BAB IV
HASIL PENELITIAN

4.1 RINGKASAN HASIL PENELITIAN

Penelitian ini berhasil merancang dan membangun *website e-commerce* untuk Arabic Store guna mendukung strategi *digital marketing* yang lebih terstruktur dan efisien. Sistem dikembangkan menggunakan metode *Waterfall* dengan teknologi Django dan basis data PostgreSQL.

Fitur utama yang diimplementasikan meliputi katalog produk, sistem keranjang belanja, halaman detail produk, form kontak, chekout serta dua komponen berbasis kecerdasan buatan: sistem rekomendasi produk berbasis *Content-Based Filtering* dan *chatbot* layanan pelanggan yang memanfaatkan *Natural Language Processing (NLP)*. Sistem rekomendasi berhasil menampilkan produk relevan berdasarkan kemiripan deskripsi dan kategori, sedangkan *chatbot* mampu memberikan respon otomatis terhadap pertanyaan umum pengguna.

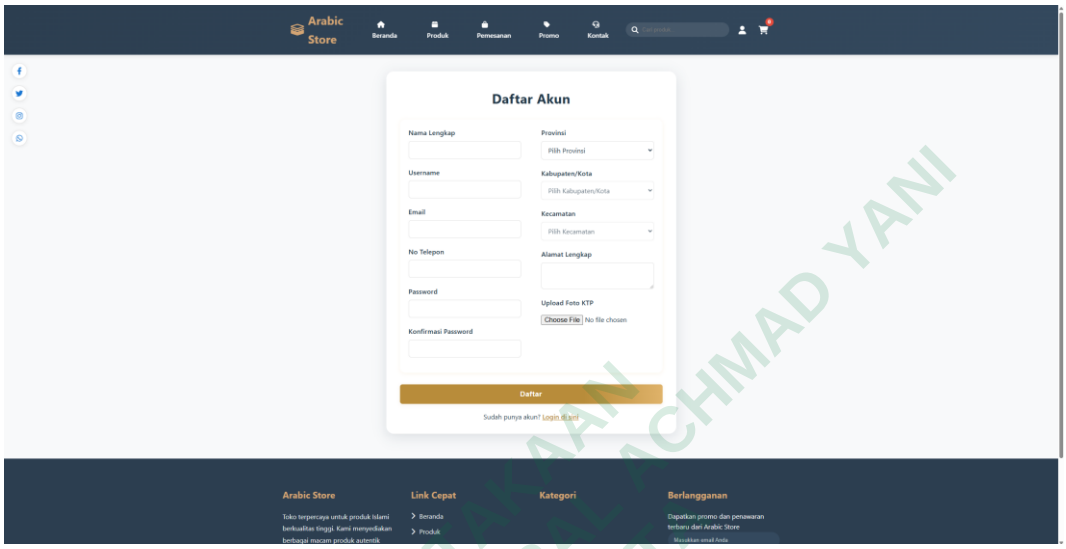
4.2 IMPLEMENTASI DESAIN ANTARMUKA

Implementasi antarmuka pengguna (*User Interface*) dirancang agar sederhana, mudah digunakan, dan fungsional. Perancangan UI/UX dilakukan dengan pendekatan berbasis data dan kebutuhan pengguna Arabic Store. Desain UI/UX dibuat untuk memudahkan navigasi pengguna, dengan tampilan bersih, responsif, dan informatif. Desain mengikuti prinsip UI/UX modern yang banyak digunakan di e-commerce. Hal ini mempermudah pengguna dalam menjelajahi produk dan menyelesaikan transaksi tanpa kebingungan. Berikut adalah penjelasan tiap halaman:

4.2.1 Halaman *Register*

Halaman *Register* pada sistem Arabic Store digunakan oleh pengguna baru untuk membuat akun. Pengguna mengisi form yang mencakup nama lengkap, *username*, email, nomor telepon, *password*, serta alamat lengkap. Form ini dilengkapi dropdown dinamis untuk memilih provinsi, kabupaten/kota, dan kecamatan menggunakan API wilayah Binderbyte. Selain itu, tersedia fitur unggah

