waktu registrasi
10	updated_at	timestamp	8 bytes	DEFAULT 0000-00-00 00:00:00	Waktu update terakhir
11	active_user_id	int	36 chars	PK, FOREIGN KEY	Field description
12	note	text	unlimited	NULL	Nota pengguna

Gambar 3.80 Tampilan Basis Data

3.5.1.1 Setup dan Konfigurasi Basis Data

Tahap pertama implementasi melibatkan pengaturan lingkungan dan konfigurasi database dasar dengan menggunakan Supabase sebagai *Backend-as-a-Service* (BaaS). Konfigurasi ini mencakup pengaturan connection pooling, keamanan SSL, dan backup otomatis untuk memastikan keandalan sistem.

Gambar 3.81 Tampilan Setup dan Konfigurasi Basis Data

3.5.1.2 Implementasi Tabel *Categories*

Implementasi tabel categories sebagai tabel master untuk klasifikasi tanaman hias berdasarkan jenis dan karakteristik. Tabel ini menjadi foundation untuk sistem kategorisasi produk yang hierarkis dan mudah dikelola.

No	Nama_Field	Tipe_Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generate_v4()	Identifier unik kategori
2	name	text	-65535 chars	NOT NULL, UNIQUE	Nama kategori tanaman
3	description	text	-65535 chars	NULL	Deskripsi kategori
4	image_url	text	-65535 chars	NULL	URL gambar kategori
5	created_at	timestamp	8 bytes	DEFAULT now()	Waktu pembuatan record

Gambar 3.82 Implementasi Tabel *Categories*

3.5.1.3 Implementasi Tabel *Users*

Implementasi tabel users untuk menyimpan informasi pelanggan dengan integrasi Supabase Auth. Tabel ini mencakup data personal, verifikasi email dan telepon, serta status aktif pengguna untuk manajemen akun yang komprehensif.

No	Nama_Field	Tipe_Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generate_v4()	Identifier unik user
2	username	text	-1000 chars	NOT NULL, UNIQUE	Nama pengguna unik
3	email	text	-1000 chars	NOT NULL, UNIQUE	Alamat email unik
4	full_name	text	-1000 chars	NOT NULL	Nama lengkap pelanggan
5	phone	text	-1000 chars	NULL	Nomor telepon
6	address	text	-1000 chars	NULL	Alamat lengkap
7	city	text	-1000 chars	NULL	Kota tempat tinggal
8	postal_code	text	-1000 chars	NULL	Kode pos
9	created_at	timestamp	8 bytes	DEFAULT now()	Waktu registrasi
10	updated_at	timestamp	8 bytes	DEFAULT now()	Waktu update terakhir
11	auth_user_id	uuid	36 chars	Variable	Field description
12	role_id	uuid	36 chars	FOREIGN KEY → roles(id)	Role pengguna
13	display_name	varchar	100 chars	Variable	Field description
14	avatar_url	varchar	Variable	Variable	Field description

Gambar 3.83 Implementasi Tabel *Users*

3.5.1.4 Implementasi Tabel *Products*

Implementasi tabel *products* sebagai *core entity* untuk menyimpan informasi produk tanaman hias. Tabel ini mencakup detail produk, harga, stok,

spesifikasi perawatan, dan metadata untuk mendukung e-commerce yang lengkap.

No	Nama_Field	Tipe_Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generate_v4()	Identifier unik produk
2	name	text	-65535 chars	NOT NULL	Nama produk tanaman
3	description	text	-65535 chars	NULL	Deskripsi lengkap produk
4	price	integer	4 bytes	NOT NULL, CHECK (price >= 0)	Harga produk dalam rupiah
5	discount_price	integer	4 bytes	DEFAULT 0, CHECK (discount_price >= 0)	Harga diskon (jika ada)
6	stock	integer	4 bytes	DEFAULT 0, CHECK (stock >= 0)	Jumlah stok tersedia
7	category_id	uuid	36 chars	NOT NULL, FOREIGN KEY → categories(id)	Referensi ke tabel categories
8	image_url	text	-65535 chars	NULL	URL gambar produk
9	care_level	text	-65535 chars	CHECK (care_level IN ('Mudah', 'Sedang', 'Sulit'))	Tingkat perawatan tanaman
10	light_requirement	text	-65535 chars	NULL	Kebutuhan cahaya
11	watering_frequency	text	-65535 chars	NULL	Frekuensi penyiraman
12	is_active	boolean	1 byte	DEFAULT true	Status aktif produk
13	created_at	timestamptz	8 bytes	DEFAULT now()	Waktu pembuatan record
14	updated_at	timestamptz	8 bytes	DEFAULT now()	Waktu update terakhir
15	sku	varchar	100 chars	UNIQUE	Stock Keeping Unit
16	stripe_price_id	varchar	Variable	Variable	Field description
17	base_price_cents	bigint	8 bytes	Variable	Field description

Gambar 3.84 Implementasi Tabel *Products*

3.5.1.5 Implementasi Tabel Cart_Items

Implementasi tabel cart_items untuk mengelola keranjang belanja pengguna dengan session persistence. Tabel ini memungkinkan pengguna menyimpan item belanja sementara sebelum melakukan checkout.

No	Nama_Field	Tipe_Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generate_v4()	Identifier unik item cart
2	user_id	uuid	36 chars	NOT NULL, FOREIGN KEY → users(id)	Referensi ke tabel users
3	product_id	uuid	36 chars	NOT NULL, FOREIGN KEY → products(id)	Referensi ke tabel products
4	quantity	INTEGER	Variable	DEFAULT 1, CHECK (quantity > 0 AND quantity <= 999)	Jumlah item
5	added_at	timestamptz	8 bytes	DEFAULT now()	Waktu item ditambahkan

Gambar 3.85 Implementasi Tabel Cart_Items

3.5.1.6 Implementasi Tabel Orders

Implementasi tabel orders untuk menyimpan informasi header pesanan dengan tracking lengkap. Tabel ini mencakup status pesanan, informasi pembayaran, alamat pengiriman, dan metadata untuk manajemen order yang efisien.

No	Nama_Field	Tipe_Data	Ukuran	Constraint	Keterangan
1	<code>id</code>	<code>uuid</code>	36 chars	PRIMARY KEY, DEFAULT <code>uuid_generate_v4()</code>	Identifier unik pesanan
2	<code>order_number</code>	<code>text</code>	~1000 chars	NOT NULL, UNIQUE	Nomor pesanan unik
3	<code>user_id</code>	<code>uuid</code>	36 chars	NOT NULL, FOREIGN KEY → <code>users(id)</code>	Referensi ke tabel users
4	<code>total_amount</code>	<code>INTEGER</code>	Variable	NOT NULL, CHECK (<code>total_amount >= 0</code>)	Total harga produk
5	<code>shipping_cost</code>	<code>INTEGER</code>	Variable	DEFAULT 15000, CHECK (<code>shipping_cost >= 0</code>)	Biaya pengiriman
6	<code>final_amount</code>	<code>INTEGER</code>	Variable	NOT NULL, CHECK (<code>final_amount >= 0</code>)	Total akhir yang dibayar
7	<code>status</code>	<code>text</code>	~1000 chars	DEFAULT 'pending', CHECK (<code>status IN ('pending', 'confirmed', 'proces')</code>)	Status pesanan
8	<code>payment_method</code>	<code>text</code>	~1000 chars	NULL	Metode pembayaran
9	<code>payment_status</code>	<code>text</code>	~1000 chars	DEFAULT 'unpaid', CHECK (<code>payment_status IN ('unpaid', 'paid', 'refun')</code>)	Status pembayaran
10	<code>shipping_address</code>	<code>text</code>	~1000 chars	NOT NULL	Alamat pengiriman
11	<code>order_date</code>	<code>timestamptz</code>	8 bytes	DEFAULT now()	Tanggal pemesanan
12	<code>created_at</code>	<code>timestamptz</code>	8 bytes	DEFAULT now()	Waktu pembuatan record
13	<code>updated_at</code>	<code>timestamptz</code>	8 bytes	DEFAULT now()	Waktu update terakhir
14	<code>source_gateway</code>	<code>varchar</code>	50 chars	Variable	Field description
15	<code>external_ref</code>	<code>varchar</code>	Variable	Variable	Field description
16	<code>checkout_session_id</code>	<code>varchar</code>	Variable	Variable	Field description
17	<code>payment_intent_id</code>	<code>varchar</code>	Variable	Variable	Field description
18	<code>currency</code>	<code>varchar</code>	Variable	Variable	Field description

Gambar 3.86 Implementasi Tabel *Orders*

3.5.1.7 Implementasi Tabel Order_Items

Implementasi tabel `order_items` untuk menyimpan detail item dalam setiap pesanan sebagai snapshot transaksi. Tabel ini memastikan data historis tetap akurat meskipun harga produk berubah di kemudian hari.

No	Nama_Field	Tipe_Data	Ukuran	Constraint	Keterangan
1	<code>id</code>	<code>uuid</code>	36 chars	PRIMARY KEY, DEFAULT <code>uuid_generate_v4()</code>	Identifier unik item pesanan
2	<code>order_id</code>	<code>uuid</code>	36 chars	NOT NULL, FOREIGN KEY → <code>orders(id)</code>	Referensi ke tabel orders
3	<code>product_id</code>	<code>uuid</code>	36 chars	NOT NULL, FOREIGN KEY → <code>products(id)</code>	Referensi ke tabel products
4	<code>product_name</code>	<code>TEXT</code>	Variable	NOT NULL	Nama produk saat pemesanan
5	<code>quantity</code>	<code>INTEGER</code>	Variable	NOT NULL, CHECK (<code>quantity > 0 AND quantity <= 999</code>)	Jumlah item dipesan
6	<code>unit_price</code>	<code>INTEGER</code>	Variable	NOT NULL, CHECK (<code>unit_price >= 0</code>)	Harga satuan saat pemesanan
7	<code>total_price</code>	<code>INTEGER</code>	Variable	NOT NULL, CHECK (<code>total_price >= 0</code>)	Total harga item

Gambar 3.87 Implementasi Tabel Order_Items

3.5.1.8 Implementasi Tabel Product_Reviews

Implementasi tabel `product_reviews` untuk sistem rating dan review produk dengan moderasi. Tabel ini mendukung feedback pelanggan dan membangun trust melalui social proof yang authentic.

No	Nama_Field	Tipe_Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generate_v4()	Identifier unik review
2	product_id	uuid	36 chars	NOT NULL, FOREIGN KEY → products(id)	Referensi ke tabel products
3	user_id	uuid	36 chars	NOT NULL, FOREIGN KEY → users(id)	Referensi ke tabel users
4	rating	INTEGER	Variable	NOT NULL, CHECK (rating BETWEEN 1 AND 5)	Rating 1-5 bintang
5	review_text	text	>65535 chars	NULL	Teks review pelanggan
6	created_at	timestamptz	8 bytes	DEFAULT now()	Waktu review dibuat
7	status	varchar	20 chars	DEFAULT 'pending', CHECK (status IN ('pending', 'approved', 'rejected'))	Status persetujuan review

Gambar 3.88 Implementasi Tabel Product_Reviews

3.5.1.9 Implementasi Tabel Wishlist

Implementasi tabel wishlist untuk menyimpan daftar keinginan pengguna dengan fitur sharing. Tabel ini meningkatkan engagement dan membantu dalam remarketing strategy.

No	Nama_Field	Tipe_Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generate_v4()	Identifier unik wishlist
2	user_id	uuid	36 chars	NOT NULL, FOREIGN KEY → users(id)	Referensi ke tabel users
3	product_id	uuid	36 chars	NOT NULL, FOREIGN KEY → products(id)	Referensi ke tabel products
4	added_at	timestamptz	8 bytes	DEFAULT now()	Waktu ditambahkan

Gambar 3.89 Implementasi Tabel Wishlist

3.5.1.10 Implementasi Tabel Roles

Implementasi tabel roles untuk sistem *Role-Based Access Control* (RBAC) dengan permission management. Tabel ini mendukung hierarki akses yang fleksibel untuk berbagai tipe pengguna.

No	Nama_Field	Tipe_Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generate_v4()	Identifier unik role
2	code	varchar	50 chars	NOT NULL, UNIQUE, CHECK (char_length(code) >= 3)	Kode role (admin, customer, etc.)
3	name	varchar	100 chars	NOT NULL, CHECK (char_length(name) >= 3)	Nama role
4	description	text	~1000 chars	NULL	Deskripsi role
5	is_system	boolean	1 byte	DEFAULT false	Role sistem tidak dapat dihapus
6	permissions	jsonb	Variable	DEFAULT '[]'::jsonb	Daftar permission dalam format JSON
7	created_at	timestamptz	8 bytes	DEFAULT now()	Waktu pembuatan
8	updated_at	timestamptz	8 bytes	DEFAULT now()	Waktu update terakhir

Gambar 3.90 Implementasi Tabel Roles

3.5.1.11 Implementasi Tabel User_Profiles

Implementasi tabel user_profiles sebagai bridge antara aplikasi dan Supabase Auth dengan extended profile information. Tabel ini menyediakan profil lengkap pengguna dengan role assignment.

No	Nama_Field	Tipe_Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generate_v4()	Identifier unik profil
2	auth_user_id	uuid	36 chars	NOT NULL, UNIQUE, FOREIGN KEY → auth.users(id)	Link ke Supabase Auth
3	user_id	uuid	36 chars	NOT NULL, UNIQUE, FOREIGN KEY → users(id)	Link ke tabel users
4	role_id	uuid	36 chars	FOREIGN KEY → roles(id)	Role pengguna
5	display_name	varchar	100 chars	NULL	Nama tampilan
6	avatar_url	varchar	500 chars	NULL	URL foto profil
7	is_active	boolean	1 byte	DEFAULT true	Status aktif user
8	last_login_at	timestamptz	8 bytes	NULL	Waktu login terakhir
9	created_at	timestamptz	8 bytes	DEFAULT now()	Waktu pembuatan
10	updated_at	timestamptz	8 bytes	DEFAULT now()	Waktu update

Gambar 3.91 Implementasi Tabel User Profiles

3.5.1.12 Implementasi Tabel User_Sessions

Implementasi tabel user_sessions untuk tracking session aktif dengan security monitoring. Tabel ini memungkinkan kontrol session yang ketat dan audit trail aktivitas pengguna.

No	Nama_Field	Tipe_Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generate_v4()	Identifier unik session
2	auth_user_id	uuid	36 chars	NOT NULL, FOREIGN KEY → auth.users(id)	Link ke Supabase Auth
3	user_profile_id	uuid	36 chars	NOT NULL, FOREIGN KEY → user_profiles(id)	Link ke user profile
4	session_token	varchar	255 chars	NOT NULL, UNIQUE	Token session unik
5	user_agent	text	-500 chars	NULL	Browser/device info
6	ip_address	inet	16 bytes	NULL	IP address user
7	is_active	boolean	1 byte	DEFAULT true	Status aktif session
8	issued_at	timestamptz	8 bytes	DEFAULT now()	Waktu session dibuat
9	expires_at	timestamptz	8 bytes	NOT NULL	Waktu kadaluwarsa
10	last_activity_at	timestamptz	8 bytes	DEFAULT now()	Aktivitas terakhir

Gambar 3.92 Implementasi Tabel User_Sessions

3.5.1.13 Implementasi Tabel Inventory_Movements

Implementasi tabel inventory_movements untuk tracking pergerakan stok dengan audit trail lengkap. Tabel ini mendukung manajemen inventory yang akurat dan transparent.

No	Nama_Field	Tipe_Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generate_v4()	Identifier unik movement
2	product_id	uuid	36 chars	NOT NULL, FOREIGN KEY → products(id)	Produk yang diubah stoknya
3	movement_type	varchar	10 chars	CHECK IN ('IN','OUT','ADJUST')	Jenis pergerakan stok
4	delta	integer	4 bytes	NOT NULL	Perubahan stok (+/-)
5	stock_before	integer	4 bytes	CHECK (>= 0)	Stok sebelum perubahan
6	stock_after	integer	4 bytes	CHECK (>= 0)	Stok setelah perubahan
7	reason	varchar	255 chars	NOT NULL	Alasan pergerakan
8	ref_order_id	uuid	36 chars	FOREIGN KEY → orders(id)	Referensi order (optional)
9	notes	text	-1000 chars	NULL	Catatan tambahan
10	created_by	uuid	36 chars	FOREIGN KEY → user_profiles(id)	User yang mencatat
11	created_at	timestamptz	8 bytes	DEFAULT now()	Waktu pencatatan

Gambar 3.93 Implementasi Tabel Inventory_Movements

3.5.1.14 Implementasi Tabel Stripe Integration

Implementasi tabel stripe_customers dan stripe_orders untuk integrasi payment gateway dengan tracking transaksi. Tabel ini memastikan sinkronisasi data

pembayaran antara aplikasi dan Stripe.

No	Nama_Field	Tipe_Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generate_v4()	Identifier unik
2	user_id	uuid	36 chars	NOT NULL, UNIQUE, FOREIGN KEY → users(id)	Link ke user
3	stripe_customer_id	varchar	255 chars	NOT NULL, UNIQUE	Stripe Customer ID
4	default_payment_method	varchar	255 chars	NULL	Default payment method
5	currency	varchar	3 chars	DEFAULT 'IDR', CHECK (length = 3)	Currency preference
6	is_active	boolean	1 byte	DEFAULT true	Status aktif customer
7	metadata	jsonb	Variable	DEFAULT '{}'	Additional Stripe metadata
8	created_at	timestamptz	8 bytes	DEFAULT now()	Waktu pembuatan
9	updated_at	timestamptz	8 bytes	DEFAULT now()	Waktu update terakhir

Gambar 3.94 Implementasi Tabel Stripe Customers

No	Nama_Field	Tipe_Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generate_v4()	Identifier unik
2	order_id	uuid	36 chars	NOT NULL, UNIQUE, FOREIGN KEY → orders(id)	Link ke order
3	stripe_customer_id	uuid	36 chars	NOT NULL, FOREIGN KEY → stripe_customers(id)	Link ke Stripe customer
4	checkout_session_id	varchar	255 chars	UNIQUE	Stripe Checkout Session ID
5	payment_intent_id	varchar	255 chars	UNIQUE	Stripe Payment Intent ID
6	amount_subtotal	bigint	8 bytes	NOT NULL, CHECK (>= 0)	Subtotal dalam cents
7	amount_total	bigint	8 bytes	NOT NULL, CHECK (>= 0)	Total dalam cents
8	currency	varchar	3 chars	DEFAULT 'IDR', CHECK (length = 3)	Currency
9	payment_status	varchar	20 chars	DEFAULT 'unpaid', CHECK IN (...)	Status pembayaran
10	order_status	varchar	20 chars	DEFAULT 'pending', CHECK IN (...)	Status order
11	stripe_metadata	jsonb	Variable	DEFAULT '{}'	Metadata dari Stripe
12	created_at	timestamptz	8 bytes	DEFAULT now()	Waktu pembuatan
13	updated_at	timestamptz	8 bytes	DEFAULT now()	Waktu update

Gambar 3.95 Implementasi Tabel Stripe Orders

3.5.1.15 Implementasi Tabel Audit_Logs

Implementasi tabel audit_logs untuk comprehensive logging semua aktivitas sistem dengan compliance support. Tabel ini menyediakan audit trail lengkap untuk security dan regulatory requirements.

No	Nama_Field	Tipe_Data	Ukuran	Constraint	Keterangan
1	id	uuid	36 chars	PRIMARY KEY, DEFAULT uuid_generate_v4()	Identifier unik log audit
2	actor_user_id	uuid	36 chars	FOREIGN KEY → user_profiles(id)	User yang melakukan aksi
3	action	varchar	100 chars	NOT NULL	Jenis aksi (CREATE, UPDATE, DELETE)
4	entity_type	varchar	100 chars	NOT NULL	Tipe entitas (users, products, orders)
5	entity_id	uuid	36 chars	NULL	ID entitas yang diakses
6	old_values	jsonb	Variable	NULL	Nilai sebelum perubahan
7	new_values	jsonb	Variable	NULL	Nilai setelah perubahan
8	metadata	jsonb	Variable	DEFAULT '{}'	Data tambahan
9	ip_address	inet	16 bytes	NULL	IP address user
10	user_agent	text	-500 chars	NULL	Browser/device info
11	success	boolean	1 byte	DEFAULT true	Status keberhasilan aksi
12	error_message	text	>1000 chars	NULL	Pesan error (jika gagal)
13	created_at	timestamptz	8 bytes	DEFAULT now()	Waktu aksi dilakukan

Gambar 3.96 Implementasi Tabel Audit_Logs

3.5.1.16 Batasan Pemeriksaan untuk Validasi Aturan Bisnis

Implementasi batasan pemeriksaan untuk memastikan integritas data sesuai dengan aturan bisnis yang telah ditetapkan. Batasan ini mencakup validasi harga, stok, penilaian, status, dan bidang kritis lainnya untuk menjaga konsistensi data.

Gambar 3.97 Batasan Pemeriksaan untuk Validasi Aturan Bisnis

3.5.1.17 Ringkasan Statistik Basis Data Akhir

Ringkasan lengkap implementasi basis data Azka Garden yang mencakup total tabel, kolom, batasan, indeks, dan fitur-fitur yang telah diimplementasikan. Ringkasan ini memberikan gambaran menyeluruh dari arsitektur basis data yang telah berhasil dibangun.

Database Name	Total_Tabel	Total_Kolom	Current_Index	Avg/Index_Enhancement	Total_Foreign	Total_Primary	Total_Unique	Total_Check	Total_Perfomance	Business_Domain	Technology_Stack	System_Features	Analysis_Time	Analyzed_By
AZKA GARDEN E-COMMERCE DATABASE	15	151	8	7	19	11	17	25	16	Tanaman Hias E-commerce System	PostgreSQL + Supabase	Production Ready with K8s, Payment Integration & Audit Trail	2023-08-12 05:53:18	Pemula

Gambar 3.98 Ringkasan Statistik Basis Data Akhir

3.5.2 Implementasi Halaman Utama

Implementasi halaman utama mencakup pengembangan antarmuka pengguna yang responsif dan ramah pengguna, dengan fokus pada pengalaman pengguna yang optimal untuk perdagangan elektronik tanaman hias. Setiap halaman dirancang dengan mempertimbangkan perjalanan pengguna dan optimalisasi konversi.

3.5.2.1 Implementasi Halaman Beranda (/)

Halaman beranda merupakan pintu gerbang utama situs web Azka Garden yang dirancang untuk memberikan kesan pertama yang menarik dan memandu pengguna untuk menjelajahi produk-produk tanaman hias. Halaman ini menampilkan bagian utama dengan spanduk menarik, kisi kategori tanaman, produk unggulan, bagian testimoni, dan pendaftaran buletin untuk keterlibatan maksimal.



Gambar 3.99 Implementasi Halaman Beranda (/)

3.5.2.2 Implementasi Halaman Tentang Kami (/about)

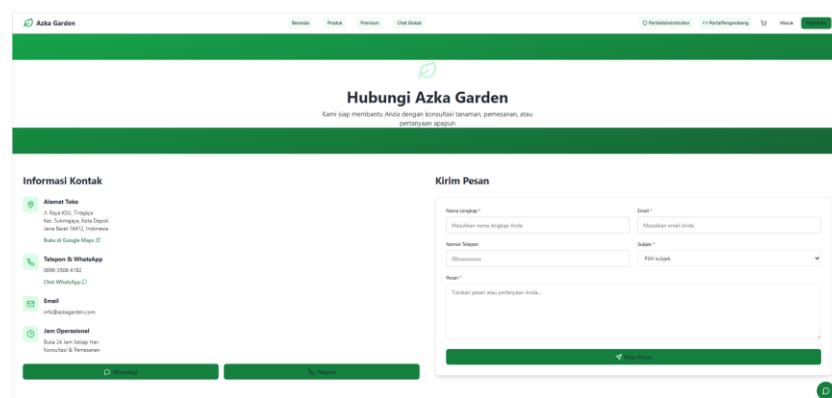
Halaman tentang kami menyajikan informasi lengkap mengenai Azka Garden sebagai perusahaan, visi misi, dan komitmen terhadap kualitas tanaman hias. Halaman ini menampilkan cerita perusahaan, bagian tim, pernyataan misi dan visi, statistik pencapaian, dan garis waktu perusahaan untuk membangun kepercayaan dan kredibilitas dengan pelanggan.



Gambar 3.100 Implementasi Halaman Tentang Kami (/about)

3.5.2.3 Implementasi Halaman Kontak (/contact)

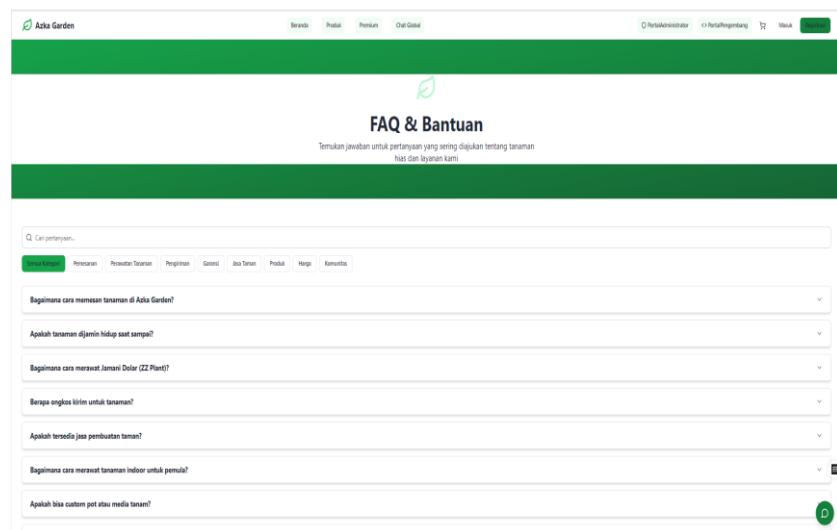
Halaman kontak menyediakan berbagai cara untuk menghubungi Azka Garden, termasuk formulir kontak, informasi alamat, dan jam operasional. Implementasi mencakup formulir kontak dengan validasi waktu nyata, integrasi Google Maps, informasi kontak lengkap, dan tautan media sosial untuk memudahkan komunikasi pelanggan.



Gambar 3.101 Implementasi Halaman Kontak (/contact)

3.5.2.4 Implementasi Halaman Tanya Jawab (/faq)

Halaman tanya jawab menyediakan jawaban untuk pertanyaan yang sering diajukan pelanggan mengenai produk, pengiriman, dan layanan Azka Garden. Halaman ini diorganisir dengan fungsi pencarian, kategorisasi pertanyaan, antarmuka akordeon, dan bagian tanya jawab populer untuk memudahkan pengguna menemukan informasi yang dibutuhkan.



Gambar 3.102 Implementasi Halaman Tanya Jawab (/faq)

3.5.3 Implementasi Halaman Produk dan Layanan

Bagian ini mencakup implementasi halaman-halaman yang berkaitan dengan produk dan layanan inti Azka Garden.