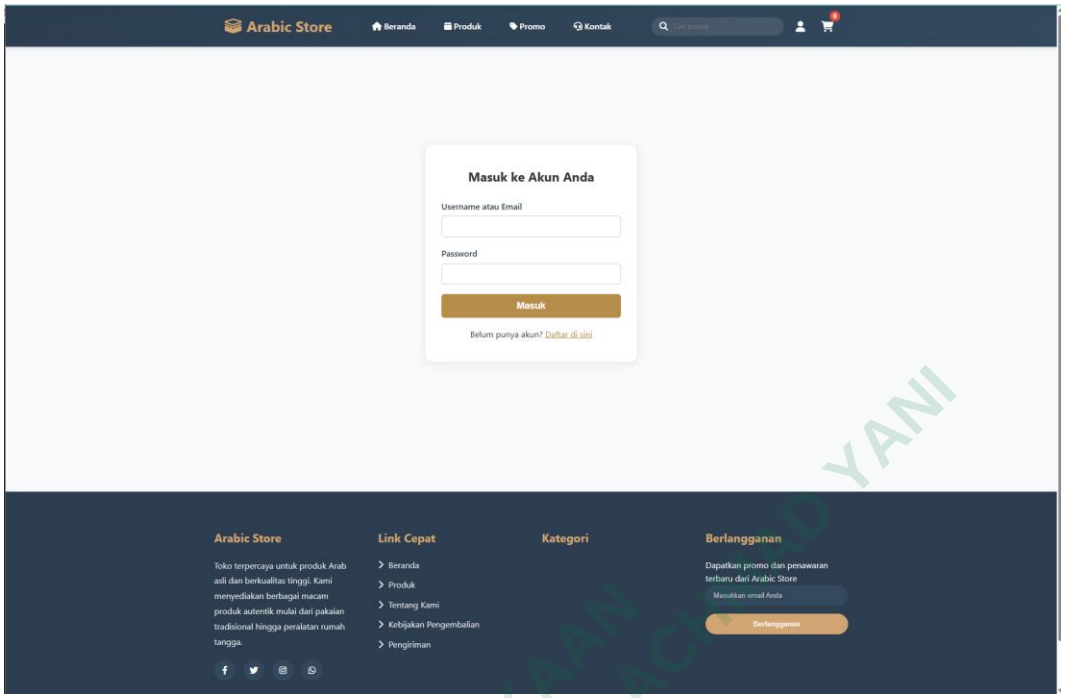
foto KTP sebagai verifikasi identitas. Data yang dikirim akan disimpan di *database* dan otomatis ditampilkan pada halaman profil serta digunakan saat *checkout*, sehingga pengguna tidak perlu mengisi ulang alamat dan informasi pribadi. Gambar 4.1 menampilkan halaman *register*.



Gambar 0.1 Halaman *Register*

4.2.2 Halaman *Login*

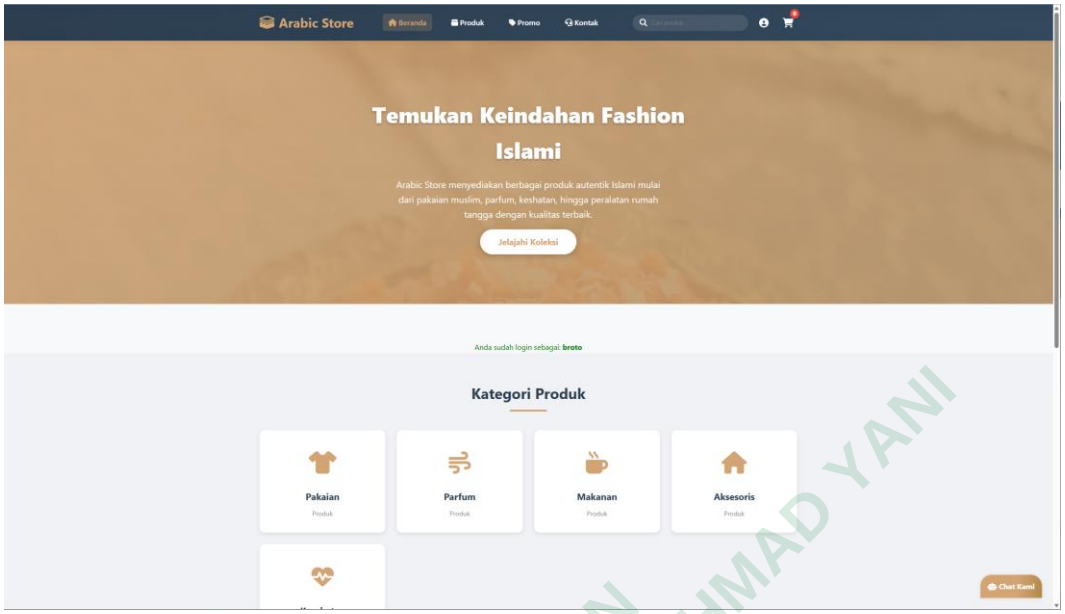
Halaman *login* digunakan untuk mengautentikasi pengguna sebelum mengakses halaman profil dan fitur personal lainnya. Halaman ini menampilkan form *username* dan *password*, dengan tombol *login* serta tautan ke halaman *register* jika pengguna belum memiliki akun. Desainnya sederhana dan responsif, dengan tampilan yang konsisten di perangkat desktop maupun mobile. Gambar 4.2 menampilkan halaman *login* dan Gambar 4.3 menampilkan validasi setelah *login*.