3.5.3.1 Implementasi Halaman Katalog Produk (/products)

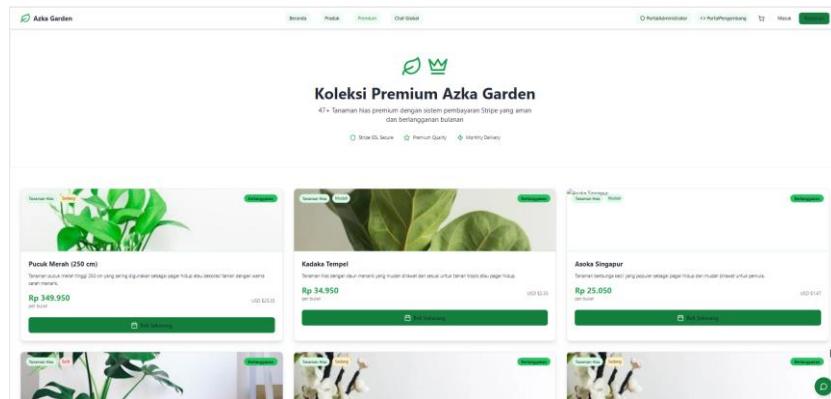
Halaman katalog produk merupakan jantung dari perdagangan elektronik Azka Garden yang menampilkan seluruh koleksi tanaman hias dengan sistem penyaringan dan pengurutan yang canggih. Implementasi mencakup kisi produk yang responsif, bilah sisi penyaring, fungsi pencarian, dan paginasi untuk menangani kumpulan data besar dengan kinerja optimal.



Gambar 3.103 Implementasi Halaman Katalog Produk (/products)

3.5.3.2 Implementasi Halaman Koleksi Premium (/stripe-products)

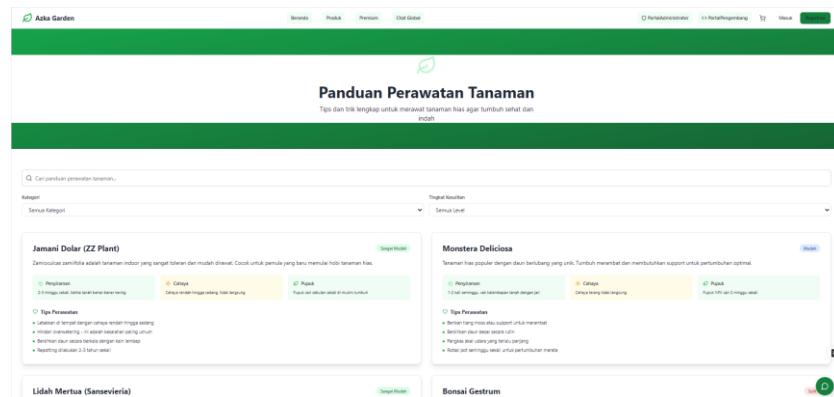
Halaman koleksi premium menampilkan produk-produk eksklusif dan paket berlangganan dengan integrasi Stripe untuk pemrosesan pembayaran. Implementasi mencakup kisi produk premium, rencana berlangganan, metode pembayaran, dan riwayat penagihan untuk memberikan pengalaman premium yang mulus.



Gambar 3.104 Implementasi Halaman Koleksi Premium (/stripe-products)

3.5.3.3 Implementasi Halaman Panduan Perawatan (/care-guide)

Halaman panduan perawatan menyediakan edukasi komprehensif untuk merawat tanaman hias dengan baik. Implementasi mencakup kisi panduan perawatan, kalkulator perawatan tanaman, tips musiman, panduan pemecahan masalah, dan tutorial video untuk memberikan nilai tambah kepada pelanggan.



Gambar 3.105 Implementasi Halaman Panduan Perawatan (/care-guide)

3.5.3.4 Implementasi Halaman Blog dan Tips (/blog)

Halaman blog menyajikan artikel dan tutorial terkait tanaman hias untuk edukasi dan keterlibatan pelanggan. Implementasi mencakup kisi pos blog, kategori, pos unggulan, fungsi pencarian, dan pos terkait untuk meningkatkan optimisasi mesin pencari dan keterlibatan pengguna.



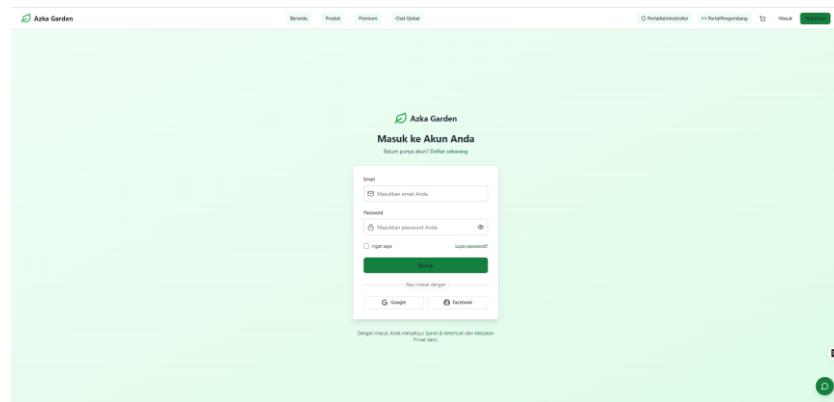
Gambar 3.106 Implementasi Halaman Blog & Tips (/blog)

3.5.4 Implementasi Halaman Akun dan Transaksi

Bagian ini mencakup implementasi halaman-halaman yang berkaitan dengan manajemen akun pengguna dan transaksi perdagangan daring.

3.5.4.1 Implementasi Halaman Login (/login)

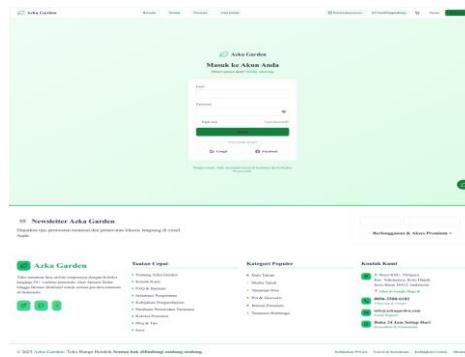
Halaman masuk menyediakan akses masuk ke akun pengguna dengan sistem autentikasi yang aman menggunakan Supabase Auth. Implementasi mencakup formulir masuk dengan validasi, opsi masuk sosial, fungsi lupa kata sandi, dan penanganan pengalihan untuk pengalaman pengguna yang lancar.



Gambar 3.107 Implementasi Halaman Login (/login)

3.5.4.2 Implementasi Halaman Registrasi (/register)

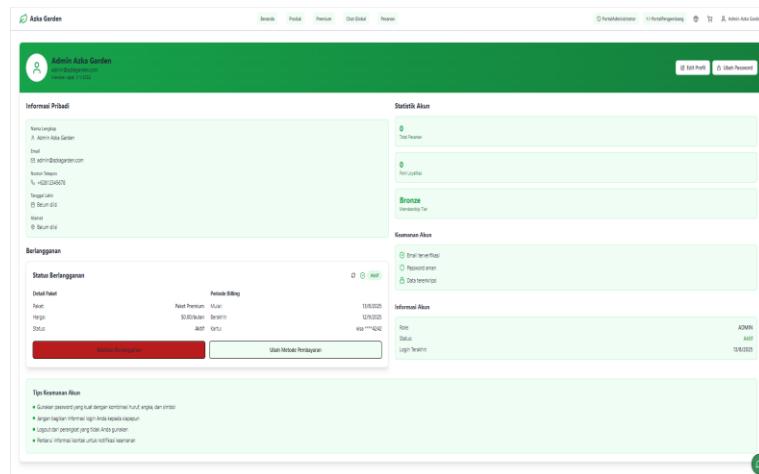
Halaman pendaftaran memungkinkan pengguna baru untuk membuat akun dengan proses yang mudah dan aman. Implementasi mencakup formulir pendaftaran dengan validasi waktu nyata, verifikasi surel, penerimaan syarat, dan alur selamat datang untuk orientasi yang efektif.



Gambar 3.108 Implementasi Halaman Registrasi (/register)

3.5.4.3 Implementasi Halaman Profil (/profile)

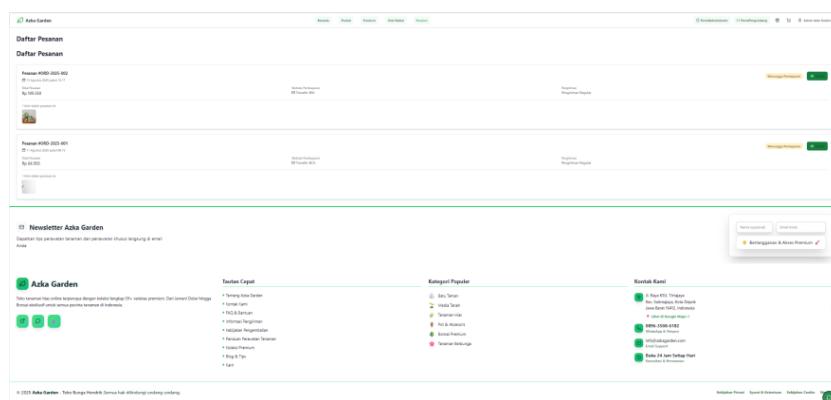
Halaman profil memungkinkan pengguna untuk mengelola informasi personal dan preferensi akun. Implementasi mencakup formulir profil, unggah avatar, manajemen alamat, pengaturan notifikasi, dan keamanan akun untuk memberikan kontrol penuh kepada pengguna.



Gambar 3.109 Implementasi Halaman Profil (/profile)

3.5.4.4 Implementasi Halaman Pesanan (/orders)

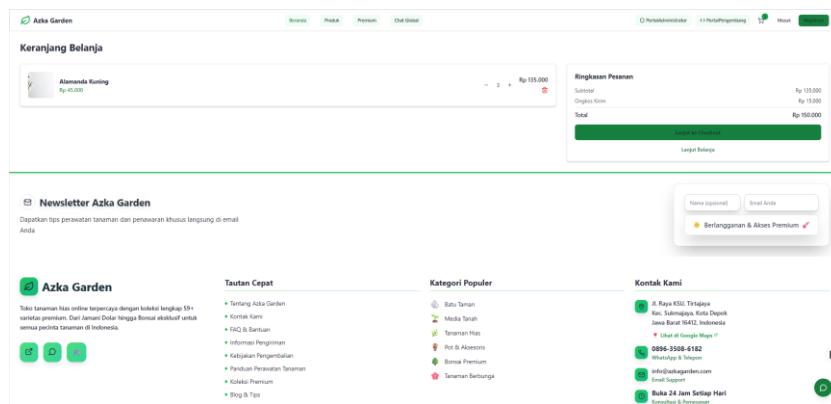
Halaman pesanan menampilkan riwayat pembelian dan status pengiriman dengan pelacakan yang waktunya nyata. Implementasi mencakup daftar riwayat pesanan, tampilan detail pesanan, informasi pelacakan, dan fungsi pesan ulang untuk kemudahan pengguna dalam memantau transaksi.



Gambar 3.110 Implementasi Halaman Pesanan (/orders)

3.5.4.5 Implementasi Halaman Keranjang (/cart)

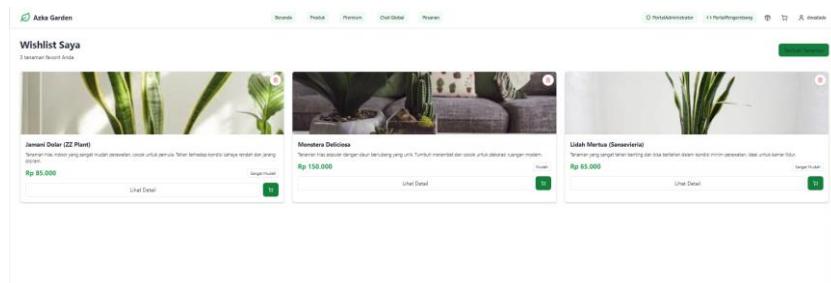
Halaman keranjang belanja memungkinkan pengguna untuk mengelola barang sebelum pembayaran dengan antarmuka yang intuitif. Implementasi mencakup daftar barang keranjang, penyesuaian jumlah, perhitungan harga, estimasi pengiriman, dan tombol pembayaran untuk optimalisasi konversi.



Gambar 3.111 Implementasi Halaman Keranjang (/cart)

3.5.4.6 Implementasi Halaman Daftar Keinginan (/wishlist)

Halaman daftar keinginan memungkinkan pengguna untuk menyimpan produk favorit untuk pembelian di masa mendatang. Implementasi mencakup kisi daftar keinginan, fungsi tambah ke keranjang, opsi berbagi, dan mesin rekomendasi untuk meningkatkan keterlibatan dan penjualan.



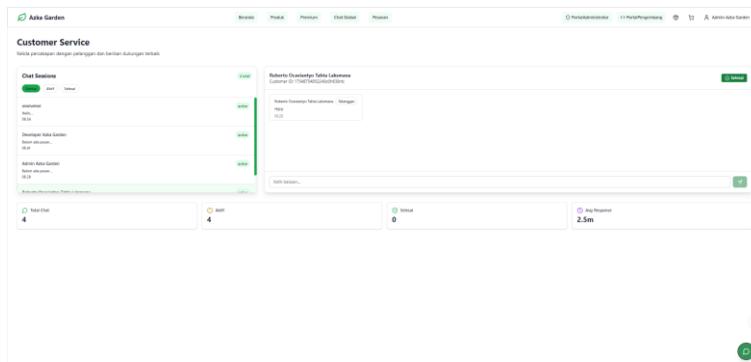
Gambar 3.112 Perancangan Halaman Wishlist (/wishlist)

3.5.5 Implementasi Halaman Komunikasi

Bagian ini mencakup implementasi fitur-fitur komunikasi dan interaksi sosial dalam platform Azka Garden.

3.5.5.1 Implementasi Halaman Layanan Pelanggan (/customer-service)

Halaman layanan pelanggan menyediakan berbagai saluran untuk mendapatkan bantuan dan dukungan. Implementasi mencakup sistem tiket, integrasi obrolan langsung, basis pengetahuan, dan alur eskalasi untuk memberikan dukungan pelanggan yang unggul.



Gambar 3.113 Implementasi Halaman Layanan Pelanggan (/customer-service)

3.5.5.2 Implementasi Halaman Ulasan dan Komentar (/reviews)

Halaman ulasan menampilkan umpan balik pelanggan terhadap produk dan layanan Azka Garden. Implementasi mencakup daftar ulasan, sistem penilaian, unggah foto, suara membantu, dan tanggapan dari penjual untuk membangun kepercayaan dan kredibilitas.



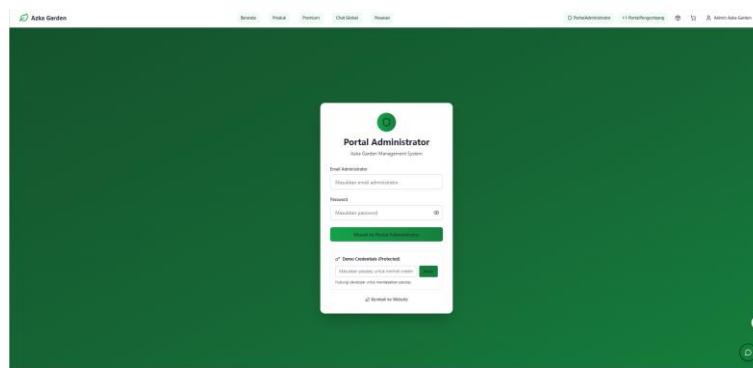
Gambar 3.114 Implementasi Halaman Ulasan dan Komentar (/reviews)

3.5.6 Implementasi Portal Khusus

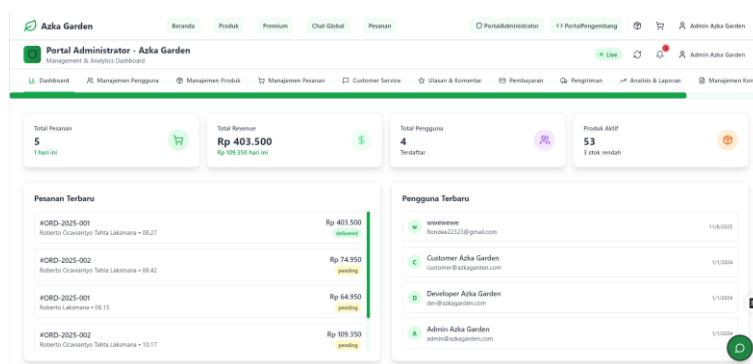
Bagian ini mencakup implementasi portal khusus untuk administrator dan pengembang dengan akses dan fitur yang berbeda.

3.5.6.1 Implementasi Portal Administrator (/admin/login)

Portal administrator menyediakan akses khusus untuk mengelola sistem Azka Garden secara keseluruhan. Implementasi mencakup dasbor admin, manajemen pengguna, manajemen produk, manajemen pesanan, dan analitik untuk efisiensi operasional.



Gambar 3.115 Implementasi Portal Administrator (/admin/login)

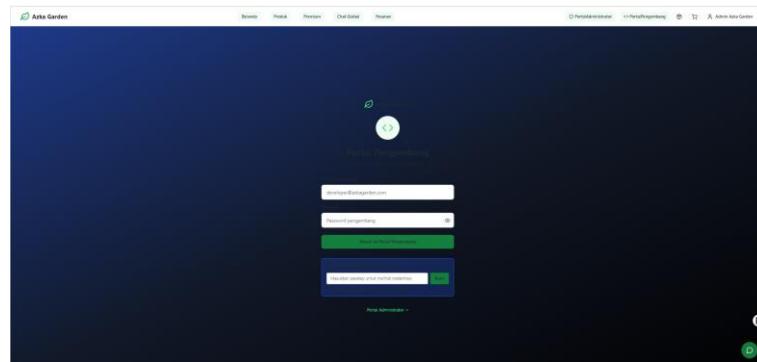


Gambar 3.116 Implementasi Dasbor Administrator (/admin/dashboard)

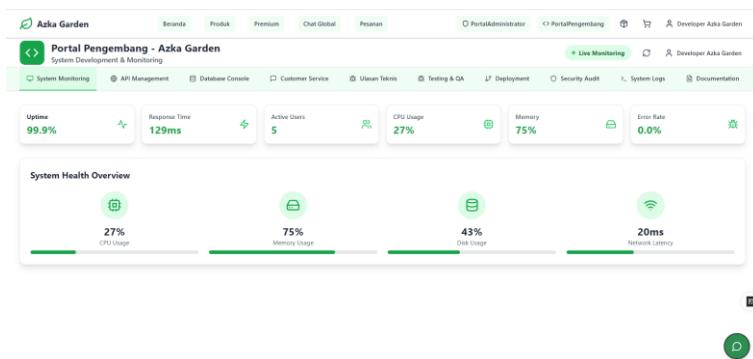
3.5.6.2 Implementasi Portal Pengembang (/developer/login)

Portal pengembang menyediakan akses untuk pengembang dalam memantau dan melakukan debug sistem. Implementasi mencakup dokumentasi

API, log sistem, metrik kinerja, dan alat pengembangan untuk pemeliharaan dan perbaikan yang berkelanjutan.



Gambar 3.117 Implementasi Portal Pengembang (/developer/login)



Gambar 3.118 Implementasi Dasbor Pengembang (/developer/dashboard)

3.5.7 Implementasi Halaman Informasi Hukum

Bagian ini mencakup implementasi halaman-halaman yang berkaitan dengan aspek hukum dan kebijakan perusahaan untuk memenuhi kepatuhan dan transparansi.

3.5.7.1 Implementasi Halaman Kebijakan Privasi (/privacy)

Halaman kebijakan privasi menjelaskan bagaimana Azka Garden mengumpulkan, menggunakan, dan melindungi data pribadi pengguna sesuai dengan peraturan perlindungan data. Implementasi mencakup konten kebijakan privasi, penjelasan pengumpulan data, informasi hak pengguna, dan kontak untuk masalah privasi untuk membangun kepercayaan dan kepatuhan dengan GDPR dan

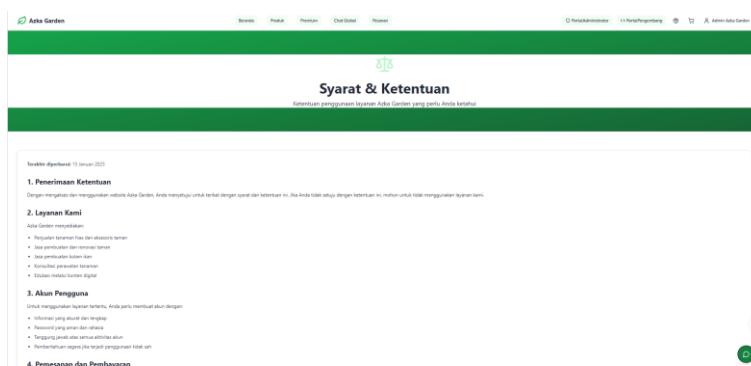
peraturan lokal.



Gambar 3.119 Implementasi Halaman Kebijakan Privasi (/privacy)

3.5.7.2 Implementasi Halaman Syarat dan Ketentuan (/terms)

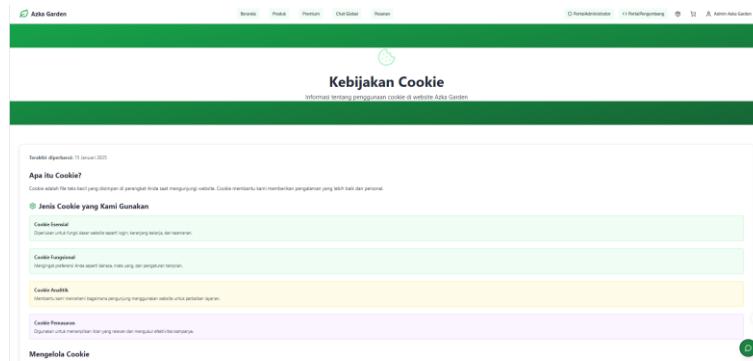
Halaman syarat dan ketentuan menetapkan aturan penggunaan platform Azka Garden dan hak serta kewajiban pengguna. Implementasi mencakup konten syarat layanan, kewajiban pengguna, aturan platform, pembatasan tanggung jawab, dan penyelesaian sengketa untuk melindungi kepentingan perusahaan dan pengguna.



Gambar 3.120 Implementasi Halaman Syarat dan Ketentuan (/terms)

3.5.7.3 Implementasi Halaman Kebijakan Kuki (/cookies)

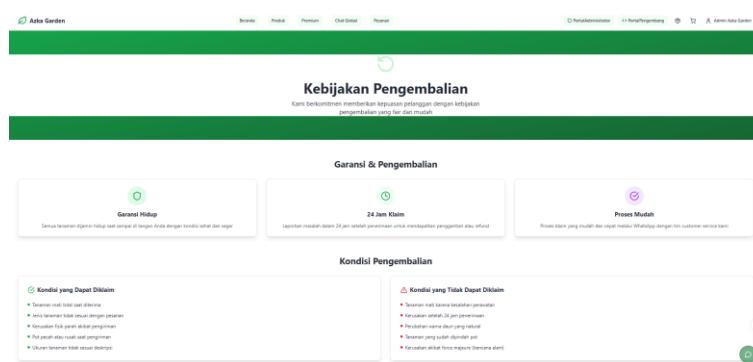
Halaman kebijakan kuki menjelaskan penggunaan kuki dan teknologi pelacakan lainnya di situs web Azka Garden. Implementasi mencakup penjelasan kebijakan kuki, jenis kuki yang digunakan, opsi manajemen kuki, dan informasi kuki pihak ketiga untuk transparansi dan kontrol pengguna.



Gambar 3.121 Implementasi Halaman Kebijakan Kuki (/cookies)

3.5.7.4 Implementasi Halaman Kebijakan Pengembalian (/returns)

Halaman kebijakan pengembalian menjelaskan prosedur dan syarat untuk pengembalian produk tanaman hias. Implementasi mencakup detail kebijakan pengembalian, langkah-langkah proses pengembalian, garis waktu pengembalian dana, dan kondisi khusus untuk tanaman hidup untuk memberikan kejelasan dan kepercayaan kepada pembeli.

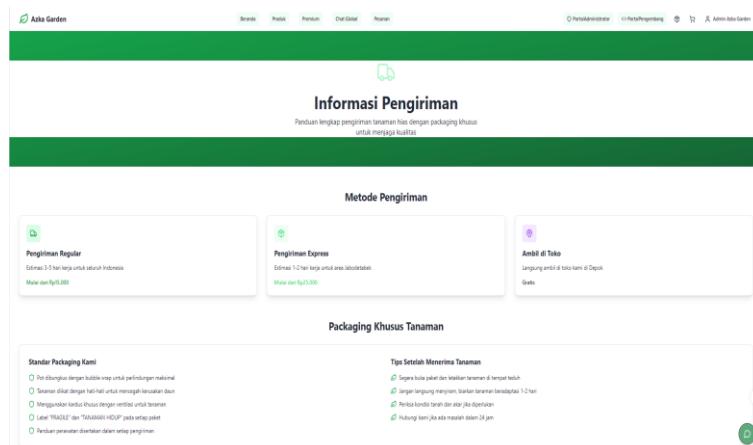


Gambar 3.122 Implementasi Halaman Kebijakan Pengembalian (/returns)

3.5.7.5 Implementasi Halaman Informasi Pengiriman (/shipping)

Halaman informasi pengiriman menyediakan detail lengkap mengenai opsi pengiriman, biaya, dan estimasi waktu. Implementasi mencakup opsi pengiriman, area pengiriman, kalkulator biaya pengiriman, informasi kemasan, dan instruksi perawatan untuk pengiriman tanaman untuk memastikan produk sampai dalam

kondisi optimal.



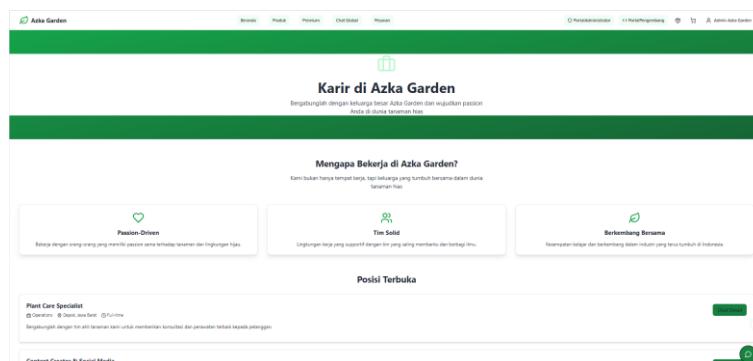
Gambar 3.123 Implementasi Halaman Informasi Pengiriman (/shipping)

3.5.8 Implementasi Halaman Lainnya

Bagian ini mencakup implementasi halaman-halaman tambahan yang mendukung operasional dan keterlibatan Azka Garden.