Gambar 0.2 Tampilan Halaman *Login*

```
def login_view(request):
    if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
                user    =    authenticate(request,    username=username,
password=password)
        if user is not None:
            login(request, user)
            # Set flag login sukses di session
            request.session['login_success'] = True
            return redirect('store:homepage')
        else:
            from django.contrib import messages
            messages.error(request, 'Username atau password salah.')
    return render(request, 'store/login.html')
```

Kode ini berfungsi untuk mengautentikasi pengguna berdasarkan input *username* dan *password* yang dikirim menggunakan metode POST. Jika autentikasi berhasil, pengguna akan diarahkan ke halaman beranda (*homepage*) dan sesi *login* akan ditandai berhasil. Jika gagal, sistem akan menampilkan pesan error menggunakan Django messages *framework*. Pada tampilan HTML, ditambahkan juga notifikasi sementara untuk menampilkan status *login* sukses.



Gambar 0.3 Tampilan Validasi *Login*

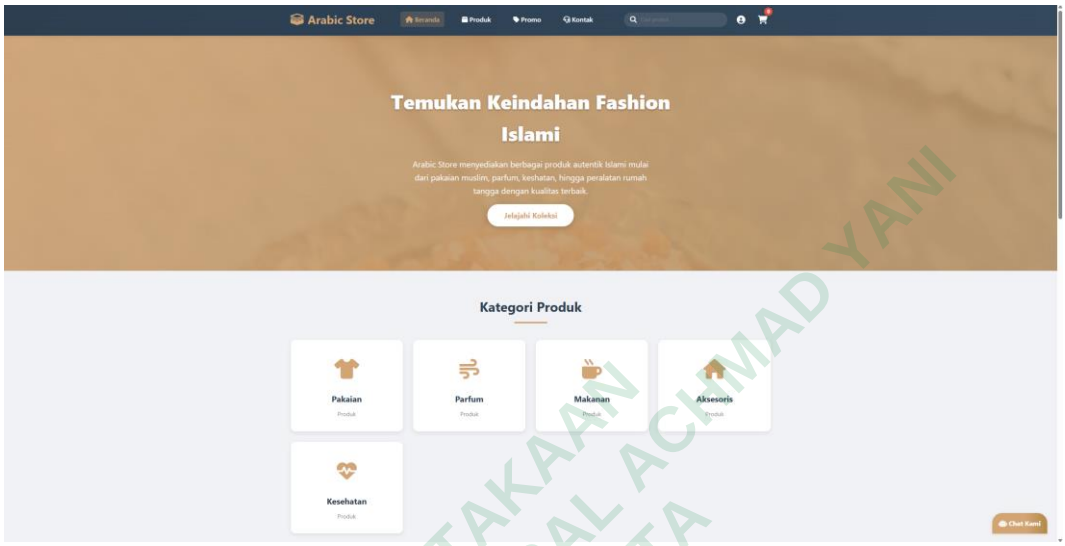
```
    {% if login_success %}
    <div id="login-status" style="text-align:center;margin-top:80px;">
            <p  style="color:green;">Anda  sudah  login  sebagai:  <b>{{
user.username }}</b></p>
    </div>
    <script>
    // Hilangkan pesan login setelah 5 detik
    window.addEventListener('DOMContentLoaded', function() {
        var statusDiv = document.getElementById('login-status');
        if (statusDiv) {
            setTimeout(function() {
                statusDiv.style.display = 'none';
            }, 5000);
        }
    });
    </script>
```