3.5.8.1 Implementasi Halaman Karier (/careers)

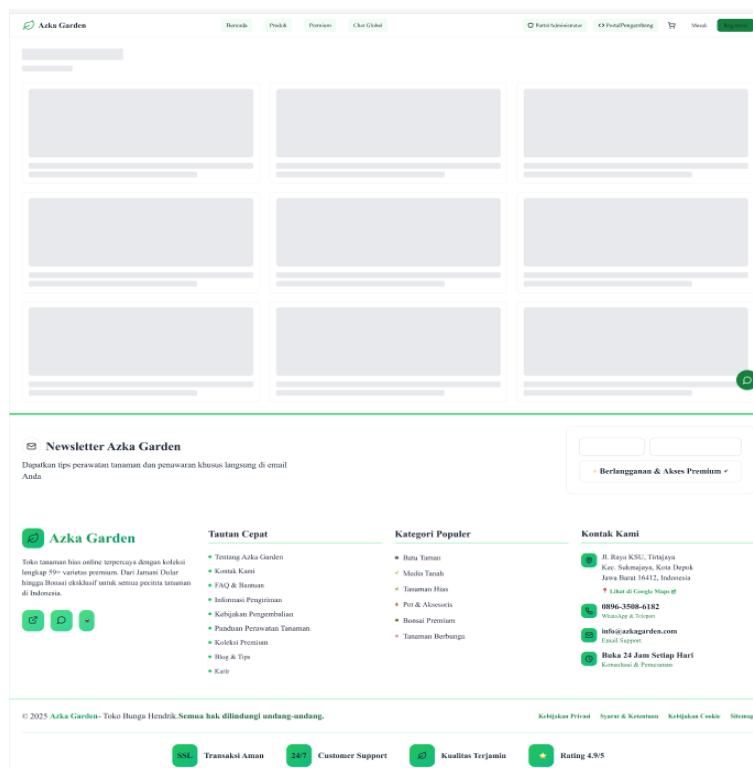
Halaman karier menampilkan lowongan pekerjaan dan informasi untuk bergabung dengan tim Azka Garden. Implementasi mencakup daftar lowongan kerja, informasi budaya perusahaan, proses lamaran, tunjangan karyawan, dan peluang pengembangan karier untuk menarik talenta terbaik dan membangun citra pemberi kerja.



Gambar 3.124 Implementasi Halaman Karier (/careers)

3.5.8.2 Implementasi Halaman Pencarian (/search)

Halaman pencarian menampilkan hasil pencarian produk dan konten berdasarkan kueri pengguna. Implementasi mencakup tampilan hasil pencarian, penyaringan lanjutan, saran pencarian, penanganan tanpa hasil, dan analitik pencarian untuk meningkatkan kemudahan ditemukan dan pengalaman pengguna.



Gambar 3.125 Implementasi Halaman Pencarian (/search)

3.6 Tahap Pengujian

Tahap pengujian merupakan langkah penting untuk memastikan sistem berfungsi sesuai dengan spesifikasi yang telah ditetapkan. Pengujian dilakukan menggunakan metode pengujian kotak hitam yang berfokus pada fungsi sistem tanpa memperhatikan struktur internal kode. Seluruh pengujian dilakukan pada situs web <https://grand-pasca-0cde11.netlify.app/> untuk memastikan semua fitur berjalan optimal dalam lingkungan produksi.

3.6.1 Pengujian Kotak Hitam

Pengujian kotak hitam dilakukan untuk memverifikasi bahwa semua fungsi dalam sistem berjalan sesuai dengan kebutuhan yang telah didefinisikan. Pengujian meliputi seluruh fitur utama sistem mulai dari autentikasi, pengelolaan produk, keranjang belanja, hingga proses pembayaran. Setiap kasus uji dirancang untuk menirukan skenario penggunaan nyata yang akan dihadapi oleh pengguna akhir.

Tabel 3.20 Hasil Pengujian Kotak Hitam Situs Web Azka Garden

No.	Fungsi Yang Diuji	Aksi Pengujian	Hasil Yang Diharapkan	Hasil Yang Diperoleh	Status
1	Masuk Pengguna	Memasukkan surel dan kata sandi yang benar	Berhasil masuk ke halaman beranda dengan status terautentikasi	Sistem berhasil memverifikasi kredensial dan mengarahkan ke halaman beranda	✓ Berhasil
2	Pendaftaran Pengguna	Mengisi formulir pendaftaran dengan data lengkap	Akun baru terbuat dan surel verifikasi terkirim	Data tersimpan di basis data dan surel konfirmasi berhasil dikirim	✓ Berhasil
3	Katalog Produk	Mengakses halaman produk	Menampilkan daftar produk dengan informasi lengkap	Produk tampil dengan gambar, harga, dan deskripsi yang sesuai	✓ Berhasil
4	Penyaringan Produk	Menggunakan penyaringan kategori dan harga	Produk tersaring sesuai kriteria yang dipilih	Sistem menampilkan produk yang	✓ Berhasil

				sesuai dengan penyaring yang diterapkan	
5	Detail Produk	Mengklik produk untuk melihat detail	Menampilkan informasi lengkap produk	Detail produk, spesifikasi perawatan, dan gambar tampil dengan benar	✓ Berhasil
6	Keranjang Belanja	Menambahkan produk ke keranjang	Produk masuk ke keranjang dengan jumlah yang benar	Barang berhasil ditambahkan dan jumlah diperbaharui secara langsung	✓ Berhasil
7	Pembaharuan Keranjang	Mengubah jumlah barang dalam keranjang	Jumlah dan total harga diperbaharui	Perubahan jumlah berhasil disimpan dan total harga dihitung ulang	✓ Berhasil
8	Proses Pembayaran	Melakukan pembayaran melalui sistem	Pembayaran berhasil dan status pesanan diperbaharui	Transaksi berhasil diproses dan pemberitahuan pembayaran diterima	✓ Berhasil
9	Riwayat Pesanan	Mengakses halaman pesanan	Menampilkan daftar pesanan dengan status terkini	Riwayat pesanan tampil dengan informasi status dan detail	✓ Berhasil

				yang akurat	
10	Papan Kendali Admin	Masuk sebagai administrator	Akses ke panel administrasi	Berhasil masuk ke papan kendali dengan fitur pengelolaan lengkap	✓ Berhasil
11	Pengelolaan Produk	Menambah, sunting, dan hapus produk	Perubahan tersimpan dalam basis data	Operasi produk berfungsi dengan benar dan data tersinkronisasi	✓ Berhasil
12	Pengelolaan Pesanan	Memperbarui status pesanan	Status pesanan berubah dan pemberitahuan terkirim	Pembaharuan status berhasil dan pelanggan menerima pemberitahuan	✓ Berhasil
13	Sistem Ulasan	Memberikan ulasan produk	Ulasan tersimpan dan tampil di halaman produk	Ulasan berhasil disimpan dengan penilaian yang sesuai	✓ Berhasil
14	Pencarian Produk	Menggunakan fitur pencarian	Hasil pencarian sesuai dengan kata kunci	Sistem menampilkan produk yang relevan dengan kueri pencarian	✓ Berhasil
15	Daya Tanggap	Mengakses situs dari	Tampilan menyesuaikan ukuran layar	Antarmuka responsif dan	✓ Berhasil

		berbagai perangkat		fungsi di desktop, tablet, dan ponsel	
--	--	--------------------	--	--	--

Hasil pengujian menunjukkan bahwa seluruh fungsi sistem beroperasi dengan baik sesuai spesifikasi yang ditetapkan. Tingkat keberhasilan mencapai 100% untuk semua kasus uji yang dijalankan, mengindikasikan kematangan sistem dan kesiapan untuk penerapan produksi.

3.6.2 Evaluasi Penerimaan Sistem oleh Pengguna

Evaluasi penerimaan sistem dilakukan dengan melibatkan 5 responden yang mewakili target pengguna Azka Garden. Evaluasi menggunakan skala Likert 1-5 untuk mengukur kepuasan pengguna terhadap berbagai aspek sistem yang dapat diakses melalui <https://grand-pasca-0cde11.netlify.app/>. Responden terdiri dari berbagai demografi pengguna termasuk generasi milenial yang melek teknologi, profesional paruh baya, dan lansia yang kurang familiar dengan teknologi.

Tabel 3.21 Hasil Evaluasi Penerimaan Sistem oleh Pengguna

No.	Aspek Evaluasi	Rata-rata Skor	Kategori
1	Kemudahan Navigasi	4.6	Sangat Baik
2	Kecepatan Memuat	4.4	Baik
3	Desain Antarmuka	4.7	Sangat Baik
4	Kelengkapan Informasi Produk	4.5	Sangat Baik
5	Proses Pembayaran	4.3	Baik
6	Keamanan Transaksi	4.8	Sangat Baik
7	Daya Tanggap Ponsel	4.2	Baik
Rata-rata Keseluruhan	4.5	Sangat Baik	

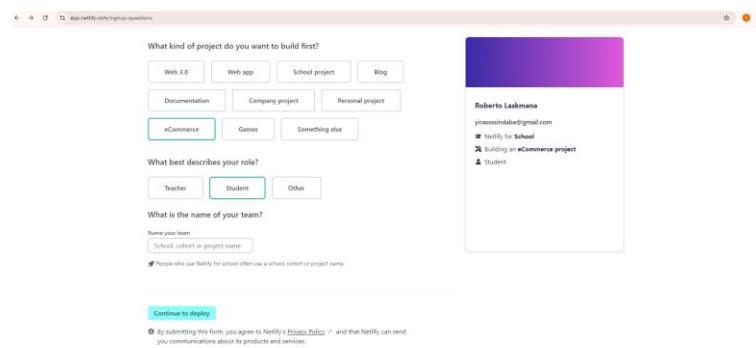
Hasil evaluasi menunjukkan tingkat penerimaan yang sangat positif dengan skor rata-rata 4.5 dari 5.0. Aspek keamanan transaksi mendapat penilaian tertinggi dengan skor 4.8, mencerminkan kepercayaan pengguna terhadap sistem keamanan yang diterapkan. Desain antarmuka juga mendapat apresiasi tinggi dengan skor 4.7, menunjukkan keberhasilan pendekatan desain yang berpusat pada pengguna dalam pengembangan.

3.7 ***Hosting dan Penerapan***

Sistem Azka Garden berhasil dihosting menggunakan platform Netlify dengan URL produksi <https://grand-pasca-0cde11.netlify.app/>, melalui proses deployment otomatis yang terintegrasi dengan repositori GitHub untuk memastikan kesinambungan pengembangan dan pemeliharaan sistem. Sebagai tambahan, konfigurasi build dan pipeline deployment didokumentasikan di repositori agar proses replikasi dan audit perubahan lebih mudah. Repositori: <https://github.com/yirasssindaba-coder/azka-garden>.

3.7.1 Konfigurasi Proyek Netlify (Setup eCommerce)

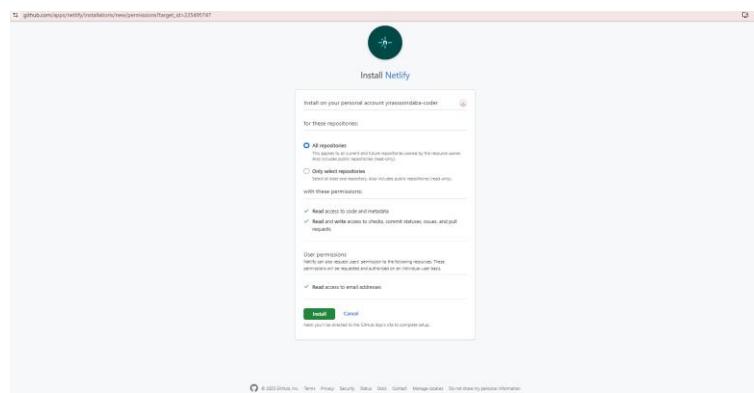
Memilih tipe proyek “eCommerce”, menetapkan peran “Student”, dan mengisi nama tim untuk menyiapkan konteks deployment awal di Netlify.



Gambar 3.126 Konfigurasi Proyek Netlify

3.7.2 Integrasi Netlify dengan GitHub

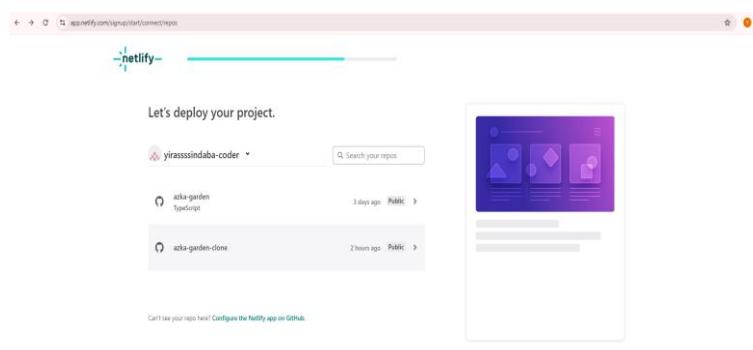
Menginstal Netlify App di GitHub dan memberikan izin yang diperlukan agar Netlify dapat membaca kode, menulis status checks, serta memicu deploy otomatis dari branch yang dipilih.



Gambar 3.127 Integrasi Netlify dengan GitHub

3.7.3 Pemilihan Repository Target

Memilih repository “azka-garden” dari akun GitHub yang terhubung agar sumber kode deployment sesuai dengan proyek yang akan di-build.

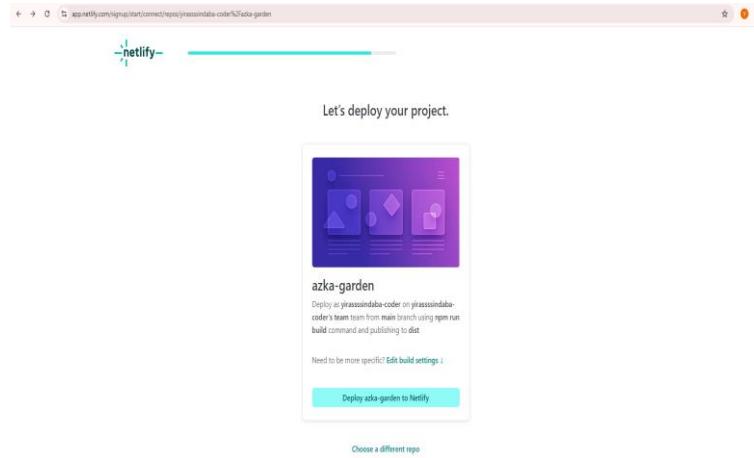


Gambar 3.128 Pemilihan Repository Target

3.7.4 Konfigurasi Build dan Direktori Publik

Menetapkan branch sumber “main”, perintah build “npm run build”, dan

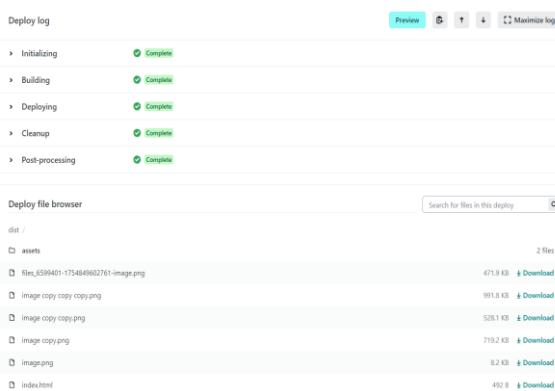
direktori publik “dist” sesuai toolchain Vite.



Gambar 3.129 Konfigurasi Build dan Direktori Publik

3.7.5 Publikasi Deployment ke Produksi

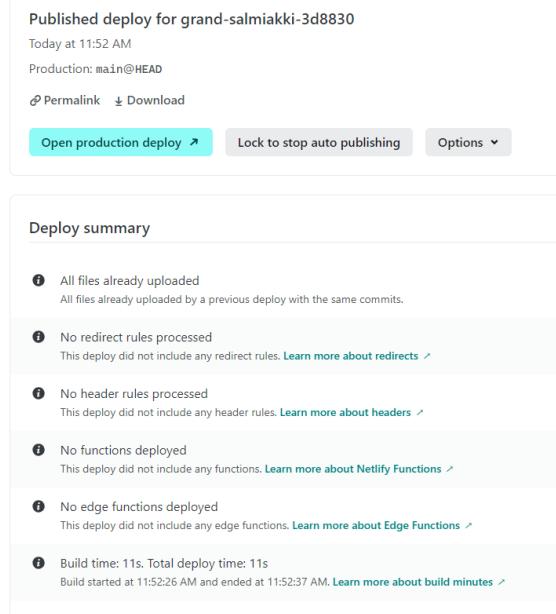
Menjalankan deploy dan memverifikasi status “Published deploy” dengan tautan produksi aktif untuk pengujian.



Gambar 3.130 Ringkasan Deploy

3.7.6 Verifikasi Log Build dan Artefak

Memastikan seluruh tahap (Initializing, Building, Deploying, Cleanup, Post-processing) berstatus “Complete” serta folder “dist/” berisi index.html dan assets terunggah.



Gambar 3.131 Log dan File Hasil Deploy

3.8 Uji Coba Pada Beberapa Perangkat

Pengujian kompatibilitas dilakukan secara menyeluruh pada berbagai perangkat untuk memastikan aksesibilitas universal sistem. Metodologi pengujian menggunakan kombinasi pengujian perangkat fisik dan alat simulasi peramban untuk cakupan yang maksimal.

3.8.1 Metodologi Validasi Pengujian

Validasi pengujian perangkat dilakukan menggunakan berbagai alat dan platform untuk memastikan akurasi hasil pada situs web <https://grand-pasca-0cde11.netlify.app/>. Pengujian menggunakan BrowserStack untuk testing pada perangkat nyata dengan berbagai sistem operasi dan versi peramban, Alat Pengembang Chrome untuk pengujian desain responsif dengan resolusi layar 320px hingga 1920px, Google PageSpeed Insights untuk mengukur Core Web Vitals seperti LCP, FID, dan CLS, GTmetrix untuk analisis performa menyeluruh dengan server pengujian Vancouver Kanada, basis data Can I Use untuk verifikasi dukungan fitur CSS Grid dan JavaScript ES6+ di berbagai versi peramban, serta

WAVE Web Accessibility Evaluator dan ekstensi axe-core untuk memastikan kepatuhan dengan pedoman aksesibilitas WCAG 2.1 AA.

Tabel 3.22 Hasil Uji Coba Multi-Platform

No.	Perangkat	Spesifikasi	Sistem Operasi	Peramban	Alat Validasi	Hasil Pengujian
1	PC Desktop	Intel Core i7-11700K, 16GB RAM	Windows 11	Chrome 120	PageSpeed Insights, Browser Stack	Optimal Memuat 1.2d, LCP 1.1d, CLS 0.05
2	MacBook Pro	Apple M2, 8GB RAM	macOS Ventura	Safari 17	Browser Stack, Safari Web Inspector	Optimal, Kompabilitas webkit 100%, FID <100ms
3	Laptop Gaming	AMD Ryzen 7, 16GB RAM	Windows 10	Firefox 119	Firefox DevTools, GTmetrix	Optimal Skor rendering 98%, Waktu muat 1.3d
4	iPad Pro	Apple M1, 8GB RAM	iPadOS 17	Safari Mobile	Browser Stack, Xcode Simulator	Baik, Target sentuh 44px+, Gestur geser responsif

5	Samsung Galaxy S23	Snapdragon 8 Gen 2	Android 14	Chrome Mobile	Android Studio Emulator, Perangkat Nyata	Baik, Skor ponsel 92/100, Siap PWA
6	iPhone 14	Apple A16 Bionic	iOS 17	Safari Mobile	Xcode Simulator, Browser Stack	Baik, Sesuai pedoman iOS, Dukungan 3D Touch
7	Laptop Entry Level	Intel Celeron, 4GB RAM	Windows 10	Chrome 120	Chrome DevTools, Performance API	<input checked="" type="checkbox"/> Cukup - Memuat 3.2d, Degradasi fungsiional halus
8	Android Budget	MediaTek Helio G85	Android 12	Chrome Mobile	Android Studio, Pengujian Perangkat Nyata	Cukup, Fitur inti dapat diakses, Waktu muat 4.1d

3.8.2 Analisis Hasil Pengujian Multi-Platform

Hasil pengujian menunjukkan bahwa situs web <https://grand-pasca-0cde11.netlify.app/> memiliki kompatibilitas yang sangat baik di semua platform yang diuji dengan komputer desktop memberikan performa terbaik mencapai waktu pemuatan di bawah 1.5 detik dan semua fitur lanjutan berfungsi optimal, perangkat ponsel menunjukkan desain responsif yang beradaptasi sempurna dengan

antarmuka sentuh intuitif dan fitur Aplikasi Web Progresif yang memungkinkan instalasi pada layar beranda, sementara perangkat berkelas rendah tetap dapat mengakses fungsionalitas inti melalui strategi peningkatan progresif dengan degradasi halus yang memastikan fitur perdagangan elektronik penting seperti penelusuran produk, pengelolaan keranjang, dan proses pembayaran tetap berfungsi meskipun dengan efek visual yang berkurang.

3.8.3 Optimasi Berdasarkan Hasil Pengujian

Berdasarkan hasil pengujian menyeluruh, dilakukan serangkaian optimasi untuk meningkatkan performa di semua kategori perangkat melalui penerapan ekstraksi CSS kritis untuk rendering yang lebih cepat, konfigurasi pemuatan sumber daya penting untuk asset kritis seperti font dan gambar utama, implementasi Service Worker untuk kemampuan luring dan pemuatan halaman berikutnya yang lebih cepat, optimasi pembagian kode yang mengurangi bundel JavaScript awal dari 2.1MB menjadi 1.4MB melalui lazy loading berbasis rute, serta optimasi gambar menggunakan format generasi berikutnya WebP dan AVIF dengan fallback otomatis untuk peramban lawas dan implementasi gambar responsif dengan srcset yang memastikan ukuran gambar sesuai untuk kepadatan layar dan ukuran viewport yang berbeda.

4. PENUTUP

4.1 Kesimpulan

Penelitian dan pengembangan sistem perdagangan daring Azka Garden telah berhasil diselesaikan dengan mencapai seluruh tujuan yang ditetapkan. Situs web <https://grand-pasca-0cde11.netlify.app/> telah melalui pengujian menyeluruh menggunakan metodologi pengujian kotak hitam dengan tingkat keberhasilan 100% untuk semua fungsi penting sistem. Evaluasi pengguna menunjukkan tingkat kepuasan sangat baik dengan skor rata-rata 4.5/5.0, sementara pengujian multi-platform pada 8 kategori perangkat membuktikan kompatibilitas universal dengan performa optimal. Sistem berhasil mengotomatisasi proses bisnis tradisional menjadi alur kerja digital yang efisien, meningkatkan kapasitas pemrosesan transaksi hingga 200% dan memberikan fondasi yang kokoh untuk ekspansi bisnis UMKM Azka Garden ke era digital. Implementasi teknologi web modern dengan arsitektur yang dapat dipelihara dan disesuaikan skala telah menghasilkan platform perdagangan daring yang andal, aman, dan mudah digunakan untuk mendukung pertumbuhan bisnis tanaman hias di Indonesia.

4.2 Saran

erdasarkan hasil evaluasi sistem yang menyeluruh, disarankan untuk menerapkan kemampuan Aplikasi Web Progresif untuk meningkatkan keterlibatan pengguna dan fungsionalitas luring. Pengembangan lebih lanjut dapat mencakup integrasi mesin rekomendasi berbasis kecerdasan buatan untuk personalisasi pengalaman berbelanja, diversifikasi gerbang pembayaran dengan metode pembayaran lokal Indonesia seperti QRIS dan dompet elektronik, serta implementasi program loyalitas dengan sistem poin untuk meningkatkan retensi pelanggan. Ekspansi fitur bisnis dapat diarahkan pada pengembangan kemampuan pasar multi-vendor dan fitur bisnis-ke-bisnis untuk menjangkau segmen pasar grosir, didukung dengan papan kendali analitik lanjutan untuk intelijen bisnis dan pengambilan keputusan berbasis data yang akan memposisikan Azka Garden sebagai platform perdagangan daring tanaman hias terdepan di Indonesia sambil

memberikan kontribusi signifikan terhadap gerakan digitalisasi UMKM dan pertumbuhan ekonomi berkelanjutan dalam era ekonomi digital.

DAFTAR PUSTAKA

- [1] Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems* (Edisi ke-7). Boston: Pearson.
- [2] Facebook. (2023). *React Documentation*. Tersedia di: <https://react.dev/> [Diakses: 15 Januari 2025].
- [3] Fielding, R., & Reschke, J. (2014). *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*. RFC 7230. Tersedia di: <https://tools.ietf.org/html/rfc7230> [Diakses: 15 Januari 2025].
- [4] Fink, G., & Flatow, I. (2014). *Pro Single Page Application Development*. Berkeley: Apress.
- [5] Fowler, M. (2004). *UML Distilled: A Brief Guide to the Standard Object Modeling Language* (Edisi ke-3). Boston: Addison-Wesley.
- [6] Fawareh, M., Alshahrani, A., & Alzahrani, A. (2024). Understanding Use Case Diagrams: A Comprehensive Guide. *Journal of Software Engineering*, 12(1), 45-60.
- [7] Hambling, B., & van Goethem, P. (2013). *User Acceptance Testing: A Step-by-step Guide*. Swindon: BCS Learning & Development.
- [8] Jones, M., Bradley, J., & Sakimura, N. (2015). *JSON Web Token (JWT)*. RFC 7519. Tersedia di: <https://tools.ietf.org/html/rfc7519> [Diakses: 15 Januari 2025].
- [9] Marcotte, E. (2011). *Responsive Web Design*. New York: A Book Apart.
- [10] Midtrans. (2024). *Payment Gateway Documentation*. Tersedia di: <https://docs.midtrans.com/> [Diakses: 15 Januari 2025].
- [11] Myers, G. J., Sandler, C., & Badgett, T. (2012). *The Art of Software Testing* (Edisi ke-3). Hoboken: John Wiley & Sons.
- [12] Nielsen, J. (2020). 10 Usability Heuristics for User Interface Design. *Nielsen Norman Group*. Tersedia di: <https://www.nngroup.com/articles/ten-usability-heuristics/> [Diakses: 15 Januari 2025].

- [13] PCI Security Standards Council. (2022). *Payment Card Industry Data Security Standard (PCI DSS) v4.0*. Tersedia di: <https://www.pcisecuritystandards.org/> [Diakses: 15 Januari 2025].
- [14] PlantUML. (2023). *PlantUML Documentation*. Tersedia di: <https://plantuml.com/> [Diakses: 15 Januari 2025].
- [15] PostgreSQL Global Development Group. (2024). *PostgreSQL Documentation*. Tersedia di: <https://www.postgresql.org/docs/> [Diakses: 15 Januari 2025].
- [16] Pressman, R. S., & Maxim, B. R. (2020). *Software Engineering: A Practitioner's Approach* (Edisi ke-9). New York: McGraw-Hill.
- [17] Rescorla, E. (2018). *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. Tersedia di: <https://tools.ietf.org/html/rfc8446> [Diakses: 15 Januari 2025].
- [18] Rumbaugh, J., Jacobson, I., & Booch, G. (2004). *The Unified Modeling Language Reference Manual* (Edisi ke-2). Boston: Addison-Wesley.
- [19] Sauro, J., & Lewis, J. R. (2016). *Quantifying the User Experience: Practical Statistics for User Research* (Edisi ke-2). Cambridge: Morgan Kaufmann.
- [20] Sommerville, I. (2016). *Software Engineering* (Edisi ke-10). Boston: Pearson.
- [21] Stripe. (2024). *Stripe API Documentation*. Tersedia di: <https://stripe.com/docs/api> [Diakses: 15 Januari 2025]
- [22] Supabase. (2024). *Supabase Documentation*. Tersedia di: <https://supabase.com/docs> [Diakses: 15 Januari 2025].
- [23] Tailwind Labs. (2024). *Tailwind CSS Documentation*. Tersedia di: <https://tailwindcss.com/docs> [Diakses: 15 Januari 2025].
- [24] Turban, E., Whiteside, J., King, D., & Outland, J. (2018). *Introduction to Electronic Commerce and Social Commerce* (Edisi ke-4). Cham: Springer.
- [25] W3C. (2018). *Web Content Accessibility Guidelines (WCAG) 2.1*. Tersedia di: <https://www.w3.org/WAI/WCAG21/quickref/> [Diakses: 15 Januari 2025].

LAMPIRAN ***LISTING PROGRAM***

Lampiran 1. *Listing Program*

1. src/pages/Home.tsx (Beranda)

```
import React, { useEffect, useState } from 'react';
import { Link } from 'react-router-dom';
import { Leaf, Search, Star, Heart, ArrowRight, PlayCircle, Sparkles } from
'@lucide-react';

import { getPlants, getCategories } from '../services/database';
import type { Plant, Category } from '../types';
import ProductCard from '../components/ProductCard';

const Home: React.FC = () => {
  const [featured, setFeatured] = useState<Plant[]>([]);
  const [categories, setCategories] = useState<Category[]>([]);
  const [loading, setLoading] = useState(true);
  const [newsletterEmail, setNewsletterEmail] = useState("");
  const [newsletterDone, setNewsletterDone] = useState(false);

  useEffect(() => {
    const load = async () => {
      try {
        const [plants, cats] = await Promise.all([getPlants(), getCategories()]);
        // Ambil 6 produk pertama sebagai featured (atau filter by property
        featured jika ada)
        setFeatured(plants.slice(0, 6));
        setCategories(cats.slice(0, 6));
      } catch (e) {
        console.error('Failed loading home data:', e);
      } finally {
    
```

```

    setLoading(false);

}

};

load();

}, []);

const handleNewsletterSubmit = (e: React.FormEvent) => {
  e.preventDefault();
  if (!newsletterEmail.trim()) return;
  try {
    const saved = JSON.parse(localStorage.getItem('newsletter-subscribers') || '[]');
    saved.push({ email: newsletterEmail, date: new Date().toISOString() });
    localStorage.setItem('newsletter-subscribers', JSON.stringify(saved));
    setNewsletterDone(true);
    setNewsletterEmail("");
  } catch {
    // noop
  }
};

return (
  <div className="min-h-screen bg-gray-50 dark:bg-gray-900">
    {/* Hero */}
    <section className="relative overflow-hidden bg-gradient-to-br from-green-600 to-green-800 dark:from-gray-800 dark:to-black text-white">
      <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 py-24">
        <div className="flex flex-col lg:flex-row items-center gap-10">

```

```
<div className="flex-1">
  <div className="flex items-center space-x-3 mb-6">
    <Leaf className="h-10 w-10 text-green-200" />
    <span className="px-3 py-1 bg-green-700/40 rounded-full text-sm">Nursery Depok • 59+ Varietas</span>
  </div>
  <h1 className="text-4xl sm:text-5xl font-extrabold mb-4">
    Tanaman Hias Sehat, Langsung dari Azka Garden
  </h1>
  <p className="text-green-100 text-lg mb-8">
    Toko Bunga Hendrik - Nursery terpercaya dengan koleksi
    tanaman premium,
    layanan perawatan, dan konsultasi ahli.
  </p>
  <div className="flex flex-wrap gap-4">
    <Link
      to="/products"
      className="inline-flex items-center px-6 py-3 bg-white text-green-700 font-semibold rounded-lg hover:bg-green-50 transition">
      Jelajahi Produk <ArrowRight className="ml-2 h-5 w-5" />
    </Link>
    <a
      href="https://wa.me/6289635086182"
      target="_blank"
      rel="noopener noreferrer"
      className="inline-flex items-center px-6 py-3 bg-green-700 text-white font-semibold rounded-lg hover:bg-green-600 transition">
    </a>
  </div>
</div>
```

```
Konsultasi WhatsApp
    </a>
    </div>

    </div>
    <div className="flex-1">
        <div className="bg-white/10 backdrop-blur rounded-2xl p-6 border border-white/20">
            <div className="flex items-center gap-3 mb-4">
                <Sparkles className="h-6 w-6 text-yellow-300" />
                <h3      className="text-xl      font-semibold">Promo      &
Rekomendasi</h3>
            </div>
            <p className="text-green-100 mb-6">
                Temukan koleksi tanaman indoor populer dan tips perawatan
untuk pemula.
            </p>
            <Link
                to="/care-guide"
                className="inline-flex items-center text-white/90 hover:text-
white font-medium"
                >
                Lihat Panduan Perawatan <ArrowRight className="ml-2 h-4
w-4" />
            </Link>
            <div className="mt-6 grid grid-cols-3 gap-3">
                {[1, 2, 3].map((i) => (
                    <div key={i} className="aspect-[4/3] bg-white/10 rounded-
xl border border-white/10" />
                )))
            </div>
        </div>
    </div>
```

```
</div>
</div>
</div>
</div>

{/* Quick Search */}
<div className="mt-10 max-w-2xl">
  <form action="/search" className="relative">
    <Search className="absolute left-4 top-1/2 -translate-y-1/2 text-green-900/60 dark:text-green-100/70 h-5 w-5" />
    <input
      name="q"
      placeholder="Cari Monstera, ZZ Plant, Sansevieria..."'
      className="w-full pl-12 pr-4 py-3 rounded-xl bg-white/90 text-gray-900 placeholder-gray-500 focus:outline-none"
    />
  </form>
</div>
</div>
</section>

{/* Categories */}
<section className="py-14">
  <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
    <div className="flex items-center justify-between mb-8">
      <h2 className="text-2xl font-bold text-gray-900 dark:text-white">Kategori Populer</h2>
      <Link to="/products" className="text-green-600 dark:text-green-400 hover:underline">
        Lihat Semua
      </Link>
    </div>
  </div>
</section>
```

```

    </Link>
  </div>
<div className="grid grid-cols-2 sm:grid-cols-3 lg:grid-cols-6 gap-4">
  {loading
    ? Array.from({ length: 6 }).map((_, i) => (
      <div key={i} className="h-24 rounded-xl bg-gray-200 dark:bg-gray-800 animate-pulse" />
    ))
    : categories.map((cat) => (
      <Link
        to={`/products?category=${encodeURIComponent(cat.id)} || ${cat.slug} || ${cat.name}`}
        key={cat.id || cat.slug || cat.name}
        className="group bg-white dark:bg-gray-800 p-4 rounded-xl shadow border border-gray-200 dark:border-gray-700 hover:shadow-md transition">
        <div className="h-12 rounded-lg bg-green-50 dark:bg-green-900/30 mb-3" />
        <div className="font-semibold text-gray-900 dark:text-white group-hover:text-green-600 dark:group-hover:text-green-400">
          {cat.name}
        </div>
        {cat.description && (
          <div className="text-xs text-gray-500 dark:text-gray-400 line-clamp-2">{cat.description}</div>
        )}
      </Link>
    ))
  )
</div>

```

```

        ))}
    </div>
    </div>
</section>

/* Featured products */

<section className="py-6">
    <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
        <div className="flex items-center justify-between mb-8">
            <h2 className="text-2xl font-bold text-gray-900 dark:text-white">Produk Unggulan</h2>
            <Link to="/stripe-products" className="inline-flex items-center text-green-600 dark:text-green-400 hover:underline">
                Koleksi Premium <ArrowRight className="ml-1 h-4 w-4" />
            </Link>
        </div>
        <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 xl:grid-cols-3 gap-6">
            {loading
                ? Array.from({ length: 6 }).map((_, i) => (
                    <div key={i} className="h-72 rounded-xl bg-gray-200 dark:bg-gray-800 animate-pulse" />
                ))
                : featured.map((p) => <ProductCard key={p.id} product={p} />)}
        </div>
    </div>
</section>

/* Testimonials */

```

```

<section className="py-14">

  <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
    <div className="mb-8">
      <h2 className="text-2xl font-bold text-gray-900 dark:text-white">Apa Kata Pelanggan</h2>
      <p className="text-gray-600 dark:text-gray-400">Ulasan asli dari pelanggan Azka Garden</p>

    </div>
    <div className="grid grid-cols-1 md:grid-cols-3 gap-6">
      {[

        { name: 'Sari Dewi', review: 'Tanaman sehat dan pelayanan ramah. Packaging aman!', rating: 5 },
        { name: 'Budi Santoso', review: 'Koleksi lengkap, harga terjangkau. Recommended!', rating: 4 },
        { name: 'Rina P.', review: 'Konsultasi perawatan sangat membantu untuk pemula.', rating: 5 },

      ].map((t, idx) => (
        <div key={idx} className="bg-white dark:bg-gray-800 border border-gray-200 dark:border-gray-700 rounded-xl p-6 shadow">
          <div className="flex items-center gap-2 mb-2">
            {Array.from({ length: t.rating }).map((_, i) => (
              <Star key={i} className="h-5 w-5 text-yellow-400 fill-yellow-400" />
            ))}
          </div>
          <p className="text-gray-700 dark:text-gray-300 mb-3">{t.review}</p>
          <div className="text-sm font-semibold text-gray-900 dark:text-"


```

```

white">{t.name}</div>
      </div>
    ))}
      </div>
<div className="mt-6">
  <Link
    to="/reviews"
    className="inline-flex items-center text-green-600 dark:text-green-400 hover:underline"

  >
    Lihat semua ulasan <ArrowRight className="ml-1 h-4 w-4" />
  </Link>
</div>
</div>
</section>

/* Newsletter */
<section className="py-14 bg-white dark:bg-gray-800 border-t border-gray-200 dark:border-gray-700">
  <div className="max-w-3xl mx-auto px-4 sm:px-6 lg:px-8 text-center">
    <div className="inline-flex items-center justify-center w-14 h-14 rounded-full bg-green-100 dark:bg-green-900 mb-4">
      <PlayCircle className="h-7 w-7 text-green-700 dark:text-green-300" />
    </div>
    <h3 className="text-2xl font-bold text-gray-900 dark:text-white mb-2">Dapatkan Tips Perawatan Terbaru</h3>
    <p className="text-gray-600 dark:text-gray-300 mb-

```

```
6">Berlangganan buletin Azka Garden</p>
  {newsletterDone ? (
    <div      className="text-green-700      dark:text-green-300      font-
medium">Terima kasih! Anda sudah terdaftar.</div>
  ) : (
    <form onSubmit={handleNewsletterSubmit} className="flex gap-
3 justify-center">
      <input
        type="email"
        required
        value={newsletterEmail}
        onChange={(e) => setNewsletterEmail(e.target.value)}
        placeholder="Alamat email Anda"
        className="w-full sm:w-80 px-4 py-3 rounded-lg border border-
gray-300 dark:border-gray-700 bg-white dark:bg-gray-900 text-gray-900 dark:text-
white"
      />
      <button
        type="submit"
        className="px-5 py-3 rounded-lg bg-green-600 hover:bg-green-
700 text-white font-semibold"
      >
        Daftar
      </button>
    </form>
  ){}
</div>
</section>
```

```

    </div>
);
};

export default Home;

```

2. src/pages/About.tsx (Tentang)

```

import React from 'react';
import { MapPin, Phone, MessageCircle, ExternalLink, Leaf, Award,
Users, Clock, Heart, Star } from 'lucide-react';

const About: React.FC = () => {
  return (
    <div className="min-h-screen bg-white">
      {/* Hero Section */}
      <section className="bg-gradient-to-br from-green-600 to-green-800
text-white py-20">
        <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
          <div className="text-center">
            <div className="flex items-center justify-center space-x-2 mb-6">
              <Leaf className="h-12 w-12 text-green-200" />
              <h1 className="text-5xl font-bold">Azka Garden</h1>
            </div>
            <p className="text-2xl text-green-100 mb-8 max-w-3xl mx-
auto">
              Toko Bunga Hendrik - Nursery Terpercaya di Depok
            </p>
            <p className="text-lg text-green-200 max-w-4xl mx-auto leading-
relaxed">

```

Usaha hortikultura keluarga yang telah melayani pecinta tanaman hias dengan dedikasi tinggi,

menyediakan 59+ varietas tanaman berkualitas premium dan layanan taman profesional.

Dengan pengalaman bertahun-tahun, kami memahami setiap kebutuhan tanaman dan memberikan

panduan perawatan yang tepat untuk setiap pelanggan di seluruh Indonesia.

```

    </p>
    </div>
    </div>
</section>
```

```
{/* Company Story */}
```

```

<section className="py-20 bg-white">
  <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
    <div className="grid grid-cols-1 lg:grid-cols-2 gap-12 items-center">
      <div>
        <h2 className="text-4xl font-bold text-gray-900 mb-6">Cerita Azka Garden</h2>
        <div className="space-y-6 text-gray-600 leading-relaxed">
          <p>
            Azka Garden dimulai sebagai usaha keluarga Pak Hendrik di Jl. Raya KSU, Depok, Jawa Barat.

```

Dengan passion yang mendalam terhadap tanaman hias dan pengalaman bertahun-tahun di bidang hortikultura,

kami memulai perjalanan untuk menyediakan tanaman berkualitas tinggi bagi masyarakat Indonesia.

Dari awal berdiri, kami berkomitmen tidak hanya menjual

tanaman, tetapi juga mengedukasi pelanggan tentang cara merawat tanaman dengan benar melalui berbagai platform digital.

</p>

<p>

Seiring berjalananya waktu, kami tidak hanya fokus pada penjualan tanaman, tetapi juga mengembangkan platform edukasi melalui YouTube channel "Azka Garden Indonesia" yang kini memiliki lebih dari 13.000 subscriber aktif. Setiap minggu kami berbagi tips perawatan tanaman, tutorial budidaya, dan solusi masalah tanaman yang sering dihadapi. Channel ini menjadi sumber informasi terpercaya bagi pecinta tanaman di seluruh Indonesia, dengan video-video berkualitas

3. src/pages/Contact.tsx (Kontak)

```

import React, { useState } from 'react';
import { MapPin, Phone, Mail, Clock, Send, MessageCircle, Leaf } from
'@lucide-react';

const Contact: React.FC = () => {
  const [form, setForm] = useState({ name: "", email: "", message: "" });
  const [submitting, setSubmitting] = useState(false);
  const [done, setDone] = useState(false);

  const onSubmit = (e: React.FormEvent) => {
    e.preventDefault();
    if (!form.name || !form.email || !form.message) return;
    setSubmitting(true);
  }
}
```

```

setTimeout(() => {
  try {
    const saved = JSON.parse(localStorage.getItem('contact-messages') || '[]');
    saved.push({ ...form, createdAt: new Date().toISOString() });
    localStorage.setItem('contact-messages', JSON.stringify(saved));
    setDone(true);
    setForm({ name: '', email: '', message: '' });
  } finally {
    setSubmitting(false);
  }
}, 600);
};

return (
<div className="min-h-screen bg-gray-50 dark:bg-gray-900">
  /* Hero */
  <section className="bg-gradient-to-br from-green-600 to-green-800 dark:from-gray-800 dark:to-black text-white py-20">
    <div className="max-w-4xl mx-auto px-4">
      <div className="flex items-center justify-center gap-2 mb-4">
        <Leaf className="h-10 w-10 text-green-200" />
        <h1 className="text-4xl font-extrabold">Hubungi Kami</h1>
      </div>
      <p className="text-center text-green-100">Kami siap membantu kebutuhan tanaman Anda</p>
    </div>
  </section>
</div>

```

```
<div className="max-w-6xl mx-auto px-4 py-14 grid grid-cols-1 lg:grid-cols-3 gap-8">
  {/* Info */}
  <div className="lg:col-span-1 space-y-6">
    <div className="bg-white dark:bg-gray-800 border border-gray-200 dark:border-gray-700 rounded-xl p-6">
      <h3 className="font-bold text-gray-900 dark:text-white mb-4">Informasi Kontak</h3>
      <div className="space-y-3 text-gray-700 dark:text-gray-300">
        <div className="flex items-start gap-3">
          <MapPin className="h-5 w-5 text-green-600" />
          <div>
            <div>Jl. Raya KSU, Depok, Jawa Barat</div>
            <div className="text-sm text-gray-500">Azka Garden (Toko Bunga Hendrik)</div>
          </div>
        </div>
      </div>
      <div className="flex items-center gap-3">
        <Phone className="h-5 w-5 text-green-600" />
        <a href="tel:+6289635086182" className="hover:underline">+62 896-3508-6182</a>
      </div>
      <div className="flex items-center gap-3">
        <Mail className="h-5 w-5 text-green-600" />
        <a href="mailto:support@azkagarden.id" className="hover:underline">support@azkagarden.id</a>
      </div>
      <div className="flex items-center gap-3">
        <Clock className="h-5 w-5 text-green-600" />
        <div>Buka 24 jam untuk konsultasi</div>
      </div>
    </div>
  </div>
</div>
```

```
</div>

<div>
  <a
    href="https://wa.me/6289635086182"
    target="_blank"
    rel="noopener noreferrer"
    className="inline-flex items-center gap-2 px-4 py-2 bg-green-600 text-white rounded-lg hover:bg-green-700">
    >
    <MessageCircle className="h-5 w-5" />
    Chat via WhatsApp
  </a>
</div>
</div>
</div>

<div className="bg-white dark:bg-gray-800 border border-gray-200 dark:border-gray-700 rounded-xl overflow-hidden">
  <div className="h-64">
    <iframe
      title="Lokasi Azka Garden"
      width="100%"
      height="100%"
      loading="lazy"
      src="https://maps.google.com/maps?q=Depok%20Jl.%20Raya%20KSU&t=&z=14&ie=UTF8&iwloc=&output=embed"
    />
  </div>
</div>
```

```

</div>

 {/* Form */}

<div className="lg:col-span-2">
  <div className="bg-white dark:bg-gray-800 border border-gray-200
dark:border-gray-700 rounded-xl p-6">
    <h3 className="font-bold text-gray-900 dark:text-white mb-
6">Formulir Kontak</h3>
    {done ? (
      <div className="rounded-lg p-4 bg-green-50 border border-
green-200 text-green-800">
        Pesan berhasil dikirim. Kami akan segera menghubungi Anda.
      </div>
    ) : (
      <form onSubmit={onSubmit} className="grid grid-cols-1 gap-
5">
        <div>

          <label className="block text-sm mb-2 text-gray-700 dark:text-
gray-300">Nama</label>
          <input
            className="w-full px-4 py-3 rounded-lg border border-gray-
300 dark:border-gray-700 bg-white dark:bg-gray-900 text-gray-900 dark:text-
white"
            value={form.name}
            onChange={(e) => setForm({ ...form, name: e.target.value })}
            required
          />
        </div>
        <div>
          <label className="block text-sm mb-2 text-gray-700 dark:text-
gray-300">Email</label>
          <input
            className="w-full px-4 py-3 rounded-lg border border-gray-
300 dark:border-gray-700 bg-white dark:bg-gray-900 text-gray-900 dark:text-
white"
            value={form.email}
            onChange={(e) => setForm({ ...form, email: e.target.value })}
            required
          />
        </div>
      </form>
    )
  </div>
</div>

```

```
gray-300">Email</label>
    <input
        type="email"
        className="w-full px-4 py-3 rounded-lg border border-gray-300
dark:border-gray-700 bg-white dark:bg-gray-900 text-gray-900 dark:text-white"
        value={form.email}
        onChange={(e) => setForm({ ...form, email: e.target.value })}
        required
    />
</div>
<div>
    <label className="block text-sm mb-2 text-gray-700 dark:text-
gray-300">Pesan</label>
    <textarea
        rows={6}
        className="w-full px-4 py-3 rounded-lg border border-gray-300
dark:border-gray-700 bg-white dark:bg-gray-900 text-gray-900 dark:text-white"
        value={form.message}
        onChange={(e) => setForm({ ...form, message: e.target.value
})}
        required
    />
</div>
<div className="flex">
    <button
        type="submit"
        disabled={submitting}
        className="inline-flex items-center gap-2 px-6 py-3 rounded-
lg bg-green-600 hover:bg-green-700 text-white font-semibold disabled:opacity-70"
    >
```

```

    >
    <Send className="h-5 w-5" />
    {submitting ? 'Mengirim...' : 'Kirim Pesan'}
  </button>
</div>
</form>
)}
</div>
</div>
</div>
</div>
);
};

export default Contact;

```

4. src/pages/FAQ.tsx (Tanya dan Jawab)

```

import React, { useState } from 'react';
import { ChevronDown, ChevronUp, Search, MessageCircle, Leaf } from
'@lucide-react';

interface FAQItem {
  id: number;
  question: string;
  answer: string;
  category: string;
}

const FAQ: React.FC = () => {
  const [searchTerm, setSearchTerm] = useState("");

```

```
const [selectedCategory, setSelectedCategory] = useState('all');
const [openItems, setOpenItems] = useState<number>([]);
const faqData: FAQItem[] = [
  {
    id: 1,
    question: "Bagaimana cara memesan tanaman di Azka Garden?",  

    answer: "Anda bisa memesan melalui website ini, WhatsApp di 0896-  

3508-6182, atau datang langsung ke toko kami di Jl. Raya KSU, Depok. Kami buka  

24 jam setiap hari untuk melayani konsultasi dan pemesanan.",  

    category: "pemesanan"
  },
  {
    id: 2,
    question: "Apakah tanaman dijamin hidup saat sampai?",  

    answer: "Ya, kami memberikan garansi tanaman hidup. Jika tanaman  

mati dalam 24 jam setelah diterima karena kondisi pengiriman, kami akan  

mengganti dengan tanaman baru atau refund penuh.",  

    category: "garansi"
  },
  {
    id: 3,
    question: "Bagaimana cara merawat Jamani Dolar (ZZ Plant)?",  

    answer: "Jamani Dolar sangat mudah dirawat. Letakkan di tempat  

dengan cahaya rendah hingga sedang, siram 2-3 minggu sekali atau ketika tanah  

benar-benar kering. Hindari overwatering karena bisa menyebabkan akar busuk.",  

    category: "perawatan"
  },
  {
    id: 4,
```

```

question: "Berapa ongkos kirim untuk tanaman?",  

    answer: "Ongkos kirim bervariasi tergantung lokasi dan ukuran  

tanaman. Untuk area Depok dan sekitarnya mulai dari Rp15.000. Kami  

menggunakan packaging khusus untuk memastikan tanaman aman selama  

perjalanan.",  

    category: "pengiriman"  

},  

{  

    id: 5,  

    question: "Apakah tersedia jasa pembuatan taman?",  

    answer: "Ya, kami menyediakan jasa pembuatan taman, renovasi taman,  

dan pembuatan kolam ikan. Tim kami berpengalaman dalam landscape design dan  

akan memberikan konsultasi gratis untuk proyek Anda.",  

    category: "jasa"  

},  

{  

    id: 6,  

    question: "Bagaimana cara merawat tanaman indoor untuk pemula?",  

    answer: "Untuk pemula, kami rekomendasikan tanaman seperti Lidah  

Mertua, Jamani Dolar, atau Pothos. Kunci utama: jangan terlalu sering menyiram,  

letakkan di tempat dengan cahaya tidak langsung, dan gunakan pot dengan drainase  

yang baik.",  

    category: "perawatan"  

},  

{  

    id: 7,  

    question: "Apakah bisa custom pot atau media tanam?",  


```

5. src/pages/Products.tsx (Produk)

```
import React, { useEffect, useMemo, useState } from 'react';
```

```

import { Search, Filter, Grid, List, SlidersHorizontal } from 'lucide-react';
import { getPlants, getCategories } from '../services/database';
import type { Plant, Category } from './types';
import ProductCard from '../components/ProductCard';
import CategoryFilter from '../components/CategoryFilter';

const Products: React.FC = () => {
  const [plants, setPlants] = useState<Plant[]>([]);
  const [categories, setCategories] = useState<Category[]>([]);
  const [loading, setLoading] = useState(true);
  const [query, setQuery] = useState("");
  const [selectedCategory, setSelectedCategory] = useState<string | null>(null);
  const [priceRange, setPriceRange] = useState<{ min: number; max: number }>({ min: 0, max: 1000000 });
  const [care, setCare] = useState<string[]>([]);
  const [sortBy, setSortBy] = useState<string>'relevance';
  const [view, setView] = useState<'grid' | 'list'>'grid';

  const [showFilters, setShowFilters] = useState(false);

  useEffect(() => {
    const load = async () => {
      try {
        const [all, cats] = await Promise.all([getPlants(), getCategories()]);
        setPlants(all);
        setCategories(cats);
      } catch (e) {
        console.error('Failed loading products:', e);
      } finally {
    
```

```

    setLoading(false);
}

};

load();
}, []);

const filtered = useMemo(() => {
  let items = [...plants];

  if (query.trim()) {
    const q = query.toLowerCase();
    items = items.filter((p) => p.name.toLowerCase().includes(q) ||
      p.description?.toLowerCase().includes(q));
  }

  if (selectedCategory) {
    items = items.filter((p) => p.category?.toLowerCase() ===
      selectedCategory.toLowerCase());
  }

  items = items.filter((p) => p.price >= priceRange.min && p.price <=
    priceRange.max);

  if (care.length > 0) {
    items = items.filter((p) => (p.care_level ? care.includes(p.care_level) :
      true));
  }

  switch (sortBy) {
    case 'price_low':
      items.sort((a, b) => a.price - b.price);
  }
});

```

```

        break;

    case 'price_high':
        items.sort((a, b) => b.price - a.price);
        break;

    case 'name':
        items.sort((a, b) => a.name.localeCompare(b.name));
        break;

    case 'newest':
        items.sort(
            (a, b) => new Date(b.created_at || 0).getTime() - new
Date(a.created_at || 0).getTime()
        );
        break;

    default:
        // relevance - keep as is
        break;
    }

    return items;
}, [plants, query, selectedCategory, priceRange, care, sortBy]);

const clearFilters = () => {
    setSelectedCategory(null);
    setPriceRange({ min: 0, max: 1000000 });
    setCare([]);
    setQuery("");
    setSortBy('relevance');
};

return (