Kode ini berguna untuk meningkatkan pengalaman pengguna (UX) dengan memberikan feedback visual bahwa *login* berhasil dengan menampilkan *username*, tanpa harus menampilkan notifikasi permanen. Setelah 5 detik, pesan akan hilang otomatis, menjaga tampilan halaman tetap bersih.

4.2.3 Halaman Beranda

Halaman ini merupakan halaman pertama yang diakses oleh pengunjung. Menampilkan banner promosi, daftar kategori produk, serta produk unggulan. Navigasi utama ditempatkan di bagian atas, memudahkan pengguna menjelajahi katalog produk, *login*, keranjang belanja, dan halaman profil. Desain beranda

menitikberatkan pada visual yang menarik dan CTA (Call to Action) yang jelas seperti “Beli Sekarang” dan “Lihat Produk”. Gambar 4.4 menampilkan halaman beranda dan Gambar 4.5 memperlihatkan rekomendasi produk unggulan di beranda.

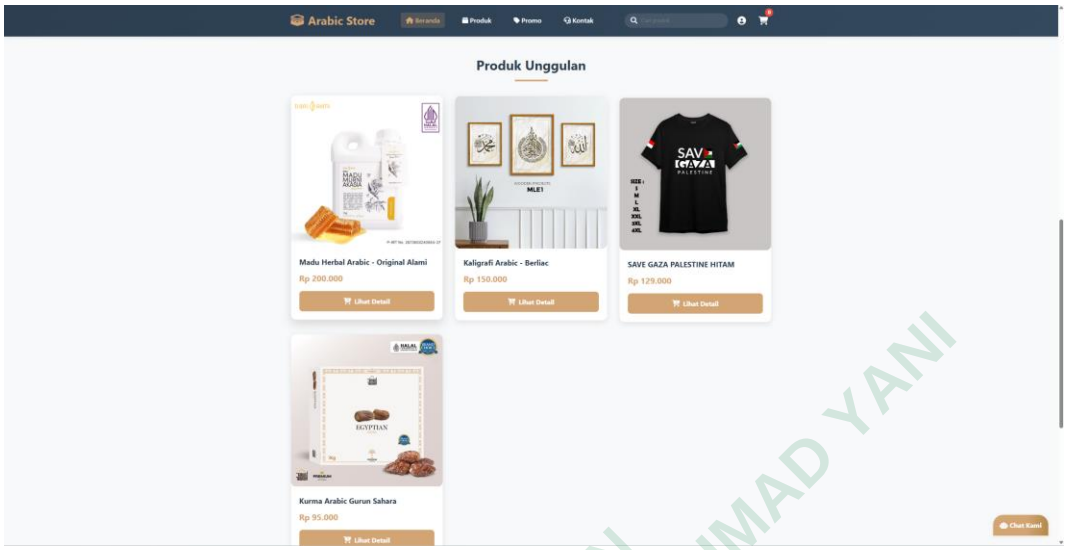


Gambar 0.4 Tampilan Halaman Beranda

```
def homepage(request):
    categories = Category.objects.all()[:4]
    all_products = list(Product.objects.filter(available=True))
        featured_products = sample(all_products, 4) if
len(all_products) > 4 else all_products

    login_success = request.session.pop('login_success', False)
    return render(request, 'store/homepage.html', {
        'categories': categories,
        'featured_products': featured_products,
        'login_success': login_success,
    })
```

Fungsi homepage digunakan untuk mengambil data kategori dan produk unggulan dari *database*. Produk unggulan dipilih secara acak dan ditampilkan pada halaman utama. Selain itu, halaman ini juga menampilkan notifikasi *login* jika *login* baru saja dilakukan. Tampilan visual ditekankan pada banner, kategori, dan produk

unggulan untuk menarik perhatian pengunjung sejak awal.



Gambar 0.5 Tampilan Rekomendasi Produk Unggulan

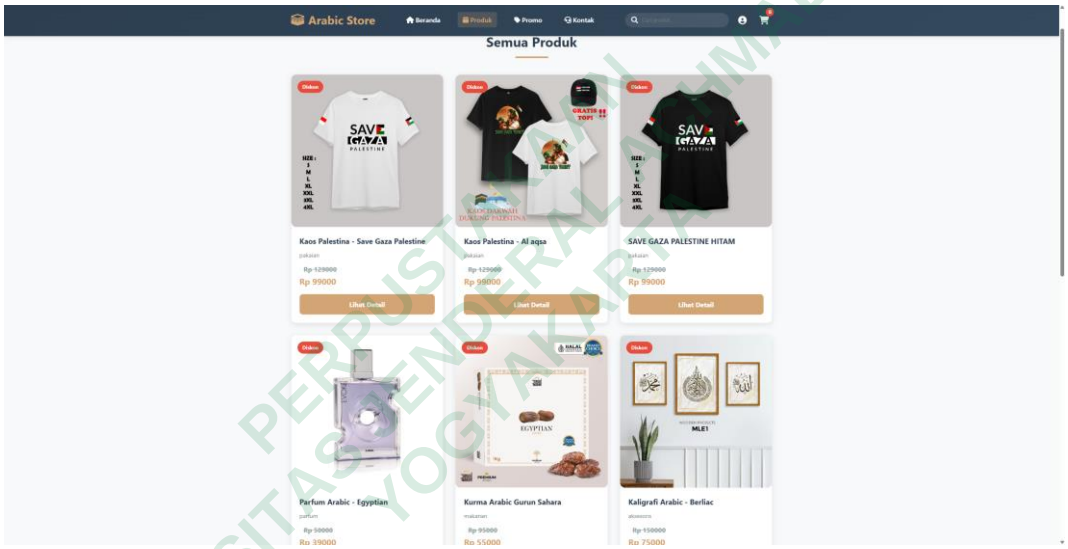
```
<section class="product-section">
 <div class="container">
   <h2 class="section-title">Produk Unggulan</h2>
    <div class="product-grid">
       {% for product in featured_products %}
       <div class="product-card">
       <div class="product-image">
           {% if product.image %}
           <img src="{{ product.image.url }}" alt="{{ product.name }}">
           {% else %}
           <img src="{% static 'images/no_image.png' %}" alt="No image">
           {% endif %}
           </div>
           <div class="product-info">
               <div class="product-name">{{ product.name }}</div>
                <div class="product-price" style="display: flex; flex-direction:
column; align-items: flex-start;">
                {% if product.discount_price %}
                <span class="original-price">Rp {{ product.price }}</span>
                      <span  class="discount-price">Rp  {{  product.discount_price
}}</span>
                       {% else %}
                       Rp {{ product.price }}
                       {% endif %}
               </div>
                 <a href="{{ product.get_absolute_url }}" class="add-to-cart">
                   <i class="fas fa-cart-plus"></i> Lihat Detail
                 </a>
               </div>
           </div>
```

Kode tersebut digunakan untuk menampilkan daftar produk unggulan pada halaman beranda *website* Arabic Store. Melalui perulangan featured_products, setiap produk ditampilkan dengan gambar, nama, harga (termasuk harga diskon jika tersedia), serta tombol “Lihat Detail” yang mengarahkan ke halaman detail produk. Jika gambar produk tidak tersedia, maka ditampilkan gambar default. Kode ini

bertujuan untuk memberikan rekomendasi produk yang dianggap menarik atau populer, sehingga mendorong pengguna untuk menjelajahi lebih lanjut dan meningkatkan potensi pembelian.

4.2.4 Halaman Katalog

Pada halaman ini pengguna dapat melihat semua produk yang tersedia dalam bentuk grid. Setiap produk ditampilkan dalam kartu (*card*) yang berisi gambar, nama, dan harga. Fitur filter berdasarkan kategori dan pencarian produk tersedia untuk memudahkan pengguna menemukan produk sesuai kebutuhan mereka. Berikut Gambar 4.6 yang menampilkan halaman katalog produk.