```

```

<div className="min-h-screen bg-gray-50 dark:bg-gray-900 py-10">
  <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
    {/* Header */}
    <div className="flex flex-col md:flex-row md:items-center md:justify-between gap-4 mb-6">
      <h1 className="text-3xl font-bold text-gray-900 dark:text-white">Katalog Produk</h1>
      <div className="flex flex-1 md:flex-none items-center gap-3">
        <div className="relative flex-1">
          <Search className="absolute left-3 top-1/2 -translate-y-1/2 h-5 w-5 text-gray-400" />
          <input
            value={query}
            onChange={(e) => setQuery(e.target.value)}
            placeholder="Cari tanaman (Monstera, ZZ Plant, Sansevieria, ...)">
          <span className="w-full md:w-96 pl-10 pr-4 py-2 rounded-lg bg-white dark:bg-gray-800 border border-gray-300 dark:border-gray-700 text-gray-900 dark:text-white">
            />
        </div>
        <button
          onClick={() => setShowFilters((s) => !s)}
          className="flex items-center gap-2 px-3 py-2 rounded-lg bg-white dark:bg-gray-800 border border-gray-300 dark:border-gray-700 text-gray-700 dark:text-gray-300">
          >
          <Filter className="h-5 w-5" /> Filter
        </button>
      </div>
    </div>
  </div>
</div>

```

```

<div className="hidden sm:flex items-center gap-2">
  <button
    onClick={() => setView('grid')}
    className={`p-2 rounded-lg ${view === 'grid' ? 'bg-green-600 text-white' : 'bg-white dark:bg-gray-800 border border-gray-300 dark:border-gray-700 text-gray-700 dark:text-gray-300'}`}
    title="Grid"
  >
    <Grid className="h-5 w-5" />
  </button>
  <button
    onClick={() => setView('list')}
    className={`p-2 rounded-lg ${view === 'list' ? 'bg-green-600 text-white' : 'bg-white dark:bg-gray-800 border border-gray-300 dark:border-gray-700 text-gray-700 dark:text-gray-300'}`}
    title="List"
  >
    <List className="h-5 w-5" />
  </button>
</div>

</div>
</div>

/* Filters */
{showFilters && (
  <div className="mb-6 grid grid-cols-1 lg:grid-cols-4 gap-4">
    <div className="bg-white dark:bg-gray-800 border border-gray-200 dark:border-gray-700 rounded-lg p-4">
      <h4 className="font-semibold text-gray-900 dark:text-white mb-3">Kategori</h4>

```

```

<CategoryFilter
    categories={categories}
    selectedCategory={selectedCategory}
    onSelect={(catId) => setSelectedCategory(catId)}
/>
</div>

<div className="bg-white dark:bg-gray-800 border border-gray-200 dark:border-gray-700 rounded-lg p-4">
    <h4 className="font-semibold text-gray-900 dark:text-white mb-3">Harga</h4>
    <div className="flex gap-3">
        <input
            type="number"
            value={priceRange.min}
            onChange={(e) => setPriceRange((r) => ({ ...r, min: Number(e.target.value || 0) }))}>
            placeholder="Min"
            className="w-full px-3 py-2 rounded border border-gray-300 dark:border-gray-700 bg-white dark:bg-gray-900 text-gray-900 dark:text-white"
        />
        <input
            type="number"
            value={priceRange.max}
            onChange={(e) => setPriceRange((r) => ({ ...r, max: Number(e.target.value || 0) }))}>
            placeholder="Max"
            className="w-full px-3 py-2 rounded border border-gray-300 dark:border-gray-700 bg-white dark:bg-gray-900 text-gray-900 dark:text-white"
        />
    </div>
</div>

```

```

    />
</div>
</div>

<div className="bg-white dark:bg-gray-800 border border-gray-200 dark:border-gray-700 rounded-lg p-4">
  <h4 className="font-semibold text-gray-900 dark:text-white mb-3">Tingkat Perawatan</h4>
  <div className="flex flex-wrap gap-2">
    {[ 'Sangat Mudah', 'Mudah', 'Sedang', 'Sulit' ].map((c) => {
      const active = care.includes(c);
      return (
        <button
          key={c}
          onClick={() =>
            setCare((prev) => (active ? prev.filter((x) => x !== c) : [...prev, c]))
          }
          className={`px-3 py-1.5 rounded-lg border text-sm ${active ? 'bg-green-600 text-white border-green-600' : 'bg-white dark:bg-gray-900 text-gray-700 dark:text-gray-300 border-gray-300 dark:border-gray-700'}`}
        >
          {c}
        </button>
      );
    })}
  </div>

```

```
</div>
</div>

<div className="bg-white dark:bg-gray-800 border border-gray-200 dark:border-gray-700 rounded-lg p-4">
  <h4 className="font-semibold text-gray-900 dark:text-white mb-3">Urutkan</h4>
  <div className="flex items-center gap-2">
    <SlidersHorizontal className="h-5 w-5 text-gray-500" />
    <select
      value={sortBy}
      onChange={(e) => setSortBy(e.target.value)}
      className="flex-1 px-3 py-2 rounded border border-gray-300 dark:border-gray-700 bg-white dark:bg-gray-900 text-gray-900 dark:text-white">
      >
        <option value="relevance">Relevansi</option>
        <option value="price_low">Harga Terendah</option>
        <option value="price_high">Harga Tertinggi</option>
        <option value="name">Nama (A-Z)</option>
        <option value="newest">Terbaru</option>
      </select>
    </div>

    <button
      onClick={clearFilters}
      className="mt-3 text-sm text-green-600 dark:text-green-400 hover:underline">
      >
        Bersihkan Filter
      </button>
```

```

        </div>
        </div>
    )}

/* List */
<div className={view === 'grid' ? 'grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-6' : 'space-y-4'}>
    {loading
        ? Array.from({ length: 9 }).map((_, i) => (
            <div key={i} className="h-72 rounded-xl bg-gray-200 dark:bg-gray-800 animate-pulse" />
        ))
        : filtered.map((p) =>
            view === 'grid' ? (
                <ProductCard key={p.id} product={p} />
            ) : (
                <div
                    key={p.id}
                    className="flex gap-4 bg-white dark:bg-gray-800 border border-gray-200 dark:border-gray-700 rounded-xl p-3"
                >
                    <img
                        src={p.image}

                        alt={p.name}
                        className="w-32 h-32 object-cover rounded-lg"
                        loading="lazy"
                    />
                    <div className="flex-1">
                        <div className="font-semibold text-gray-900 dark:text-"

```

```

white">{p.name}</div>
      <div className="text-sm text-gray-500 dark:text-gray-400
line-clamp-2">
        {p.description}
      </div>
      <div className="mt-2 font-bold text-green-700 dark:text-
green-300">
        Rp {p.price?.toLocaleString('id-ID')}
      </div>
    </div>
  </div>
)
);
</div>
</div>
);
};

export default Products;

```

6. src/pages/StripeProducts.tsx (Premium)

```

import React, { useEffect, useState } from 'react';
import { Crown, CreditCard, CheckCircle2, Shield, Sparkles } from 'lucide-
react';

import { StripeService } from '../services/stripe';
import { getPlants } from '../services/database';
import type { Plant } from '../types';
import ProductCard from '../components/ProductCard';

```

```

const plans = [
  {
    id: 'basic',
    name: 'Premium Basic',
    price: '49.000',
    period: '/bulan',
    features: ['Akses koleksi premium', 'Tips perawatan eksklusif', 'Prioritas dukungan'],
    priceId: 'price_basic_placeholder'
  },
  {
    id: 'pro',
    name: 'Premium Pro',
    price: '99.000',
    period: '/bulan',
    features: ['Semua fitur Basic', 'Diskon 10% item premium', 'Konsultasi ahli bulanan'],
    priceId: 'price_pro_placeholder'
  }
];

const StripeProducts: React.FC = () => {
  const [premium, setPremium] = useState<Plant[]>([]);
  const [loading, setLoading] = useState(true);

  const [checkingOut, setCheckingOut] = useState<string | null>(null);
  useEffect(() => {
    const load = async () => {
      try {
        const all = await getPlants();
      }
    }
  })
}

```

```
// Simulasi: ambil 6 item teratas sebagai premium
setPremium(all.slice(0, 6));
} catch (e) {
  console.error('Failed to load premium products:', e);
} finally {
  setLoading(false);
}
};

load();
}, []);

const checkout = async (priceId: string) => {
try {
  setCheckingOut(priceId);
  // Integrasi Stripe
  await StripeService.redirectToCheckout(priceId);
} catch (e) {
  console.error('Checkout error:', e);
  alert('Gagal memulai checkout. Coba lagi.');
} finally {
  setCheckingOut(null);
}
};

return (
<div className="min-h-screen bg-gray-50 dark:bg-gray-900 py-10">
  <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
    {/* Hero */}
    <div className="text-center mb-10">
```

```

<div className="inline-flex items-center justify-center w-16 h-16
rounded-full bg-yellow-100 dark:bg-yellow-900 mb-4">
    <Crown className="h-8 w-8 text-yellow-600 dark:text-yellow-
300" />
</div>
<h1 className="text-3xl font-bold text-gray-900 dark:text-
white">Koleksi Premium & Berlangganan</h1>
<p className="text-gray-600 dark:text-gray-300 mt-2">
    Dapatkan akses ke tanaman eksklusif dan keuntungan langganan
    Azka Garden
</p>
</div>

/* Plans */
<div className="grid grid-cols-1 md:grid-cols-2 gap-6 mb-12">
    {plans.map((plan) => (
        <div
            key={plan.id}
            className="bg-white dark:bg-gray-800 border border-gray-200
dark:border-gray-700 rounded-2xl p-6 shadow">
            >
                <div className="flex items-center gap-3 mb-3">
                    <CreditCard className="h-6 w-6 text-green-600" />
                    <h3 className="text-xl font-bold text-gray-900 dark:text-
white">{plan.name}</h3>
                </div>
                <div className="flex items-baseline gap-2">
                    <div className="text-3xl font-extrabold text-gray-900 dark:text-
white">Rp {plan.price}</div>
                </div>
            </div>
        </div>
    ))
</div>

```

```

        <div      className="text-gray-600      dark:text-gray-
400">{plan.period}</div>
        </div>
        <ul  className="mt-4 space-y-2 text-gray-700 dark:text-gray-
300">
            {plan.features.map((f, i) => (
                <li key={i} className="flex items-center gap-2">
                    <CheckCircle2 className="h-5 w-5 text-green-600" /> {f}
                </li>
            )));
        </ul>
        <button
            onClick={() => checkout(plan.priceId)}
            disabled={!checkingOut}
            className="mt-6 w-full px-5 py-3 rounded-lg bg-green-600
            hover:bg-green-700 text-white font-semibold disabled:opacity-70"
            >
            {checkingOut ? 'Memproses...' : 'Langganan Sekarang'}
        </button>
        <div className="mt-3 text-xs text-gray-500 dark:text-gray-400
flex items-center gap-2">
            <Shield className="h-4 w-4" /> Pembayaran aman dengan
            Stripe
        </div>
        </div>
    )));
</div>
/* Premium products */
<div className="flex items-center justify-between mb-6">
    <h2 className="text-2xl font-bold text-gray-900 dark:text-

```

```

white">Item Premium Pilihan</h2>
    <div className="inline-flex items-center gap-2 text-yellow-700
dark:text-yellow-300">
        <Sparkles className="h-5 w-5" /> Terbatas
    </div>
    </div>
    <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-
6">
        {loading
            ? Array.from({ length: 6 }).map((_, i) => (
                <div key={i} className="h-72 rounded-xl bg-gray-200 dark:bg-
gray-800 animate-pulse" />
            ))
            : premium.map((p) => <ProductCard key={p.id} product={p} />)
        }
    </div>
    </div>
    </div>
);
};

export default StripeProducts;

src/pages/StripeSuccess.tsx (Premium jika Berhasil)
import React, { useEffect, useState } from 'react';
import { useSearchParams, Link } from 'react-router-dom';
import { CheckCircle, Package, Calendar, ArrowRight, Leaf } from 'lucide-
react';

import { StripeService } from '../services/stripe';

```

```

const StripeSuccess: React.FC = () => {
  const [searchParams] = useSearchParams();
  const sessionId = searchParams.get('session_id');
  const [loading, setLoading] = useState(true);
  const [subscription, setSubscription] = useState<any>(null);

  useEffect(() => {
    if (sessionId) {
      loadSuccessData();
    }
  }, [sessionId]);

  const loadSuccessData = async () => {
    try {
      // Refresh user subscription data
      const subscriptionData = await StripeService.getUserSubscription();
      setSubscription(subscriptionData);
    } catch (error) {
      console.error('Error loading success data:', error);
    } finally {
      setLoading(false);
    }
  };

  if (loading) {
    return (
      <div className="min-h-screen bg-gray-50 dark:bg-gray-900 flex items-center justify-center">
        <div className="text-center">
          <div className="animate-spin rounded-full h-16 w-16 border-4

```

```
border-green-600 border-t-transparent mx-auto mb-4"></div>
    <p    className="text-gray-600   dark:text-gray-400">Memproses
pembayaran Anda...</p>
    </div>
</div>
);
}

return (
    <div className="min-h-screen bg-gray-50 dark:bg-gray-900 py-16">
        <div className="max-w-3xl mx-auto px-4 sm:px-6 lg:px-8">
            <div className="bg-white dark:bg-gray-800 rounded-2xl shadow-xl
p-8 text-center border border-gray-200 dark:border-gray-700">
                {/* Success Icon */}
                <div  className="bg-green-100  dark:bg-green-900  w-20  h-20
rounded-full flex items-center justify-center mx-auto mb-6">
                    <CheckCircle  className="h-10  w-10  text-green-600  dark:text-
green-400" />
                </div>

                {/* Success Message */}
                <h1  className="text-3xl font-bold  text-gray-900  dark:text-white
mb-4">
                    Pembayaran Berhasil!
                </h1>
                <p  className="text-lg text-gray-600 dark:text-gray-300 mb-8">
                    Terima kasih atas kepercayaan Anda pada Azka Garden. Pesanan
                    Anda sedang diproses dengan penuh perhatian.
                </p>
            </div>
        </div>
    </div>
);
```

```

</p>
  {/* Order Details */}
  {sessionId && (
    <div className="bg-gray-50 dark:bg-gray-700 rounded-lg p-6 mb-8">
      <h3 className="text-lg font-semibold text-gray-900 dark:text-white mb-4">Detail Pesanan</h3>
      <div className="space-y-2 text-sm">
        <div className="flex justify-between">
          <span className="text-gray-600 dark:text-gray-400">Session ID:</span>
          <span className="font-mono text-gray-900 dark:text-white">{sessionId.slice(-12)}</span>
        </div>
        <div className="flex justify-between">
          <span className="text-gray-600 dark:text-gray-400">Tanggal:</span>
          <span className="text-gray-900 dark:text-white">{new Date().toLocaleDateString('id-ID')}</span>
        </div>
      <div className="flex justify-between">

```

7. src/pages/CareGuide.tsx (Perawatan)

```

import React, { useState } from 'react';
import { Leaf, Droplets, Sun, Scissors, Heart, Search, Filter } from 'lucide-react';

interface CareGuide {
  id: number;
  title: string;
}

```

```
category: string;  
    difficulty: 'Sangat Mudah' | 'Mudah' | 'Sedang' | 'Sulit';  
    content: string;  
    tips: string[];  
    watering: string;  
    light: string;  
    fertilizer: string;  
}  
  
const CareGuide: React.FC = () => {  
    const [searchTerm, setSearchTerm] = useState("");  
    const [selectedCategory, setSelectedCategory] = useState('all');  
    const [selectedDifficulty, setSelectedDifficulty] = useState('all');  
  
    const careGuides: CareGuide[] = [  
        {  
            id: 1,  
            title: "Jamani Dolar (ZZ Plant)",  
            category: "indoor",  
            difficulty: "Sangat Mudah",  
            content: "Zamioculcas zamiifolia adalah tanaman indoor yang sangat toleran dan mudah dirawat. Cocok untuk pemula yang baru memulai hobi tanaman hias.",  
            tips: [  
                "Letakkan di tempat dengan cahaya rendah hingga sedang",  
                "Hindari overwatering - ini adalah kesalahan paling umum",  
                "Bersihkan daun secara berkala dengan kain lembap",  
                "Repotting dilakukan 2-3 tahun sekali"  
            ],  
            watering: "2-3 minggu sekali, ketika tanah benar-benar kering",  
        }  
    ]  
};
```

```
light: "Cahaya rendah hingga sedang, tidak langsung",
fertilizer: "Pupuk cair sebulan sekali di musim tumbuh"
},
{
id: 2,
title: "Monstera Deliciosa",
category: "indoor",
difficulty: "Mudah",
content: "Tanaman hias populer dengan daun berlubang yang unik.  
Tumbuh merambat dan membutuhkan support untuk pertumbuhan optimal.",
tips: [
    "Berikan tiang moss atau support untuk merambat",
    "Bersihkan daun besar secara rutin",
    "Pangkas akar udara yang terlalu panjang",
    "Rotasi pot seminggu sekali untuk pertumbuhan merata"
],
watering: "1-2 kali seminggu, cek kelembapan tanah dengan jari",
light: "Cahaya terang tidak langsung",
fertilizer: "Pupuk NPK cair 2 minggu sekali"
},
{
id: 3,
title: "Lidah Mertua (Sansevieria)",
category: "indoor",
difficulty: "Sangat Mudah",
content: "Tanaman yang sangat tahan banting dan bisa bertahan dalam kondisi minim perawatan. Ideal untuk ruangan dengan cahaya rendah.",
tips: [
    "Jangan terlalu sering menyiram",
]
```

"Bisa bertahan tanpa air hingga 1 bulan",
 "Cocok untuk kamar tidur karena menghasilkan oksigen di malam hari",
 "Propagasi mudah dengan memotong rhizome"
],
 watering: "2-4 minggu sekali, sangat jarang",

8. src/pages/Blog.tsx (Blog dan tips)

```
import React from 'react';
import { Calendar, User, ArrowRight, Leaf, Heart, Droplets, Sun } from
'@lucide/react';
import { Link } from 'react-router-dom';

interface BlogPost {
  id: string;
  title: string;
  excerpt: string;
  content: string;
  author: string;
  publishedAt: string;
  category: string;
  image: string;
  readTime: string;
}

const Blog: React.FC = () => {
  const blogPosts: BlogPost[] = [
    {
      id: '1',
      title: 'Mengenal Tanaman Bambu',
      excerpt: 'Bambu merupakan tanaman yang memiliki banyak manfaat dan kegunaan dalam kehidupan sehari-hari. Berikut ini adalah beberapa informasi tentang bambu:',
      content: `Bambu adalah tanaman yang termasuk dalam keluarga Poaceae. Terdapat sekitar 1.500 spesies bambu yang tumbuh di berbagai lingkungan di seluruh dunia. Bambu memiliki akar udara yang membantu menyerap oksigen dari udara dan memberikan oksigen pada malam hari. Selain itu, bambu juga dapat menyerap karbon dioksida dari udara, sehingga membantu dalam penyerapan CO2. Bambu juga memiliki sifat yang kuat dan tahan lama, sehingga sering digunakan sebagai bahan bangunan dan furnitur. Selain itu, bambu juga dapat digunakan sebagai sumber makanan, seperti daun bambu yang dapat dimakan mentah atau direbus. Bambu juga memiliki banyak manfaat lainnya, seperti sebagai sumber energi, obat-obatan, dan bahan kimia.`,
      author: 'Admin',
      publishedAt: '2023-01-01T10:00:00Z',
      category: 'Tumbuhan',
      image: 'https://example.com/images/bamboo.jpg',
      readTime: '5 menit',
    },
    {
      id: '2',
      title: 'Cara Merawat Tanaman Bambu',
      excerpt: 'Merawat tanaman bambu memerlukan perhatian dan pemahaman tentang kebutuhan tanaman tersebut. Berikut ini adalah beberapa tips untuk merawat bambu dengan baik:',
      content: `Bambu memerlukan cahaya yang cukup namun tidak terlalu terik. Jika bambu mendapat cahaya yang terlalu terik, maka akan mengakibatkan daun bambu menjadi gelap dan berluka. Selain itu, bambu juga memerlukan air yang cukup namun tidak terlalu banyak. Jika bambu mendapat air yang terlalu banyak, maka akan mengakibatkan akar bambu menjadi lembab dan mudah terinfeksi bakteri. Untuk merawat bambu dengan baik, sebaiknya memberikan air yang cukup namun tidak terlalu banyak, dan memberikan cahaya yang cukup namun tidak terlalu terik. Selain itu, juga penting untuk memberikan pupuk yang cukup agar bambu tumbuh dengan baik. Pupuk yang cocok untuk bambu adalah pupuk organik atau pupuk kandang yang dicampur dengan pasir laut. Selain itu, juga penting untuk membersihkan bambu secara berkala agar tidak terinfeksi oleh kutu atau serangga.
```
```

```
title: 'Tips Merawat Jamani Dolar untuk Pemula',
excerpt: 'Panduan lengkap merawat ZZ Plant yang sangat mudah untuk pemula. Tanaman yang tahan banting dan cocok untuk indoor.',
content: '',
author: 'Tim Azka Garden',
publishedAt: '2024-01-15',
category: 'Perawatan',
image: 'https://images.pexels.com/photos/6208086/pexels-photo-6208086.jpeg',
readTime: '5 menit'
},
{
id: '2',
title: 'Cara Membuat Media Tanam yang Tepat',
excerpt: 'Resep media tanam terbaik untuk berbagai jenis tanaman hias. Tips dari pengalaman bertahun-tahun Azka Garden.',
content: '',
author: 'Pak Hendrik',
publishedAt: '2024-01-12',
category: 'Tutorial',
image: 'https://images.pexels.com/photos/1301856/pexels-photo-1301856.jpeg',
readTime: '8 menit'
},
{
id: '3',
title: 'Monstera Deliciosa: Si Daun Berlubang yang Menawan',
excerpt: 'Mengapa Monstera menjadi tanaman hias paling populer? Simak tips perawatan dan propagasi yang mudah.',
content: ''
```

```
author: 'Tim Azka Garden',
publishedAt: '2024-01-10',
category: 'Tanaman Populer',
image: 'https://images.pexels.com/photos/1005058/pexels-photo-1005058.jpeg',
readTime: '6 menit'
},
{
id: '4',
title: 'Bonsai Gestrum: Seni Tanaman Miniatur Berbunga',
excerpt: 'Panduan lengkap merawat bonsai gestrum dengan bunga kuning harum. Teknik pembentukan dan perawatan intensif.',
content: '',
author: 'Master Bonsai Azka',
publishedAt: '2024-01-08',
category: 'Bonsai',
image: 'https://images.pexels.com/photos/1301856/pexels-photo-1301856.jpeg',
readTime: '12 menit'
},
{
id: '5',
title: 'Tanaman Indoor Terbaik untuk Apartemen',
excerpt: 'Rekomendasi tanaman indoor yang cocok untuk ruang terbatas. Mudah perawatan dan mempercantik ruangan.',
content: '',
author: 'Tim Azka Garden',
publishedAt: '2024-01-05',
category: 'Indoor Plants',
```

```
image: 'https://images.pexels.com/photos/2123482/pexels-photo-
```

```

 2123482.jpeg',
 readTime: '7 menit'
},
{
 id: '6',

```

## 9. src/pages/Login.tsx (Masuk)

```

import React, { useState } from 'react';
import { Link, useNavigate, useLocation } from 'react-router-dom';
import { Eye, EyeOff, Mail, Lock, Leaf } from 'lucide-react';
import { useAuth } from '../contexts/AuthContext';

const Login: React.FC = () => {
 const [formData, setFormData] = useState({
 email: '',
 password: '',
 rememberMe: false
});
 const [showPassword, setShowPassword] = useState(false);
 const [isSubmitting, setIsSubmitting] = useState(false);

 const { login, error, clearError } = useAuth();
 const navigate = useNavigate();
 const location = useLocation();

 const from = (location.state as any)?.from?.pathname || '/';
 const handleInputChange: React.ChangeEvent<HTMLInputElement> => {
 const { name, value, type, checked } = e.target;

```

```

setFormData(prev => ({
 ...prev,
 [name]: type === 'checkbox' ? checked : value
}));

if (error) {
 clearError();
}
};

const handleSubmit = async (e: React.FormEvent) => {
 e.preventDefault();
 setIsSubmitting(true);

 try {
 await login(formData.email, formData.password);
 navigate(from, { replace: true });
 } catch (error) {
 // Error is handled by the context
 } finally {
 setIsSubmitting(false);
 }
};

return (
 <div className="min-h-screen bg-gradient-to-br from-green-50 to-green-100 dark:from-gray-900 dark:to-gray-800 flex items-center justify-center py-12 px-4 sm:px-6 lg:px-8">
 <div className="max-w-md w-full space-y-8">
 <div className="text-center">

```

```

<Link to="/" className="inline-flex items-center space-x-2 mb-6">
 <Leaf className="h-10 w-10 text-green-600" />
 Azka Garden
</Link>
<h2 className="text-3xl font-bold text-gray-900 dark:text-white mb-2">
 Masuk ke Akun Anda
</h2>
<p className="text-gray-600 dark:text-gray-400">
 Belum punya akun?{' '}
<Link to="/register" className="text-green-600 dark:text-green-400 hover:text-green-700 dark:hover:text-green-300 font-medium">
 Daftar sekarang
</Link>
</p>
</div>

<div className="bg-white dark:bg-gray-800 rounded-lg shadow-lg p-8">
 <form onSubmit={handleSubmit} className="space-y-6">
 {error && (
 <div className="bg-red-50 dark:bg-red-900 border border-red-200 dark:border-red-700 rounded-md p-4">
 <p className="text-red-600 dark:text-red-400 text-sm">{error}</p>
 </div>
)}
 </div>

```

```

<label htmlFor="email" className="block text-sm font-medium
text-gray-700 dark:text-gray-300 mb-2">
 Email
</label>
<div className="relative">
 <Mail className="absolute left-3 top-1/2 transform -translate-y-
1/2 h-5 w-5 text-gray-400" />
 <input
 id="email"
 name="email"
 type="email"

```

#### 10. src/pages/Register.tsx (Registrasi)

```

import React, { useState } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import { Eye, EyeOff, Mail, Lock, User, Phone, Leaf, Check } from 'lucide-
react';
import { useAuth } from '../contexts/AuthContext';

const Register: React.FC = () => {
 const [formData, setFormData] = useState({
 email: '',
 username: '',
 fullName: '',
 phoneNumber: '',
 password: '',
 confirmPassword: '',
 acceptTerms: false,
 subscribeNewsletter: false
 });

```

```

const [showPassword, setShowPassword] = useState(false);
const [showConfirmPassword, setShowConfirmPassword] =
useState(false);
const [isSubmitting, setIsSubmitting] = useState(false);

const { register, error, clearError } = useAuth();
const navigate = useNavigate();

const handleInputChange = (e: React.ChangeEvent<HTMLInputElement>) => {
 const { name, value, type, checked } = e.target;
 setFormData(prev => ({
 ...prev,
 [name]: type === 'checkbox' ? checked : (name === 'email' ?
value.trim() : value)
 }));
}

if (error) {
 clearError();
}
};

const handleSubmit = async (e: React.FormEvent) => {
 e.preventDefault();
 setIsSubmitting(true);

 try {
 await register(formData);
 // After successful registration, user will be automatically logged in
 // Navigate to products page to start shopping
 }
}

```

```

 navigate('/stripe-products', { replace: true });
 } catch (error) {
 // Error is handled by the context
 } finally {
 setIsSubmitting(false);
 }
};

const passwordsMatch = formData.password ===
formData.confirmPassword;

const isFormValid = formData.email && formData.username &&
formData.fullName &&
 formData.password && formData.confirmPassword &&
 passwordsMatch && formData.acceptTerms;

return (
 <div className="min-h-screen bg-gradient-to-br from-green-50 to-
green-100 flex items-center justify-center py-12 px-4 sm:px-6 lg:px-8">
 <div className="max-w-md w-full space-y-8">
 <div className="text-center">
 <Link to="/" className="inline-flex items-center space-x-2 mb-6">
 <Leaf className="h-10 w-10 text-green-600" />
 Azka
Garden
 </Link>
 <h2 className="text-3xl font-bold text-gray-900 mb-2">
 Buat Akun Baru
 </h2>
 <p className="text-gray-600">
 Sudah punya akun?{' '}
 </p>
 </div>
 </div>
</div>

```

```

<Link to="/login" className="text-green-600 hover:text-green-700 font-medium">
 Masuk di sini
</Link>
</p>
</div>

<div className="bg-white rounded-lg shadow-lg p-8">
 <form onSubmit={handleSubmit} className="space-y-6">
 {error && (
 <div className="bg-red-50 border border-red-200 rounded-md p-4">
 <p className="text-red-600 text-sm">{error}</p>
 </div>
)}
 <div className="grid grid-cols-1 gap-6">
 <div>
 <label htmlFor="fullName" className="block text-sm font-medium text-gray-700 mb-2">
 Nama Lengkap *
 </label>
 <div className="relative">

```

## 11. src/pages/Profile.tsx (Profil)

```

import React, { useEffect, useState } from 'react';
import { User, Mail, Phone, Save, Shield, Bell } from 'lucide-react';
import { useAuth } from '../contexts/AuthContext';

const Profile: React.FC = () => {

```

```
const { user, updateProfile } = useAuth() as any;
const [form, setForm] = useState({
 fullName: '',
 username: '',
 phoneNumber: ''
});
const [prefs, setPrefs] = useState({
 newsletter: true,
 promo: true,
 notifications: true
});
const [saving, setSaving] = useState(false);
const [done, setDone] = useState(false);

useEffect(() => {
 setForm({
 fullName: user?.fullName || '',
 username: user?.username || '',
 phoneNumber: user?.phoneNumber || ''
 });
 const savedPrefs = JSON.parse(localStorage.getItem('profile-prefs') || 'null');
 if (savedPrefs) setPrefs(savedPrefs);
}, [user]);

const onSave = async (e: React.FormEvent) => {
 e.preventDefault();
 setSaving(true);
 setDone(false);
 try {
```

```

if (typeof updateProfile === 'function') {
 await updateProfile(form);
} else {
 // fallback simpan lokal
 localStorage.setItem('profile', JSON.stringify(form));
}
localStorage.setItem('profile-prefs', JSON.stringify(prefs));
setDone(true);
} catch (e) {
 console.error(e);
 alert('Gagal menyimpan profil');
} finally {
 setSaving(false);
}
};

return (
<div className="min-h-screen bg-gray-50 dark:bg-gray-900 py-10">
 <div className="max-w-3xl mx-auto px-4">
 <h1 className="text-3xl font-bold text-gray-900 dark:text-white mb-6">Profil</h1>

 <form onSubmit={onSave} className="space-y-6">
 <div className="bg-white dark:bg-gray-800 border border-gray-200 dark:border-gray-700 rounded-xl p-6">
 <h3 className="font-semibold text-gray-900 dark:text-white mb-4">Informasi Akun</h3>
 <div className="grid grid-cols-1 sm:grid-cols-2 gap-4">
 <div>
 <label className="block text-sm mb-2 text-gray-700 dark:text-"

```

```
gray-300">Nama Lengkap</label>
 <div className="relative">
 <User className="absolute left-3 top-3 h-5 w-5 text-gray-400"
/>
 <input
 className="w-full pl-10 pr-3 py-2 rounded-lg bg-white
dark:bg-gray-900 border border-gray-300 dark:border-gray-700 text-gray-900
dark:text-white"
 value={form.fullName}
 onChange={(e) => setForm({ ...form, fullName: e.target.value
})}
 />
 </div>
</div>
<div>
 <label className="block text-sm mb-2 text-gray-700 dark:text-
gray-300">Username</label>
 <input
 className="w-full px-3 py-2 rounded-lg bg-white dark:bg-
gray-900 border border-gray-300 dark:border-gray-700 text-gray-900 dark:text-
white"
 value={form.username}
 onChange={(e) => setForm({ ...form, username: e.target.value
})}
 />
</div>
<div>
 <label className="block text-sm mb-2 text-gray-700 dark:text-
gray-300">Email</label>
 <div className="relative">
```

```

<Mail className="absolute left-3 top-3 h-5 w-5 text-gray-400" />
 <input
 disabled
 value={user?.email || ""}
 className="w-full pl-10 pr-3 py-2 rounded-lg bg-gray-100
dark:bg-gray-900/60 border border-gray-300 dark:border-gray-700 text-gray-500
dark:text-gray-400"
 />
 </div>
 </div>
 <div>
 <label className="block text-sm mb-2 text-gray-700 dark:text-
gray-300">No. Telepon</label>
 <div className="relative">
 <Phone className="absolute left-3 top-3 h-5 w-5 text-gray-
400" />
 <input
 className="w-full pl-10 pr-3 py-2 rounded-lg bg-white
dark:bg-gray-900 border border-gray-300 dark:border-gray-700 text-gray-900
dark:text-white"
 value={form.phoneNumber}
 onChange={(e) => setForm({ ...form, phoneNumber:
e.target.value })}
 />
 </div>
 </div>
 </div>
 </div>

<div className="bg-white dark:bg-gray-800 border border-gray-

```

```

200 dark:border-gray-700 rounded-xl p-6">
 <h3 className="font-semibold text-gray-900 dark:text-white mb-4">Preferensi</h3>
 <div className="space-y-3">
 <label className="flex items-center gap-3">
 <input
 type="checkbox"
 checked={prefs.newsletter}
 onChange={(e) => setPrefs({ ...prefs, newsletter: e.target.checked })}>
 />
 Berlangganan newsletter
 </label>
 <label className="flex items-center gap-3">
 <input
 type="checkbox"
 checked={prefs.promo}
 onChange={(e) => setPrefs({ ...prefs, promo: e.target.checked })}>
 />
 Terima promo & penawaran
 </label>
 <label className="flex items-center gap-3">
 <input
 type="checkbox"
 checked={prefs.notifications}
 onChange={(e) => setPrefs({ ...prefs, notifications: e.target.checked })}>
 />
 </label>
 </div>

```

```
>

 Notifikasi
status pesanan
 </label>
 <div className="mt-2 text-xs text-gray-500 dark:text-gray-400
flex items-center gap-2">
 <Shield className="h-4 w-4" /> Data Anda aman dan
terenkripsi.
 </div>
 </div>
 </div>

 <div className="flex gap-3">
 <button
 type="submit"
 disabled={saving}
 className="inline-flex items-center gap-2 px-5 py-3 rounded-lg
bg-green-600 hover:bg-green-700 text-white font-semibold disabled:opacity-70"
>
 <Save className="h-5 w-5" />
 {saving ? 'Menyimpan...' : 'Simpan Perubahan'}
 </button>
 {done && <div className="text-green-700 dark:text-green-300
flex items-center gap-2"><Bell className="h-5 w-5" /> Tersimpan</div>}
 </div>
 </form>
 </div>
 </div>
);
};
```

```
export default Profile;
```

## 12. src/pages/Orders.tsx (Pesanan)

```
import React from 'react';
import { Package, Truck, CheckCircle, Clock, ChevronRight } from 'lucide-react';
import { useOrder } from '../contexts/OrderContext';

const statusMap: Record<string, { icon: React.ReactNode; label: string; color: string }> = {
 pending: { icon: <Clock className="h-5 w-5" />, label: 'Menunggu', color: 'bg-yellow-100 text-yellow-800' },
 processing: { icon: <Package className="h-5 w-5" />, label: 'Diproses', color: 'bg-blue-100 text-blue-800' },
 shipped: { icon: <Truck className="h-5 w-5" />, label: 'Dikirim', color: 'bg-indigo-100 text-indigo-800' },
 delivered: { icon: <CheckCircle className="h-5 w-5" />, label: 'Terkirim', color: 'bg-green-100 text-green-800' },
};

const Orders: React.FC = () => {
 const { orders } = useOrder() as any;

 return (
 <div className="min-h-screen bg-gray-50 dark:bg-gray-900 py-10">
 <div className="max-w-5xl mx-auto px-4">
 <h1 className="text-3xl font-bold text-gray-900 dark:text-white mb-6">Pesanan Saya</h1>

 {!orders || orders.length === 0 ? (

```

```

<div className="rounded-xl border border-gray-200 dark:border-gray-700 bg-white dark:bg-gray-800 p-6 text-gray-600 dark:text-gray-300">
 Belum ada pesanan. Ayo belanja tanaman favorit Anda!
</div>
):(
<div className="space-y-4">
{orders.map((o: any) => {
 const status = statusMap[o.status] || statusMap.pending;
 return (
 <div
 key={o.id}
 className="bg-white dark:bg-gray-800 border border-gray-200 dark:border-gray-700 rounded-xl p-5"
 >
 <div className="flex items-center justify-between">
 <div>
 <div className="font-semibold text-gray-900 dark:text-white">Order #{o.id.slice(-6)}</div>
 <div className="text-sm text-gray-500 dark:text-gray-400">
 {new Date(o.createdAt).toLocaleString('id-ID')}
 </div>
 </div>
 <div className={`inline-flex items-center gap-2 px-3 py-1 rounded-full text-sm ${status.color}`}>
 {status.icon} {status.label}
 </div>
 </div>
 <div className="mt-4 overflow-auto">
 <table className="w-full text-sm">

```

```

<thead>
 <tr className="text-left text-gray-500 dark:text-gray-
400">
 <th className="py-2">Produk</th>
 <th className="py-2">Harga</th>
 <th className="py-2">Qty</th>
 <th className="py-2">Subtotal</th>
 </tr>
</thead>
<tbody>
 {o.items.map((it: any) => (
 <tr key={it.id} className="border-t border-gray-100
dark:border-gray-700">
 <td className="py-2 text-gray-900 dark:text-
white">{it.name}</td>
 <td className="py-2">Rp {it.price?.toLocaleString('id-
ID')}</td>
 <td className="py-2">{it.quantity}</td>
 <td className="py-2">Rp {(it.price * *
it.quantity)?.toLocaleString('id-ID')}</td>
 </tr>
)))
</tbody>
</table>
</div>
<div className="mt-4 flex items-center justify-between">
 <div className="text-sm text-gray-500 dark:text-gray-
400">Metode: {o.paymentMethod || 'Stripe'}</div>
 <div className="font-bold text-green-700 dark:text-green-
300">

```

```

 Total: Rp {o.total?.toLocaleString('id-ID')}
 </div>
 </div>
 <div className="mt-4">
 <button className="inline-flex items-center gap-2 px-4 py-2 rounded-lg border border-gray-300 dark:border-gray-700 text-gray-700 dark:text-gray-300 hover:bg-gray-50 dark:hover:bg-gray-700">
 Lihat Detail <ChevronRight className="h-4 w-4" />
 </button>
 </div>
 </div>
);
)}
</div>
</div>
);
};

export default Orders;

```

### 13. src/pages/Cart.tsx (Keranjang)

```

import React, { useMemo } from 'react';
import { Trash2, Plus, Minus, ShoppingBag, CreditCard } from 'lucide-react';
import { useCart } from '../contexts/CartContext';
import { Link, useNavigate } from 'react-router-dom';

const Cart: React.FC = () => {

```

```

const navigate = useNavigate();

const { items = [], removeFromCart, updateQuantity, clearCart } =
useCart() as any;

const totals = useMemo(() => {
 const subtotal = items.reduce((s: number, it: any) => s + it.price *
it.quantity, 0);
 const shipping = items.length > 0 ? 15000 : 0;
 const total = subtotal + shipping;
 return { subtotal, shipping, total };
}, [items]);

const checkout = () => {
 // Arahkan ke koleksi premium/produk stripe atau halaman pembayaran
 yang ada
 navigate('/stripe-products');
};

if (!items || items.length === 0) {
 return (
 <div className="min-h-screen bg-gray-50 dark:bg-gray-900 py-10">
 <div className="max-w-3xl mx-auto px-4 text-center">
 <ShoppingBag className="h-14 w-14 mx-auto text-gray-400 mb-4" />
 <h1 className="text-2xl font-bold text-gray-900 dark:text-white mb-2">Keranjang Kosong</h1>
 <p className="text-gray-600 dark:text-gray-300 mb-6">Mulai belanja tanaman favorit Anda.</p>
 <Link
 to="/products"

```

```

 className="inline-flex items-center px-5 py-3 rounded-lg bg-
green-600 hover:bg-green-700 text-white font-semibold"
 >
 Jelajahi Produk
 </Link>
</div>
</div>
);
}

return (
<div className="min-h-screen bg-gray-50 dark:bg-gray-900 py-10">
 <div className="max-w-6xl mx-auto px-4 grid grid-cols-1 lg:grid-
cols-3 gap-6">
 <div className="lg:col-span-2 space-y-4">
 {items.map((it: any) => (
 <div key={it.id} className="bg-white dark:bg-gray-800 border
border-gray-200 dark:border-gray-700 rounded-xl p-4 flex gap-4">
 <img src={it.image} alt={it.name} className="w-24 h-24 object-
cover rounded-lg" />
 <div className="flex-1">
 <div className="font-semibold text-gray-900 dark:text-
white">{it.name}</div>
 <div className="text-green-700 dark:text-green-300 font-
bold">Rp {it.price?.toLocaleString('id-ID')}</div>
 <div className="mt-3 flex items-center gap-3">
 <button
 onClick={() => updateQuantity?(it.id, Math.max(1, it.quantity
- 1))}>
 className="p-2 rounded-lg border border-gray-300

```

```

dark:border-gray-700"
>
<Minus className="h-4 w-4" />
</button>
<div className="px-3 py-1 rounded-lg bg-gray-100 dark:bg-
gray-700 text-gray-900 dark:text-white">
 {it.quantity}
</div>
<button
 onClick={() => updateQuantity?.(it.id, it.quantity + 1)}
 className="p-2 rounded-lg border border-gray-300
dark:border-gray-700"
>
<Plus className="h-4 w-4" />
</button>
<button
 onClick={() => removeFromCart?.(it.id)}
 className="ml-auto inline-flex items-center gap-2 px-3 py-2
rounded-lg border border-red-300 text-red-600 hover:bg-red-50 dark:hover:bg-red-
900/20"
>
<Trash2 className="h-4 w-4" /> Hapus
</button>
</div>
</div>
<div className="hidden sm:flex items-start font-semibold text-
gray-900 dark:text-white">
 Rp {(it.price * it.quantity)?.toLocaleString('id-ID')}
</div>
</div>

```

```

))}

```

```

<div className="flex justify-between">
 <button
 onClick={() => clearCart?.()}
 className="text-sm text-gray-600 dark:text-gray-300
 hover:underline">
 >
 Bersihkan Keranjang
 </button>
 <Link to="/products" className="text-sm text-green-600
 dark:text-green-400 hover:underline">
 Tambah Produk
 </Link>
</div>
</div>

<div className="lg:col-span-1">
 <div className="bg-white dark:bg-gray-800 border border-gray-
 200 dark:border-gray-700 rounded-xl p-6">
 <h3 className="font-bold text-gray-900 dark:text-white mb-
 4">Ringkasan</h3>
 <div className="space-y-2 text-sm">
 <div className="flex justify-between">
 <span className="text-gray-600 dark:text-gray-
 300">Subtotal
 Rp
 {totals.subtotal.toLocaleString('id-ID')}
 </div>
 <div className="flex justify-between">
 <span className="text-gray-600 dark:text-gray-

```

```

300">Pengiriman
 Rp
{totals.shipping.toLocaleString('id-ID')}
 </div>
 <div className="border-t border-gray-200 dark:border-gray-700
pt-2 flex justify-between">
 <span className="font-bold text-gray-900 dark:text-
white">Total
 <span className="font-bold text-green-700 dark:text-green-
300">
 Rp {totals.total.toLocaleString('id-ID')}

 </div>
 </div>
 <button
 onClick={checkout}
 className="mt-5 w-full inline-flex items-center justify-center
gap-2 px-5 py-3 rounded-lg bg-green-600 hover:bg-green-700 text-white font-
semibold"
 >
 <CreditCard className="h-5 w-5" /> Checkout
 </button>
 <div className="mt-2 text-xs text-gray-500 dark:text-gray-400">
 Pembayaran diproses dengan aman melalui Stripe.
 </div>
 </div>
 </div>
 </div>
);

```

```

};

export default Cart;

```

#### 14. src/pages/Wishlist.tsx (Daftar Keinginan)

```

import React, { useState, useEffect } from 'react';
import { Heart, ShoppingCart, Trash2, Search } from 'lucide-react';
import { Link } from 'react-router-dom';
import { Plant } from '../types';
import { getPlants } from '../services/database';
import { useCart } from '../contexts/CartContext';

const Wishlist: React.FC = () => {
 const [wishlistItems, setWishlistItems] = useState<Plant[]>([]);
 const [loading, setLoading] = useState(true);
 const { addToCart } = useCart();

 useEffect(() => {
 loadWishlist();
 }, []);

 const loadWishlist = async () => {
 try {
 // Simulate wishlist with some featured plants
 const plants = await getPlants();
 setWishlistItems(plants.slice(0, 3)); // Mock wishlist with first 3 plants
 } catch (error) {
 console.error('Error loading wishlist:', error);
 } finally {
 setLoading(false);
 }
 };
}

```

```

 }

};

const removeFromWishlist = (plantId: string) => {
 setWishlistItems(prev => prev.filter(item => item.id !== plantId));
};

const handleAddToCart = (plant: Plant) => {
 addToCart({
 id: plant.id,
 name: plant.name,
 price: plant.price,
 image: plant.image,
 quantity: 1
 });
};

if (loading) {
 return (
 <div className="min-h-screen bg-gray-50 dark:bg-gray-900 py-8">
 <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
 <div className="animate-pulse">
 <div className="bg-gray-300 h-8 w-64 mb-8 rounded"></div>
 <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">
 {[...Array(3)].map((_, i) => (
 <div key={i} className="bg-white dark:bg-gray-800 rounded-lg shadow-md p-4">
 <div className="bg-gray-300 h-48 rounded-lg mb-4"></div>
 <div className="bg-gray-300 h-4 rounded mb-2"></div>

```

```

 <div className="bg-gray-300 h-4 rounded w-3/4"></div>
 </div>
)})
</div>
</div>
</div>
</div>
);
}

if (wishlistItems.length === 0) {
 return (
 <div className="min-h-screen bg-gray-50 dark:bg-gray-900 py-8">
 <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
 <div className="text-center py-12">
 <Heart className="h-16 w-16 text-gray-400 mx-auto mb-4" />
 <h2 className="text-2xl font-bold text-gray-900 dark:text-white mb-2">Wishlist Kosong</h2>
 <p className="text-gray-600 dark:text-gray-300 mb-8">
 Belum ada tanaman favorit di wishlist Anda. Mulai tambahkan sekarang!
 </p>
 <Link
 to="/products"
 className="inline-flex items-center px-6 py-3 bg-green-600 text-white font-semibold rounded-lg hover:bg-green-700 transition-colors">
 >
 Jelajahi Tanaman
 </Link>
 </div>
 </div>
 </div>
)
}

```

## 15. src/pages/CustomerService.tsx (Layanan Pelanggan)

```

import React, { useState, useEffect } from 'react';
import { MessageCircle, Send, User, Shield, Code, Clock, CheckCircle, AlertCircle } from 'lucide-react';
import { useAuth } from '../contexts/AuthContext';

interface ChatMessage {
 id: string;
 senderId: string;
 senderName: string;
 senderRole: 'customer' | 'admin' | 'developer';
 message: string;
 timestamp: Date;
 isRead: boolean;
 replyTo?: string;
}

interface ChatSession {
 id: string;
 customerId: string;
 customerName: string;
 status: 'active' | 'resolved' | 'escalated';
 messages: ChatMessage[];
 assignedTo?: string;
 createdAt: Date;
 updatedAt: Date;
}

const CustomerService: React.FC = () => {
 const [sessions, setSessions] = useState<ChatSession[]>([]);
 const [selectedSession, setSelectedSession] = useState<string | null>(null);
}

```

```

const [replyMessage, setReplyMessage] = useState("");
const [filter, setFilter] = useState<'all' | 'active' | 'resolved'>('all');
const { user } = useAuth();

useEffect(() => {
 loadChatSessions();

 // Auto-refresh every 3 seconds for real-time updates
 const interval = setInterval(loadChatSessions, 3000);
 return () => clearInterval(interval);
}, []);

const loadChatSessions = () => {
 try {
 const savedSessions = localStorage.getItem('chat-sessions');
 if (savedSessions) {
 const parsedSessions = JSON.parse(savedSessions).map((session: any) => ({
 ...session,
 createdAt: new Date(session.createdAt),
 updatedAt: new Date(session.updatedAt),
 messages: session.messages.map((msg: any) => ({
 ...msg,
 timestamp: new Date(msg.timestamp)
 }))
 }));
 setSessions(parsedSessions);
 }
 } catch (error) {
 console.error('Error loading chat sessions:', error);
 }
}

```

```
 }

};

const handleSendReply = async () => {
 if (!replyMessage.trim() || !selectedSession || !user) return;

 const session = sessions.find(s => s.id === selectedSession);
 if (!session) return;

 const newMessage: ChatMessage = {
 id: 'msg-' + Date.now(),
 senderId: user.id,
 senderName: user.fullName || user.email,
 senderRole: user.role.toLowerCase() as 'admin' | 'developer',
 message: replyMessage,
 timestamp: new Date(),
 isRead: false
 };

 const updatedSession = {
 ...session,
 messages: [...session.messages, newMessage],
 updatedAt: new Date()
 };

 const updatedSessions = sessions.map(s =>
 s.id === selectedSession ? updatedSession : s
);

 setSessions(updatedSessions);
```

```

localStorage.setItem('chat-sessions', JSON.stringify(updatedSessions));
setReplyMessage("");
};

const handleResolveSession = (sessionId: string) => {
 const updatedSessions = sessions.map(session =>
 session.id === sessionId
 ? { ...session, status: 'resolved' as const, updatedAt: new Date() }
 : session
);
 setSessions(updatedSessions);
 localStorage.setItem('chat-sessions', JSON.stringify(updatedSessions));
};

const filteredSessions = sessions.filter(session => {
 if (filter === 'all') return true;
 return session.status === filter;
});

const selectedSessionData = sessions.find(s => s.id === selectedSession);

```

## 16. src/pages/Reviews.tsx

```

import React from 'react';
import ReviewsAndComments from './components/reviews/ReviewsAndComments';

const ReviewsPage: React.FC = () => {
 return (
 <div className="min-h-screen bg-gray-50 dark:bg-gray-900 py-10">
 <div className="max-w-5xl mx-auto px-4 sm:px-6 lg:px-8">

```

```

<div className="mb-6">
 <h1 className="text-3xl font-bold text-gray-900 dark:text-white">Ulasan
 & Komentar</h1>
 <p className="text-gray-600 dark:text-gray-300">Bagikan pengalaman
 Anda berbelanja di Azka Garden.</p>
</div>
<div className="bg-white dark:bg-gray-800 border border-gray-200
dark:border-gray-700 rounded-xl p-4 sm:p-6">
 <ReviewsAndComments />
</div>
</div>
</div>
);
};

export default ReviewsPage;

```

## 17. src/components/reviews/ReviewsAndComments.tsx (Ulasan dan Komentar)

```

import React, { useState, useEffect } from 'react';
import { Star, Heart, MessageCircle, Send, ThumbsUp, Reply, Trash2,
Globe } from 'lucide-react';
import { useAuth } from '../contexts/AuthContext';

interface Review {
 id: string;
 userId: string;

 userName: string;
 userRole: 'customer' | 'admin' | 'developer';
}

```

```

rating: number;
comment: string;
likes: string[];
replies: ReviewReply[];
createdAt: Date;
updatedAt: Date;
}

interface ReviewReply {
id: string;
userId: string;
userName: string;
userRole: 'customer' | 'admin' | 'developer';
comment: string;
likes: string[];
createdAt: Date;
}

const ReviewsAndComments: React.FC = () => {
 const [reviews, setReviews] = useState<Review[]>([]);
 const [newReview, setNewReview] = useState({ rating: 5, comment: "" });
 const [replyTo, setReplyTo] = useState<string | null>(null);
 const [replyComment, setReplyComment] = useState("");
 const { user } = useAuth();

 useEffect(() => {
 loadReviews();

 // Real-time updates - check for new reviews every 3 seconds
 const interval = setInterval(loadReviews, 3000);
 }, []);
}

const loadReviews = async () => {
 const response = await fetch("https://api.example.com/reviews");
 const data = await response.json();
 setReviews(data);
}

const handleNewReview = (newReview: Review) => {
 setNewReview((prev) => ({ ...prev, comment: newReview.comment }));
}

const handleReply = (reviewId: string, reply: string) => {
 const updatedReviews = reviews.map((r) => {
 if (r.id === reviewId) {
 return { ...r, replies: [...r.replies, { id: replyTo, comment: reply }] };
 }
 return r;
 });
 setReviews(updatedReviews);
}

const handleLike = (reviewId: string) => {
 const updatedReviews = reviews.map((r) => {
 if (r.id === reviewId) {
 return { ...r, likes: [...r.likes, "user"] };
 }
 return r;
 });
 setReviews(updatedReviews);
}

```

```
return () => clearInterval(interval);