Gambar 0.6 Tampilan Halaman Katalog

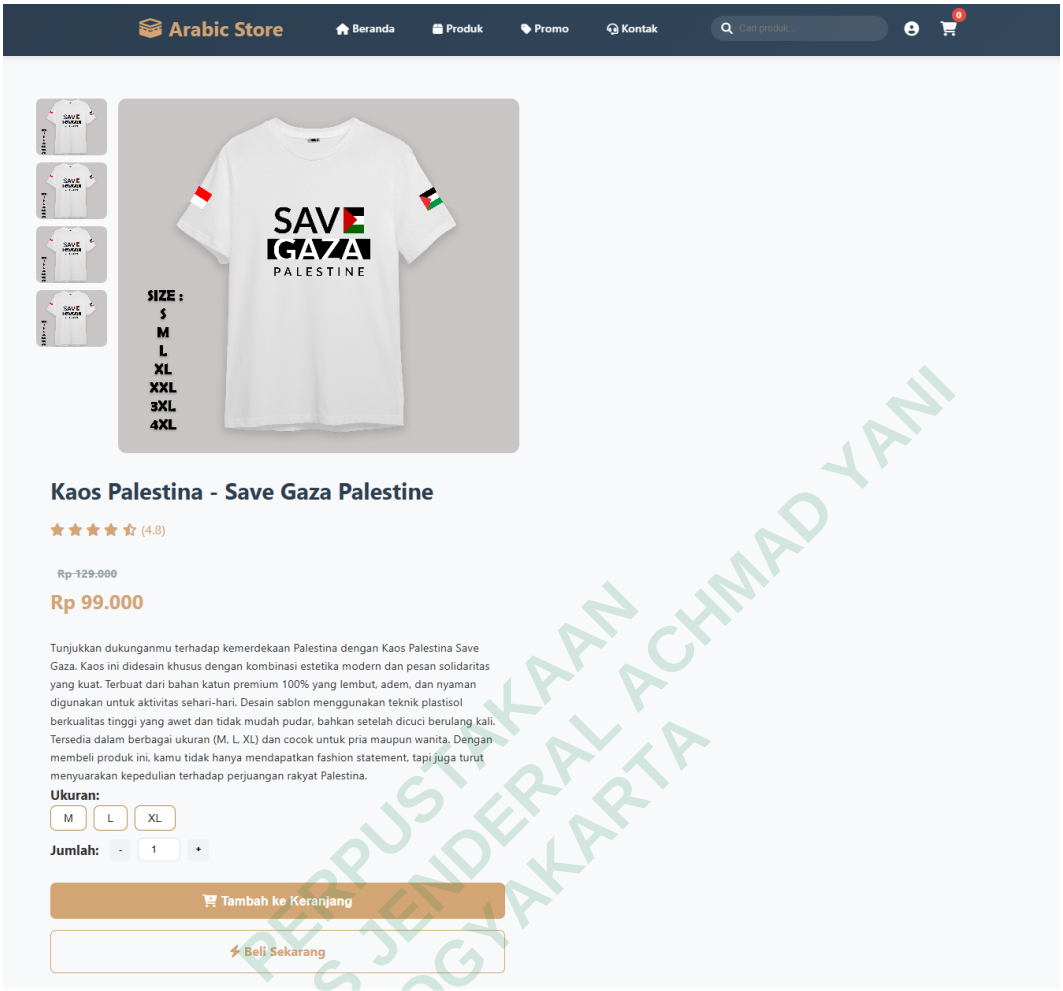
```
def catalog(request):
    category_slug = request.GET.get('category')
    query = request.GET.get('q', '').strip()
    categories = Category.objects.all()

    products = Product.objects.filter(available=True)
    if category_slug:
       category = get_object_or_404(Category, slug=category_slug)
        products = products.filter(category=category)
    else:
        category = None
    if query:
        products = products.filter(name__icontains=query)

    paginator = Paginator(products, 12)
    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)
```

```
    return render(request, 'store/catalog.html', {
        'category': category,
        'categories': categories,
        'products': page_obj,
        'query': query,
    })
```

Kode ini mengambil dan menampilkan seluruh produk dari *database* yang masih tersedia (available=True). Pengguna dapat menyaring produk berdasarkan kategori dan kata kunci pencarian. Produk ditampilkan dalam bentuk *pagination* (dibagi per halaman) agar halaman tetap ringan dan mudah diakses. Fungsi ini mendukung personalisasi pencarian produk oleh pengguna.