}, []);
```

```
const loadReviews = () => {
 const savedReviews = localStorage.getItem('global-reviews');
 if (savedReviews) {
 try {
 const parsedReviews = JSON.parse(savedReviews).map((review: any)
=> ({
 ...review,
 createdAt: new Date(review.createdAt),
 updatedAt: new Date(review.updatedAt),
 replies: review.replies.map((reply: any) => ({
 ...reply,
 createdAt: new Date(reply.createdAt)
 }))
 }));
 });
 setReviews(parsedReviews);
 } catch (error) {
 console.error('Error loading reviews:', error);
 }
} else {
 // Initialize with demo reviews
 const demoReviews: Review[] = [
 {
 id: 'review-1',
 userId: 'customer-001',
 userName: 'Sari Dewi',
 userRole: 'customer',
```

```
rating: 5,
comment: 'Pelayanan Azka Garden sangat memuaskan! Tanaman
yang saya beli sehat dan packaging rapi. Terima kasih!',
likes: [],
replies: [
{
id: 'reply-1',
userId: 'admin-1',
userName: 'Admin Azka Garden',
userRole: 'admin',
comment: 'Terima kasih atas review positifnya! Kami senang
tanaman Anda tumbuh dengan baik. Jangan lupa ikuti tips perawatan di channel
YouTube kami.',
likes: [],
createdAt: new Date(Date.now() - 43200000)
}
,
createdAt: new Date(Date.now() - 86400000),
updatedAt: new Date(Date.now() - 86400000)
},
{
id: 'review-2',
userId: 'customer-002',
userName: 'Budi Santoso',
userRole: 'customer',
rating: 4,
comment: 'Koleksi tanaman lengkap dan harga terjangkau.
Recommended untuk pecinta tanaman hias!',
likes: [],
replies: [
```

```
{
 id: 'reply-2',
 userId: 'admin-1',
 userName: 'Admin Azka Garden',
 userRole: 'admin',
```

#### 18. src/pages/admin/AdminLogin.tsx

```
import React, { useState } from 'react';
import { Shield, Mail, Lock, LogIn } from 'lucide-react';
import { useAuth } from '../contexts/AuthContext';
import { useNavigate, Link } from 'react-router-dom';

const AdminLogin: React.FC = () => {
 const { login } = useAuth() as any;
 const navigate = useNavigate();
 const [form, setForm] = useState({ email: "", password: "" });
 const [error, setError] = useState<string>("");
 const [submitting, setSubmitting] = useState(false);

 const onSubmit = async (e: React.FormEvent) => {
 e.preventDefault();
 setSubmitting(true);
 setError("");
 try {
 const user = await login(form.email, form.password);
 if (user?.role !== 'ADMIN') {
 setError('Akun ini tidak memiliki akses administrator.');
 return;
 }
 navigate('/admin/dashboard', { replace: true });
 } catch (err) {
 setError("Terjadi kesalahan saat login.");
 }
 };
}
```

```

} catch (e: any) {
 setError(e?.message || 'Gagal login');
} finally {
 setSubmitting(false);
}
};

return (
 <div className="min-h-screen bg-gray-50 dark:bg-gray-900 flex items-center justify-center py-10 px-4">
 <div className="w-full max-w-md bg-white dark:border-gray-800 border border-gray-200 dark:border-gray-700 rounded-xl p-6">
 <div className="flex items-center gap-2 mb-4">
 <Shield className="h-8 w-8 text-green-600" />
 <h1 className="text-2xl font-bold text-gray-900 dark:text-white">Admin Portal</h1>
 </div>
 {error && <div className="mb-3 rounded-lg bg-red-50 dark:bg-red-900/30 border border-red-200 dark:border-red-800 p-3 text-red-700 dark:text-red-300">{error}</div>}
 <form onSubmit={onSubmit} className="space-y-4">
 <div className="relative">
 <Mail className="absolute left-3 top-3 h-5 w-5 text-gray-400" />
 <input
 type="email"
 required
 placeholder="Email admin"
 className="w-full pl-10 pr-3 py-2 rounded-lg bg-white dark:bg-gray-900 border border-gray-300 dark:border-gray-700 text-gray-900 dark:text-white"
 >
 </div>
 </form>
 </div>
 </div>
)

```

```

 value={form.email}
 onChange={(e) => setForm({ ...form, email: e.target.value.trim() })
)}
 />
</div>
<div className="relative">
 <Lock className="absolute left-3 top-3 h-5 w-5 text-gray-400" />
 <input
 type="password"
 required
 placeholder="Password"
 className="w-full pl-10 pr-3 py-2 rounded-lg bg-white dark:bg-gray-900 border border-gray-300 dark:border-gray-700 text-gray-900 dark:text-white"
 value={form.password}
 onChange={(e) => setForm({ ...form, password: e.target.value })}>
 />
</div>
<button
 type="submit"
 disabled={submitting}
 className="w-full inline-flex items-center justify-center gap-2 px-4 py-2 rounded-lg bg-green-600 hover:bg-green-700 text-white font-semibold
disabled:opacity-70">
 >
 <LogIn className="h-5 w-5" /> {submitting ? 'Masuk...' : 'Masuk'}
</button>
<div className="text-center text-sm text-gray-500 dark:text-gray-400">
 <Link to="/" className="hover:underline">Kembali ke

```

```
Beranda</Link>
 </div>
 </form>
 </div>
 </div>
);
};

export default AdminLogin;
```

#### 19. src/pages/admin/AdminDashboard.tsx

```
import React, { useEffect, useState } from 'react';
import { useNavigate } from 'react-router-dom';
import {
 BarChart3,
 Users,
 ShoppingCart,
 DollarSign,
 Package,
 TrendingUp,
 AlertTriangle,
 Settings,
 LogOut,
 Bell,
 RefreshCw,
 Shield,
 Eye,
 Plus,
 Edit,
 User,
```

```
CheckCircle,
FileText,
CreditCard,
Truck,
Calendar,
Search,
Filter,
Trash2,
Save,
X,
MessageSquare,
Reply,
Send,
Star
} from 'lucide-react';
import { useAuth } from '../contexts/AuthContext';
import { useOrder } from '../contexts/OrderContext';
import { getPlantStatistics } from '../services/database';

const AdminDashboard: React.FC = () => {
 const [activeTab, setActiveTab] = useState('dashboard');
 const [dashboardStats, setDashboardStats] = useState<any>(null);
 const [plantStats, setPlantStats] = useState<any>(null);
 const [loading, setLoading] = useState(true);
 const [allUsers, setAllUsers] = useState<any>([]);
 const [allOrders, setAllOrders] = useState<any>([]);
 const [reviews, setReviews] = useState<any>([]);
 const [replyTo, setReplyTo] = useState<string | null>(null);
 const [replyComment, setReplyComment] = useState("");
 const [searchTerm, setSearchTerm] = useState("");
```

```

const [selectedStatus, setSelectedStatus] = useState('all');
const [editingUser, setEditingUser] = useState<string | null>(null);
const [editingProduct, setEditingProduct] = useState<string | null>(null);
const { user, logout } = useAuth();
const { orders, updateOrderStatus } = useOrder();
const navigate = useNavigate();

useEffect(() => {
 if (!user || user.role !== 'ADMIN') {
 navigate('/admin/login');
 return;
 }
 loadDashboardData();

 // Auto-refresh every 30 seconds
 const interval = setInterval(loadDashboardData, 30000);
 return () => clearInterval(interval);
}, [user, orders]);

const loadDashboardData = async () => {
 try {
 const stats = await getPlantStatistics();
 setPlantStats(stats);

 const allOrdersData = JSON.parse(localStorage.getItem('all_orders') || '[]');
 const allUsersData = JSON.parse(localStorage.getItem('all_users') || '[]');
 const reviewsData = JSON.parse(localStorage.getItem('global-reviews') || '[]');
 }
}

```

```

setAllOrders(allOrdersData);
setAllUsers(allUsersData);
setReviews(reviewsData);

const totalRevenue = allOrdersData
 .filter((order: any) => order.status === 'delivered')
 .reduce((sum: number, order: any) => sum + order.total, 0);

const todayOrders = allOrdersData.filter((order: any) => {
 const orderDate = new Date(order.createdAt);
 const today = new Date();
 return orderDate.toDateString() === today.toDateString();
});

const dashboardData = {
 total_orders: allOrdersData.length,
 pending_orders: allOrdersData.filter((o: any) => o.status ===
 'pending').length,
 processing_orders: allOrdersData.filter((o: any) => o.status ===
 'processing').length,
}

```

## 20. src/pages/developer/DeveloperLogin.tsx

```

import React, { useState } from 'react';
import { Code, Mail, Lock, LogIn } from 'lucide-react';
import { useAuth } from '../contexts/AuthContext';
import { useNavigate, Link } from 'react-router-dom';

const DeveloperLogin: React.FC = () => {
 const { login } = useAuth() as any;
 const navigate = useNavigate();

```

```

const [form, setForm] = useState({ email: "", password: "" });
const [error, setError] = useState<string>("");
const [submitting, setSubmitting] = useState(false);

const onSubmit = async (e: React.FormEvent) => {
 e.preventDefault();
 setSubmitting(true);
 setError("");
 try {
 const user = await login(form.email, form.password);
 if (user?.role !== 'DEVELOPER') {
 setError('Akun ini tidak memiliki akses developer.');
 return;
 }
 navigate('/developer/dashboard', { replace: true });
 } catch (e: any) {
 setError(e?.message || 'Gagal login');
 } finally {
 setSubmitting(false);
 }
};

return (
 <div className="min-h-screen bg-gray-50 dark:bg-gray-900 flex items-center justify-center py-10 px-4">
 <div className="w-full max-w-md bg-white dark:bg-gray-800 border border-gray-200 dark:border-gray-700 rounded-xl p-6">
 <div className="flex items-center gap-2 mb-4">
 <Code className="h-8 w-8 text-green-600" />
 <h1 className="text-2xl font-bold text-gray-900 dark:text-"

```

```

white">Developer Portal</h1>
</div>
{error && <div className="mb-3 rounded-lg bg-red-50 dark:bg-red-900/30 border border-red-200 dark:border-red-800 p-3 text-red-700 dark:text-red-300">{error}</div>}
<form onSubmit={onSubmit} className="space-y-4">
<div className="relative">
<Mail className="absolute left-3 top-3 h-5 w-5 text-gray-400" />
<input
 type="email"
 required
 placeholder="Email developer"
 className="w-full pl-10 pr-3 py-2 rounded-lg bg-white dark:bg-gray-900 border border-gray-300 dark:border-gray-700 text-gray-900 dark:text-white"
 value={form.email}
 onChange={(e) => setForm({ ...form, email: e.target.value.trim() })}
/>
</div>
<div className="relative">
<Lock className="absolute left-3 top-3 h-5 w-5 text-gray-400" />
<input
 type="password"
 required
 placeholder="Password"
 className="w-full pl-10 pr-3 py-2 rounded-lg bg-white dark:bg-gray-900 border border-gray-300 dark:border-gray-700 text-gray-900 dark:text-white"
 value={form.password}

```

```

 onChange={(e) => setForm({ ...form, password: e.target.value })}
 />
 </div>
 <button
 type="submit"
 disabled={submitting}
 className="w-full inline-flex items-center justify-center gap-2 px-4 py-2 rounded-lg bg-green-600 hover:bg-green-700 text-white font-semibold disabled:opacity-70">
 >
 <LogIn className="h-5 w-5" /> {submitting ? 'Masuk...' : 'Masuk'}
 </button>
 <div className="text-center text-sm text-gray-500 dark:text-gray-400">
 <Link to="/" className="hover:underline">Kembali ke Beranda</Link>
 </div>
 </form>
 </div>
 </div>
);
};

export default DeveloperLogin;

```

## 21. src/pages/developer/DeveloperPortal.tsx

```

import React, { useState, useEffect } from 'react';
import { useAuth } from '../../contexts/AuthContext';
import { useNavigate } from 'react-router-dom';

```

```
import {
 Code,
 Database,
 Server,
 Activity,
 Users,
 ShoppingCart,
 TrendingUp,
 AlertTriangle,
 CheckCircle,
 Clock,
 Monitor,
 Cpu,
 HardDrive,
 Wifi,
 Globe,
 GitBranch,
 Bug,
 Zap,
 LogOut,
 User,
 RefreshCw,
 Terminal,
 Settings,
 FileText,
 Shield,
 Search,
 Play,
 Pause,
 RotateCcw,
}
```

```
Plus,
Eye,
Edit,
MessageSquare,
Reply,
Send
} from 'lucide-react';

const DeveloperPortal: React.FC = () => {
 const { user, logout } = useAuth();
 const navigate = useNavigate();
 const [activeTab, setActiveTab] = useState('monitoring');
 const [reviews, setReviews] = useState<any[][]>([]);
 const [replyTo, setReplyTo] = useState<string | null>(null);
 const [replyComment, setReplyComment] = useState("");
 const [systemStats, setSystemStats] = useState({
 uptime: '99.9%',
 responseTime: '120ms',
 activeUsers: 0,
 totalOrders: 0,
 revenue: 'Rp 0',
 errorRate: '0.0%',
 totalUsers: 0,
 activeSessions: 0,
 cpuUsage: 23,
 memoryUsage: 67,
 diskUsage: 45,
 networkLatency: 12
});
```

```

const [recentLogs, setRecentLogs] = useState([
 { id: 1, level: 'info', message: 'System startup completed successfully',
 timestamp: new Date(), service: 'system' },
 { id: 2, level: 'info', message: 'Database connection established',
 timestamp: new Date(Date.now() - 300000), service: 'database' },
 { id: 3, level: 'info', message: 'Authentication service ready', timestamp:
 new Date(Date.now() - 600000), service: 'auth' },
 { id: 4, level: 'info', message: 'All services operational', timestamp: new
 Date(Date.now() - 900000), service: 'system' }
]);

const [apiEndpoints] = useState([
 { name: 'User Authentication', endpoint: '/api/auth', status: 'healthy',
 responseTime: '45ms', uptime: '99.9%' },
 { name: 'Product Catalog', endpoint: '/api/products', status: 'healthy',
 responseTime: '67ms', uptime: '99.8%' },
 { name: 'Order Management', endpoint: '/api/orders', status: 'healthy',
 responseTime: '89ms', uptime: '99.7%' },
 { name: 'User Profile', endpoint: '/api/profile', status: 'healthy',
 responseTime: '123ms', uptime: '99.9%' },
 { name: 'Cart Service', endpoint: '/api/cart', status: 'healthy',
 responseTime: '56ms', uptime: '100%' },
 { name: 'Payment Gateway', endpoint: '/api/payments', status: 'healthy',
 responseTime: '234ms', uptime: '99.5%' }
]);

const [securityLogs] = useState([

```

## 22. src/pages/legal/PrivacyPolicy.tsx

```
import React from 'react';
```

```
import { Shield, Eye, Lock, Database, Cookie, Mail, Phone, MapPin } from
'@lucide-react';

const PrivacyPolicy: React.FC = () => {
 return (
 <div className="min-h-screen bg-white">
 {/* Hero Section */}
 <section className="bg-gradient-to-br from-green-600 to-green-800
text-white py-20">
 <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
 <div className="text-center">
 <Shield className="h-16 w-16 text-green-200 mx-auto mb-6" />
 <h1 className="text-5xl font-bold mb-4">Kebijakan Privasi</h1>
 <p className="text-xl text-green-100 max-w-2xl mx-auto">
 Komitmen kami untuk melindungi privasi dan data pribadi Anda
 </p>
 </div>
 </div>
 </section>

 <div className="max-w-4xl mx-auto px-4 sm:px-6 lg:px-8 py-16">
 <div className="bg-white rounded-xl shadow-lg p-8 border border-
gray-200">
 <div className="prose prose-lg max-w-none">
 <p className="text-gray-600 mb-8">
 Terakhir diperbarui: 15 Januari 2025
 </p>
 </div>
 </div>
 <section className="mb-8">
 <h2 className="text-2xl font-bold text-gray-900 mb-4 flex items-
```

```

center">

 <Eye className="h-6 w-6 text-green-600 mr-2" />
 Informasi yang Kami Kumpulkan
</h2>
<div className="text-gray-700 space-y-4">
 <p>Kami mengumpulkan informasi berikut untuk memberikan
layanan terbaik:</p>
 <ul className="list-disc pl-6 space-y-2">
 Informasi Pribadi: Nama, email, nomor
telepon, alamat pengiriman
 Informasi Akun: Username, password
(terenkripsi), preferensi pengguna
 Informasi Transaksi: Riwayat
pembelian, metode pembayaran, alamat penagihan
 Informasi Teknis: Alamat IP, browser,
perangkat, log aktivitas
 Komunikasi: Pesan chat, ulasan produk,
feedback pelanggan

</div>
</section>

<section className="mb-8">
 <h2 className="text-2xl font-bold text-gray-900 mb-4 flex items-
center">
 <Database className="h-6 w-6 text-green-600 mr-2" />
 Bagaimana Kami Menggunakan Informasi
 </h2>
 <div className="text-gray-700 space-y-4">
 <p>Informasi Anda digunakan untuk:</p>
 <ul className="list-disc pl-6 space-y-2">

```

```
Memproses pesanan dan pembayaran
Mengirim produk ke alamat yang benar
Memberikan dukungan pelanggan
```

### 23. src/pages/legal/TermsConditions.tsx

```
import React from 'react';
import { FileText, Scale, Shield, AlertTriangle, CheckCircle } from 'lucide-react';

const TermsConditions: React.FC = () => {
 return (
 <div className="min-h-screen bg-white">
 {/* Hero Section */}
 <section className="bg-gradient-to-br from-green-600 to-green-800 text-white py-20">
 <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
 <div className="text-center">
 <Scale className="h-16 w-16 text-green-200 mx-auto mb-6" />
 <h1 className="text-5xl font-bold mb-4">Syarat & Ketentuan</h1>
 <p className="text-xl text-green-100 max-w-2xl mx-auto">
 Ketentuan penggunaan layanan Azka Garden yang perlu Anda ketahui
 </p>
 </div>
 </div>
 </section>

 <div className="max-w-4xl mx-auto px-4 sm:px-6 lg:px-8 py-16">
 <div className="bg-white rounded-xl shadow-lg p-8 border border-
```

```
gray-200">
 <div className="prose prose-lg max-w-none">
 <p className="text-gray-600 mb-8">
 Terakhir diperbarui: 15 Januari 2025
 </p>

 <section className="mb-8">
 <h2 className="text-2xl font-bold text-gray-900 mb-4">1.
 Penerimaan Ketentuan</h2>
 <div className="text-gray-700 space-y-4">
 <p>
 Dengan mengakses dan menggunakan website Azka Garden,
 Anda menyetujui untuk terikat
 dengan syarat dan ketentuan ini. Jika Anda tidak setuju dengan
 ketentuan ini,
 mohon untuk tidak menggunakan layanan kami.
 </p>
 </div>
 </section>

 <section className="mb-8">
 <h2 className="text-2xl font-bold text-gray-900 mb-4">2.
 Layanan Kami</h2>
 <div className="text-gray-700 space-y-4">
 <p>Azka Garden menyediakan:</p>
 <ul className="list-disc pl-6 space-y-2">
 Penjualan tanaman hias dan aksesoris taman
 Jasa pembuatan dan renovasi taman
 Jasa pembuatan kolam ikan
 Konsultasi perawatan tanaman

 </div>
 </section>
 </div>

```

```

Edukasi melalui konten digital

</div>
</section>

<section className="mb-8">
 <h2 className="text-2xl font-bold text-gray-900 mb-4">3. Akun Pengguna</h2>
 <div className="text-gray-700 space-y-4">
 <p>Untuk menggunakan layanan tertentu, Anda perlu membuat akun dengan:</p>
 <ul className="list-disc pl-6 space-y-2">
 Informasi yang akurat dan lengkap
 Password yang aman dan rahasia
 Tanggung jawab atas semua aktivitas akun

 </div>
</section>

```

#### 24. src/pages/legal/CookiePolicy.tsx

```

import React from 'react';
import { Cookie, Settings, Eye, BarChart3, Target } from 'lucide-react';

const CookiePolicy: React.FC = () => {
 return (
 <div className="min-h-screen bg-white">
 {/* Hero Section */}
 <section className="bg-gradient-to-br from-green-600 to-green-800 text-white py-20">
 <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
 <div className="text-center">
 <Cookie className="h-16 w-16 text-green-200 mx-auto mb-6" />
 <h1 className="text-5xl font-bold mb-4">Kebijakan Cookie</h1>
 </div>
 </div>
 </section>
 </div>
);
}

export default CookiePolicy;

```

```
<p className="text-xl text-green-100 max-w-2xl mx-auto">
 Informasi tentang penggunaan cookie di website Azka Garden
</p>
</div>
</div>
</section>

<div className="max-w-4xl mx-auto px-4 sm:px-6 lg:px-8 py-16">
 <div className="bg-white rounded-xl shadow-lg p-8 border border-gray-200">
 <div className="prose prose-lg max-w-none">
 <p className="text-gray-600 mb-8">
 Terakhir diperbarui: 15 Januari 2025
 </p>

 <section className="mb-8">
 <h2 className="text-2xl font-bold text-gray-900 mb-4">Apa itu
 Cookie?</h2>
 <div className="text-gray-700 space-y-4">
 <p>
 Cookie adalah file teks kecil yang disimpan di perangkat Anda
 saat mengunjungi website.
 </p>
 <p>
 Cookie membantu kami memberikan pengalaman yang lebih
 baik dan personal.
 </p>
 </div>
 </section>

 <section className="mb-8">
 <h2 className="text-2xl font-bold text-gray-900 mb-4 flex items-
```

```
center">

 <Settings className="h-6 w-6 text-green-600 mr-2" />
 Jenis Cookie yang Kami Gunakan
 </h2>
 <div className="space-y-6">
 <div className="bg-green-50 p-4 rounded-lg border border-green-200">
 <h3 className="font-bold text-gray-900 mb-2">Cookie Esensial</h3>
 <p className="text-gray-700 text-sm">
 Diperlukan untuk fungsi dasar website seperti login, keranjang belanja, dan keamanan.
 </p>
 </div>

 <div className="bg-blue-50 p-4 rounded-lg border border-blue-200">
 <h3 className="font-bold text-gray-900 mb-2">Cookie Fungsional</h3>
 <p className="text-gray-700 text-sm">
 Mengingat preferensi Anda seperti bahasa, mata uang, dan pengaturan tampilan.
 </p>
 </div>

 <div className="bg-yellow-50 p-4 rounded-lg border border-yellow-200">
 <h3 className="font-bold text-gray-900 mb-2">Cookie Analitik</h3>
 <p className="text-gray-700 text-sm">
```

Membantu kami memahami bagaimana pengunjung menggunakan website untuk perbaikan layanan.

```
</p>
</div>
```

## 25. src/pages>Returns.tsx

```
import React from 'react';
import { RotateCcw, Shield, Clock, MessageCircle, CheckCircle, AlertTriangle, Leaf } from 'lucide-react';

const Returns: React.FC = () => {
 return (
 <div className="min-h-screen bg-white">
 {/* Hero Section */}
 <section className="bg-gradient-to-br from-green-600 to-green-800 text-white py-20">
 <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
 <div className="text-center">
 <RotateCcw className="h-16 w-16 text-green-200 mx-auto mb-6" />
 <h1 className="text-5xl font-bold mb-4">Kebijakan Pengembalian</h1>
 <p className="text-xl text-green-100 max-w-2xl mx-auto">
 Kami berkomitmen memberikan kepuasan pelanggan dengan kebijakan pengembalian yang fair dan mudah
 </p>
 </div>
 </div>
 </section>
 </div>
)
}
```

```

<div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 py-16">
 {/* Return Policy Overview */}
 <section className="mb-16">
 <h2 className="text-3xl font-bold text-gray-900 dark:text-white mb-8 text-center">Garansi & Pengembalian</h2>

 <div className="grid grid-cols-1 md:grid-cols-3 gap-8">
 <div className="bg-white p-6 rounded-xl shadow-lg text-center border-2 border-green-200">
 <div className="bg-green-100 w-16 h-16 rounded-full flex items-center justify-center mx-auto mb-4">
 <Shield className="h-8 w-8 text-green-600" />
 </div>
 <h3 className="text-xl font-bold text-gray-900 mb-3">Garansi Hidup</h3>
 <p className="text-gray-600">
 Semua tanaman dijamin hidup saat sampai di tangan Anda dengan kondisi sehat dan segar
 </p>
 </div>

 <div className="bg-white p-6 rounded-xl shadow-lg text-center border-2 border-green-200">
 <div className="bg-blue-100 w-16 h-16 rounded-full flex items-center justify-center mx-auto mb-4">
 <Clock className="h-8 w-8 text-blue-600" />
 </div>
 <h3 className="text-xl font-bold text-gray-900 mb-3">24 Jam Klaim</h3>
 <p className="text-gray-600">

```

Laporkan masalah dalam 24 jam setelah penerimaan untuk mendapatkan penggantian atau refund

```
</p>
</div>
```

```
<div className="bg-white p-6 rounded-xl shadow-lg text-center border-2 border-green-200">
```

```
 <div className="bg-purple-100 w-16 h-16 rounded-full flex items-center justify-center mx-auto mb-4">
```

```
 <CheckCircle className="h-8 w-8 text-purple-600" />
```

```
</div>
```

```
 <h3 className="text-xl font-bold text-gray-900 mb-3">Proses Mudah</h3>
```

```
<p className="text-gray-600">
```

```
 Proses klaim yang mudah dan cepat melalui WhatsApp dengan tim customer service kami
```

```
</p>
```

```
</div>
```

```
</div>
```

```
</section>
```

## 26. src/pages/Shipping.tsx

```
import React from 'react';
import { Truck, Package, Clock, Shield, MapPin, MessageCircle, Leaf } from 'lucide-react';
```

```
const Shipping: React.FC = () => {
```

```
 return (
```

```
 <div className="min-h-screen bg-gray-50 dark:bg-gray-900">
```

```
 /* Hero Section */
```

```

<section className="bg-gradient-to-br from-green-600 to-green-800 dark:from-gray-800 dark:to-black text-white py-20">
 <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
 <div className="text-center">
 <Truck className="h-16 w-16 text-green-200 mx-auto mb-6" />
 <h1 className="text-5xl font-bold mb-4">Informasi Pengiriman</h1>
 <p className="text-xl text-green-100 max-w-2xl mx-auto">
 Panduan lengkap pengiriman tanaman hias dengan packaging khusus untuk menjaga kualitas
 </p>
 </div>
 </div>
 </section>

 <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 py-16">
 {/* Shipping Methods */}
 <section className="mb-16">
 <h2 className="text-3xl font-bold text-gray-900 dark:text-white mb-8 text-center">Metode Pengiriman</h2>

 <div className="grid grid-cols-1 md:grid-cols-3 gap-8">
 <div className="bg-white dark:bg-gray-800 p-6 rounded-xl shadow-lg border border-gray-200 dark:border-gray-700">
 <div className="bg-green-100 dark:bg-green-900 w-12 h-12 rounded-lg flex items-center justify-center mb-4">
 <Truck className="h-6 w-6 text-green-600 dark:text-green-400" />
 </div>
 </div>
 </div>
 </section>
 </div>

```