4.2.5 Halaman Detail Produk

Ketika pengguna mengklik salah satu produk, mereka diarahkan ke halaman ini. Halaman ini menampilkan informasi lengkap tentang produk, termasuk foto besar, deskripsi lengkap, harga, stok, dan tombol tambah ke keranjang. Sistem rekomendasi berbasis NLP juga ditampilkan di bagian bawah, memberikan saran produk serupa yang relevan dengan produk yang dilihat. Gambar 4.7 memperlihatkan tampilan halaman detail produk.



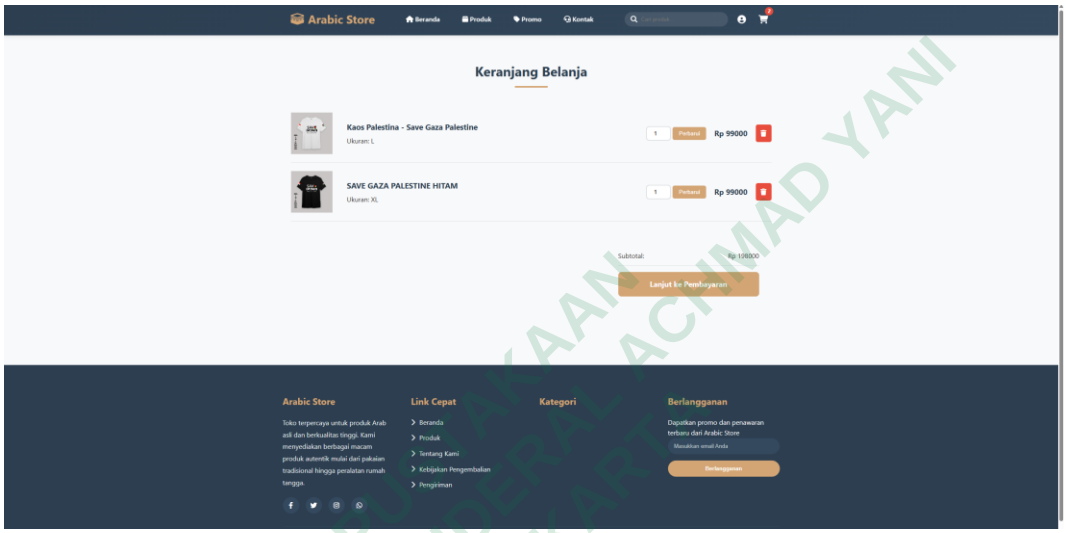
Gambar 0.7 Tampilan Halaman Detail Produk

```
def product_detail(request, id, slug):
      product = get_object_or_404(Product, id=id, slug=slug,
available=True)
    recommended_products = get_recommendations(product, top_n=4)
                                variation_types                =
product.variation_types.prefetch_related('values').all()
    return render(request, 'store/product.html', {
        'product': product,
        'related_products': recommended_products,
        'variation_types': variation_types,
    })
```

Fungsi product_detail menampilkan detail lengkap dari produk yang dipilih oleh pengguna berdasarkan id dan slug. Selain itu, sistem juga menampilkan rekomendasi produk lain yang serupa pada produk terkait berdasarkan *Content-Based Filtering* dengan bantuan NLP. Tujuannya agar pengguna menemukan alternatif atau produk pendukung dari yang sedang dilihat.

4.2.6 Halaman Keranjang Produk

Menampilkan daftar produk yang telah dipilih oleh pengguna. Tersedia fitur untuk menambah/mengurangi jumlah produk, menghapus produk dari keranjang, serta melihat total harga keseluruhan sebelum *checkout*. Desain halaman ini dibuat sesederhana mungkin untuk memfokuskan pengguna pada tindakan pembelian. Berikut tampilan halaman keranjang produk pada Gambar 4.8.