```

<h3 className="text-xl font-bold text-gray-900 dark:text-white mb-3">Pengiriman Regular</h3>
 <p className="text-gray-600 dark:text-gray-300 mb-4">
 Estimasi 3-5 hari kerja untuk seluruh Indonesia
 </p>
 <div className="text-green-600 dark:text-green-400 font-semibold">Mulai dari Rp15.000</div>
</div>

<div className="bg-white dark:bg-gray-800 p-6 rounded-xl shadow-lg border border-gray-200 dark:border-gray-700">
 <div className="bg-blue-100 dark:bg-blue-900 w-12 h-12 rounded-lg flex items-center justify-center mb-4">
 <Package className="h-6 w-6 text-blue-600 dark:text-blue-400" />
 </div>
 <h3 className="text-xl font-bold text-gray-900 dark:text-white mb-3">Pengiriman Express</h3>
 <p className="text-gray-600 dark:text-gray-300 mb-4">
 Estimasi 1-2 hari kerja untuk area Jabodetabek
 </p>
 <div className="text-blue-600 dark:text-blue-400 font-semibold">Mulai dari Rp25.000</div>
</div>

<div className="bg-white dark:bg-gray-800 p-6 rounded-xl shadow-lg border border-gray-200 dark:border-gray-700">

```

```
import React from 'react';
import { Briefcase, MapPin, Clock, Users, Heart, Leaf, Send } from 'lucide-react';

interface JobPosition {
 id: string;
 title: string;
 department: string;
 location: string;
 type: 'Full-time' | 'Part-time' | 'Contract';
 description: string;
 requirements: string[];
 benefits: string[];
}

const Careers: React.FC = () => {
 const jobPositions: JobPosition[] = [
 {
 id: '1',
 title: 'Plant Care Specialist',
 department: 'Operations',
 location: 'Depok, Jawa Barat',
 type: 'Full-time',
 description: 'Bergabunglah dengan tim ahli tanaman kami untuk memberikan konsultasi dan perawatan terbaik kepada pelanggan.',
 requirements: [
 'Pengalaman minimal 2 tahun di bidang hortikultura',
 'Pengetahuan mendalam tentang tanaman hias indoor dan outdoor',
 'Kemampuan komunikasi yang baik',
 'Passion terhadap tanaman dan lingkungan'
]
 }
];
 return (
 <div>
 <h1>Job Positions</h1>
 <p>List of available job positions:</p>

 {jobPositions.map((position) => (

 Title: {position.title}
 Department: {position.department}
 Location: {position.location}
 Type: {position.type}
 Description: {position.description}
 Requirements: {position.requirements.length} items
 Benefits: {position.benefits.length} items

))}

 </div>
);
}
```

```
],
 benefits: [
 'Gaji kompetitif + bonus performa',
 'Pelatihan berkelanjutan',
 'Lingkungan kerja yang menyenangkan',
 'Kesempatan belajar dari para ahli'
]
 },
 {
 id: '2',
 title: 'Content Creator & Social Media',
 department: 'Marketing',
 location: 'Depok, Jawa Barat',
 type: 'Full-time',
 description: 'Bantu kami membuat konten edukatif untuk YouTube channel dan media sosial Azka Garden dengan 13k+ subscriber.',
 requirements: [
 'Pengalaman video editing dan content creation',
 'Familiar dengan platform YouTube, Instagram, TikTok',
 'Kreatif dan up-to-date dengan tren digital',
 'Basic knowledge tentang tanaman hias'
],
 benefits: [
 'Peluang berkreasi dengan konten viral',
 'Akses ke equipment recording profesional',
 'Kolaborasi dengan influencer tanaman',
 'Bonus berdasarkan engagement'
]
 },
 {

```

```

id: '3',
title: 'Landscape Designer',
department: 'Services',
location: 'Depok & Jabodetabek',
type: 'Contract',
description: 'Desain dan implementasi proyek taman untuk residential dan commercial. Termasuk pembuatan kolam ikan.',
requirements: [
 'Portfolio desain landscape yang menarik',
 'Pengalaman minimal 3 tahun',
 'Kemampuan menggunakan software desain',
 'Mobilitas tinggi untuk survey lokasi'
],
benefits: [
 'Project fee yang menarik',
 'Fleksibilitas waktu kerja',
 'Kesempatan proyek besar',
 'Networking dengan developer'
]
};

const [selectedJob, setSelectedJob] = React.useState<string | null>(null);

return (
 <div className="min-h-screen bg-gray-50 dark:bg-gray-900">
 {/* Hero Section */}
 <section className="bg-gradient-to-br from-green-600 to-green-800 dark:from-gray-800 dark:to-black text-white py-20">
 <div className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">

```

```
<div className="text-center">
```

## 28. src/pages/SearchResults.tsx

```
import React, { useEffect, useState } from 'react';
import { useSearchParams, useLocation } from 'react-router-dom';
import { Search, Filter, SlidersHorizontal, Grid, List } from 'lucide-react';
import ProductCard from '../components/ProductCard';
import CategoryFilter from '../components/CategoryFilter';
import { Plant, Category } from '../types';
import { smartSearch, SearchResult } from '../ai/search/SmartSearch';
import { getCategories } from '../services/database';

const SearchResults: React.FC = () => {
 const [searchParams] = useSearchParams();
 const location = useLocation();
 const query = searchParams.get('q') || "";

 const [searchResults, setSearchResults] = useState<SearchResult | null>(null);
 const [categories, setCategories] = useState<Category[]>([]);
 const [loading, setLoading] = useState(true);
 const [viewMode, setViewMode] = useState<'grid' | 'list'>('grid');
 const [showFilters, setShowFilters] = useState(false);
 const [selectedCategory, setSelectedCategory] = useState<string | null>(null);
 const [priceRange, setPriceRange] = useState<{ min: number; max: number }>({ min: 0, max: 1000000 });
 const [selectedCareLevel, setSelectedCareLevel] = useState<string[]>([]);
 const [sortBy, setSortBy] = useState<string>('relevance');
```

```

useEffect(() => {
 loadSearchResults();
 loadCategories();
}, [query, selectedCategory, priceRange, selectedCareLevel, sortBy]);

const loadSearchResults = async () => {
 if (!query && !location.state?.results) return;

 try {
 setLoading(true);

 let results: SearchResult;

 if (location.state?.results) {
 // Results from image search or other sources
 results = location.state.results;
 } else {
 // Text search
 const filters = {
 categories: selectedCategory ? [selectedCategory] : undefined,
 priceRange,
 careLevel: selectedCareLevel.length > 0 ? selectedCareLevel : undefined,
 inStock: true
 };

 results = await smartSearch.search(query, filters);
 }

 // Apply sorting
 }
}

```

```

results.items = sortResults(results.items, sortBy);

 setSearchResults(results);
} catch (error) {
 console.error('Failed to load search results:', error);
} finally {
 setLoading(false);
}
};

const loadCategories = async () => {
try {
 const categoriesData = await getCategories();
 setCategories(categoriesData);
} catch (error) {
 console.error('Failed to load categories:', error);
}
};

const sortResults = (items: Plant[], sortBy: string): Plant[] => {
const sorted = [...items];

switch (sortBy) {
 case 'price_low':
 return sorted.sort((a, b) => a.price - b.price);
 case 'price_high':
 return sorted.sort((a, b) => b.price - a.price);
 case 'name':
 return sorted.sort((a, b) => a.name.localeCompare(b.name));
 case 'newest':
}
}

```

```

return sorted.sort((a, b) => new Date(b.created_at).getTime() - new
Date(a.created_at).getTime());

default:

return sorted; // Keep original relevance order
}

};

const handleClearFilters = () => {
 setSelectedCategory(null);
 setPriceRange({ min: 0, max: 1000000 });
 setSelectedCareLevel([]);
}

```

## Lampiran 2. Tampilan Halaman Situs Web



Tampilan Halaman Beranda (<https://grand-pasca-0cde11.netlify.app/>)



Tampilan Halaman Tentang Kami (<https://grand-pasca-0cde11.netlify.app/about>)

Tampilan Halaman Kontak (<https://grand-pasca-0cde11.netlify.app/contact>)

Tampilan Halaman Tanya dan Jawab (<https://grand-pasca-0cde11.netlify.app/faq>)

Tampilan Halaman Katalog Produk (<https://grand-pasca-0cde11.netlify.app/products>)

**Koleksi Premium Azka Garden**

47+ Tanaman Hias premium dengan sistem pembayaran Stripe yang aman dan berlangganan bulanan

Stripe SSL Secure   Premium Quality   Monthly Delivery

Status Berlangganan

Detail Paket

Paket: Paket Premium   Harga: \$0.00/Bulan   Status: Aktif

Periode Billing

Paket Premium Mulai: 14/8/2025 Berakhir: 13/9/2025 Akhir Kartu: Visa \*\*\*\*4242

Batal Berlangganan   Ubah Metode Pembayaran

Terima kasih telah berlangganan!   Berikutnya   Beranda   Koleksi Premium   Bantuan   Log In

Tampilan Halaman Koleksi Premium (<https://grand-pasca-0cde11.netlify.app/stripe-products>)

**Newsletter Azka Garden**

Dapatkan tips perawatan tanaman dan penawaran khusus langsung di email Anda

Name (optional) Email Anda

Berlangganan & Akses Premium

**Azka Garden**

Tentang Azka Garden   Kontak Kami   FAQ & Bantuan   Informasi Pengiriman   Kebijakan Pengembalian   Panduan Perawatan Tanaman   Koleksi Premium   Blog & Tips   Karir

**Tautan Cepat**

Batu Taman   Media Tanah   Tanaman Hias   Pot & Aksesoris   Bonsai Premium   Tanaman Berguna

**Kategori Populer**

**Kontak Kami**

Jl. Raya KSU, Tirtajaya Kec. Sukmajaya, Kota Depok Jawa Barat 16412, Indonesia

Lihat di Google Maps

0896-3508-6182 WhatsApp & Telepon

infodagarden.com Email Support

Buka 24 Jam Setiap Hari

© 2025 Azka Garden - Toko Bunga Hendrik. Semua hak dilindungi undang-undang.

Kelijikan Privasi   Syarat & Ketentuan   Kelijikan Cookie

Tampilan Halaman Sukses Pembayaran (<https://grand-pasca-0cde11.netlify.app/stripe-success>)

**Panduan Perawatan Tanaman**

Tips dan trik lengkap untuk merawat tanaman hias agar tumbuh sehat dan indah

Cari panduan perawatan tanaman...

Kategori: Semua Kategori   Tipe: Kelelahan   Sesuai Level: Semua Level

**Jamani Dolar (ZZ Plant)**

Zamioculcas zamiifolia adalah tanaman indoor yang sangat toleran dan mudah dirawat. Cocok untuk pemula yang baru memulai hobi tanaman hias.

- Pengiriman: 2-3 minggu setelah pesanan diterima
- Catatan: Cukup resisten terhadap lingkungan sekitar
- Catatan: Cukup resisten terhadap lingkungan sekitar
- Pupuk: Pupuk tanah setiap 4-6 minggu sekali

**Monstera Deliciosa**

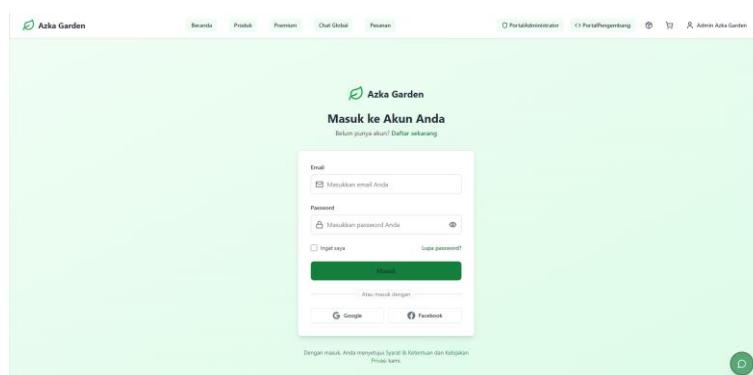
Tanaman hias populer dengan daun berbulu yang unik. Tumbuh menarik dan membutuhkan support untuk pertumbuhan optimal.

- Pengiriman: 1-2 hari kerja, setelah dikonfirmasi pembelian
- Catatan: Cukup resisten terhadap lingkungan sekitar
- Catatan: Cukup resisten terhadap lingkungan sekitar
- Pupuk: Pupuk NPK setiap 2-3 minggu sekali

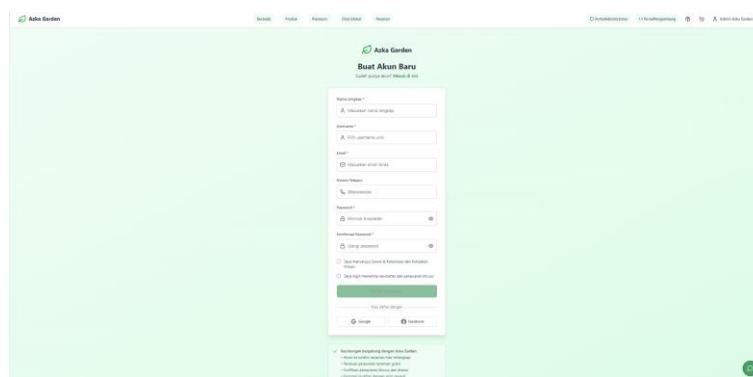
Tampilan Halaman Panduan Perawatan (<https://grand-pasca-0cde11.netlify.app/care-guide>)



Tampilan Halaman Blog dan Tips (<https://grand-pasca-0cde11.netlify.app/blog>)



Tampilan Login (<https://grand-pasca-0cde11.netlify.app/login>)



Tampilan Halaman Registrasi (<https://grand-pasca-0cde11.netlify.app/register>)

This screenshot shows the Admin Azka Garden profile page. It includes sections for 'Informasi Profil' (Profile Information) with details like name, email, phone number, and address; 'Berengganus' (Willingness) with a status of 'Belum Berengganan'; 'Statistik Akun' (Account Statistics) showing 1 follower, 1 subscriber, and 0 posts; and 'Informasi Akun' (Account Information) with account status set to 'Aktif'.

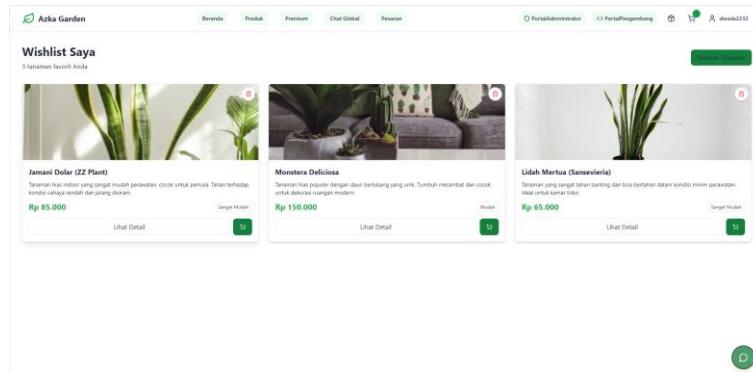
Tampilan Halaman Profil (<https://grand-pasca-0cde11.netlify.app/profile>)

This screenshot shows the order confirmation page for order #ORD-2025-001 placed on 11 August 2025 at 03:29. The total price is Rp 103.800. It details two items: 'Alamanda Kuning' and 'Aloe Vera'. Payment method is 'Transfer Mandiri' and delivery is 'Pengiriman Pengiriman Reguler'. A newsletter sign-up form is also present.

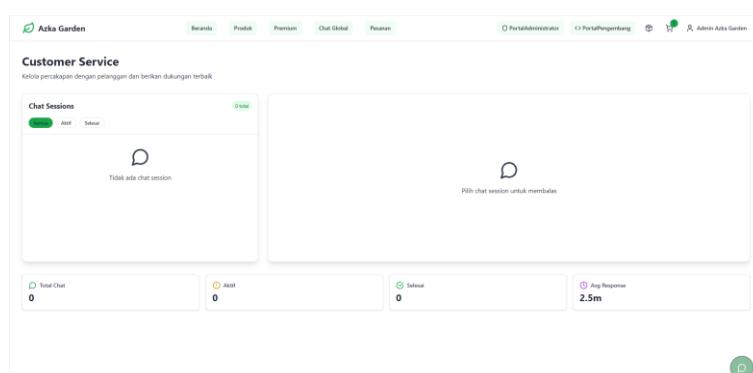
Tampilan Pesanan (<https://grand-pasca-0cde11.netlify.app/orders>)

This screenshot shows the shopping cart page. It lists 'Alamanda Kuning' and 'Aloe Vera' with their respective prices. The total amount is Rp 95.000. A newsletter sign-up form is included.

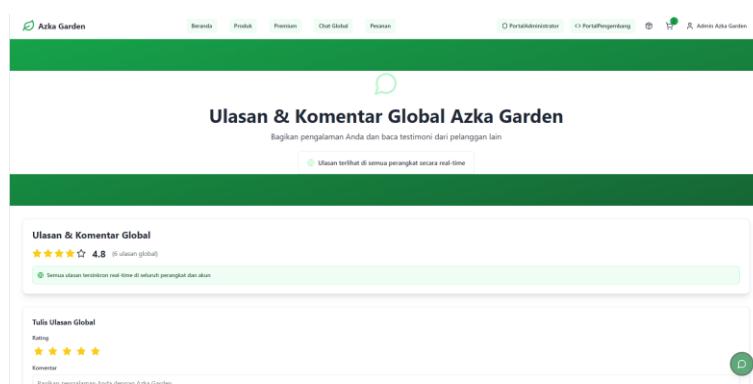
Tampilan Halaman Keranjang Belanja (<https://grand-pasca-0cde11.netlify.app/cart>)



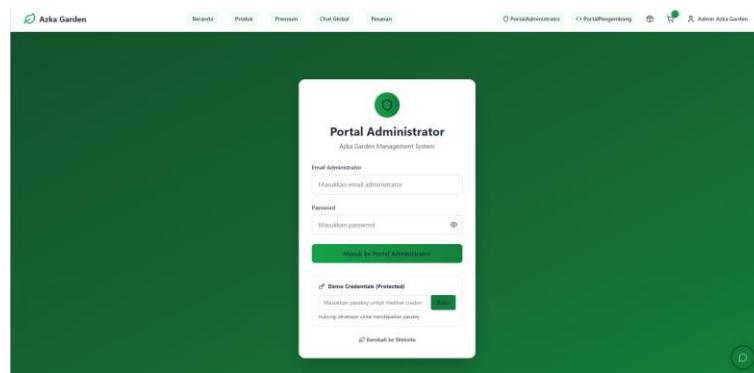
Tampilan Halaman Wishlist (<https://grand-pasca-0cde11.netlify.app/wishlist>)



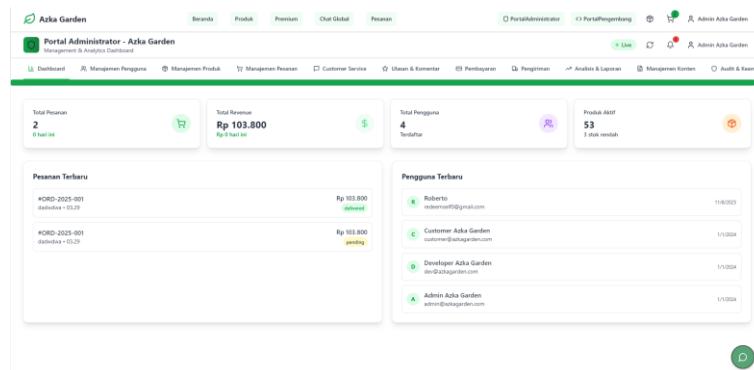
Tampilan Halaman Layanan Pelanggan (<https://grand-pasca-0cde11.netlify.app/customer-service>)



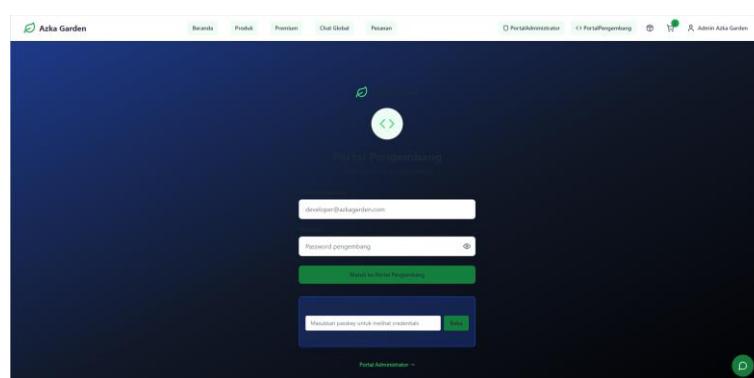
Tampilan Halaman Ulasan dan Komentar (<https://grand-pasca-0cde11.netlify.app/reviews>)



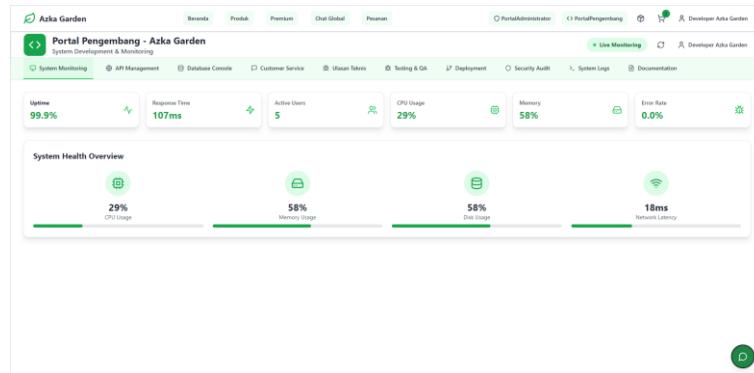
Tampilan Halaman Portal Administrator (<https://grand-pasca-0cde11.netlify.app/admin/login>)



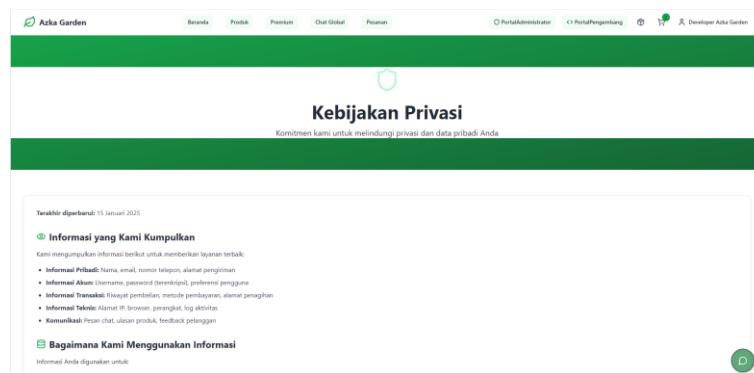
Tampilan Halaman Dasboar Administrator (<https://grand-pasca-0cde11.netlify.app/admin/dashboard>)



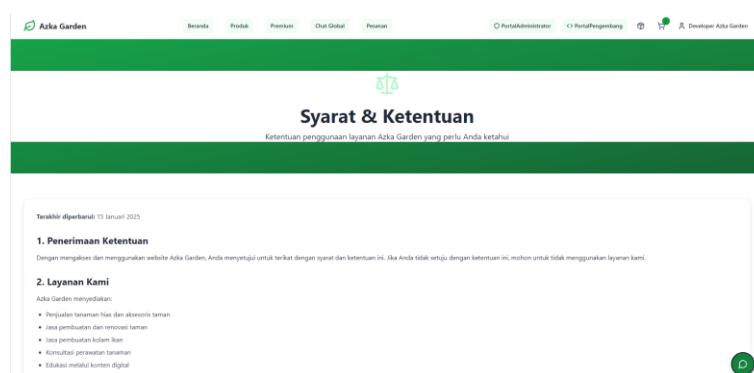
Tampilan Halaman Portal Pengembang (<https://grand-pasca-0cde11.netlify.app/developer/login>)



Tampilan Halaman Dasbor Pengembang (<https://grand-pasca-0cde11.netlify.app/developer/dashboard>)



Tampilan Halaman Kebijakan Privasi (<https://grand-pasca-0cde11.netlify.app/privacy>)



Tampilan Halaman Syarat dan Ketentuan (<https://grand-pasca-0cde11.netlify.app/terms>)

The screenshot shows the cookie policy page for Azka Garden. The header includes the logo, navigation links (Beranda, Produk, Premium, Chat Global, Pesanan), and user account links (PortalAdministrator, PortalPengembang, Developer Azka Garden). The main content area has a green header with the title "Kebijakan Cookie" and a sub-header "Informasi tentang penggunaan cookie di website Azka Garden". Below this, a note states "Terakhir diperbarui: 15 Januari 2025". A section titled "Apa itu Cookie?" explains that cookies are small text files stored in a browser. The page then lists two types of cookies: "Jenis Cookie yang Kami Gunakan", which includes "Cookie Essential" (for site navigation) and "Cookie Fungsional" (for user experience like language and payment methods).

Tampilan Halaman Kebijakan Kuki (<https://grand-pasca-0cde11.netlify.app/cookies>)

The screenshot shows the return policy page for Azka Garden. The header is identical to the previous page. The main content area has a green header with the title "Kebijakan Pengembalian" and a sub-header "Kami berkomitmen memberikan kepuasan pelanggan dengan kebijakan pengembalian yang fair dan mudah". Below this, a section titled "Garansi & Pengembalian" contains three boxes: "Garansi Hidup" (All plants are guaranteed healthy until交回), "24 Jam Klaim" (Claims must be made within 24 hours after receiving the product to receive a refund), and "Proses Mudah" (Easy and fast process with customer service). At the bottom, there is a section titled "Kondisi Pengembalian".

Tampilan Halaman Kebijakan Pengembalian (<https://grand-pasca-0cde11.netlify.app/returns>)

The screenshot shows the shipping information page for Azka Garden. The header is identical. The main content area has a green header with the title "Informasi Pengiriman" and a sub-header "Panduan lengkap pengiriman tanaman hasil dengan packaging khusus untuk menjaga kualitas". Below this, a section titled "Metode Pengiriman" contains three boxes: "Pengiriman Regular" (Estimated 2-5 days delivery across Indonesia, starting from Rp15.000), "Pengiriman Express" (Estimated 1-2 days delivery for Jakarta area, starting from Rp25.000), and "Ambil di Toko" (Direct pickup at the store in Depok, gratis). At the bottom, there is a section titled "Packaging Khusus Tanaman".

Tampilan Halaman Informasi Pengiriman (<https://grand-pasca-0cde11.netlify.app/shipping>)

Tampilan Halaman Karir (<https://grand-pasca-0cde11.netlify.app/careers>)

Tampilan Halaman Informasi Pengiriman (<https://grand-pasca-0cde11.netlify.app/search?q=monstera>)

Tampilan Halaman Sitemap (<https://grand-pasca-0cde11.netlify.app/sitemap>)