Gambar 0.8 Tampilan Halaman Keranjang Produk

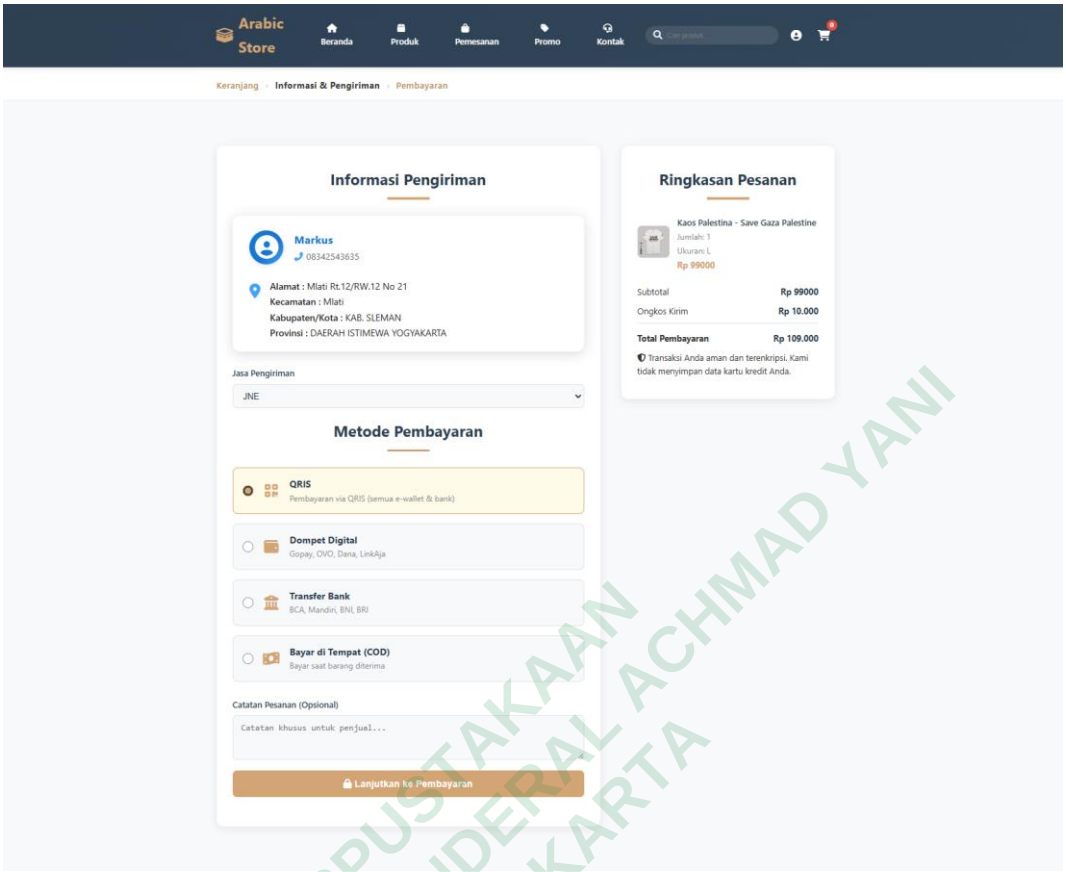
```
def cart_detail(request):
    cart = Cart(request)
    return render(request, 'store/cart.html', {'cart': cart})
def cart_add(request, product_id):
    cart = Cart(request)
    product = get_object_or_404(Product, id=product_id)
    size = request.POST.get('size') or request.GET.get('size') or
'L'
    try:
        quantity = int(request.POST.get('quantity', 1))
    except (TypeError, ValueError):
        quantity = 1
     if request.method == 'POST' and 'quantity' in request.POST
and request.POST.get('size'):
        override = True
    else:
        override = False
       cart.add(product=product,  quantity=quantity,  size=size,
override_quantity=override)
    return redirect('store:cart_detail')
def cart_remove(request, product_id):
    cart = Cart(request)
    product = get_object_or_404(Product, id=product_id)
    size = request.GET.get('size') or request.POST.get('size') or
'L'
```

```
    cart.remove(product, size=size)
    return redirect('store:cart_detail')
def cart_clear(request):
    cart = Cart(request)
    cart.clear()
    return redirect('store:cart_detail')
```

Bagian ini mengatur tampilan dan interaksi pengguna terhadap keranjang belanja. Fungsi *cart_detail* menampilkan seluruh produk yang telah ditambahkan ke keranjang. Fungsi *cart_add*, *cart_remove*, dan *cart_clear* digunakan untuk menambah, menghapus, atau mengosongkan isi keranjang belanja. Seluruh proses ini menjaga agar keranjang tetap responsif terhadap tindakan pengguna.

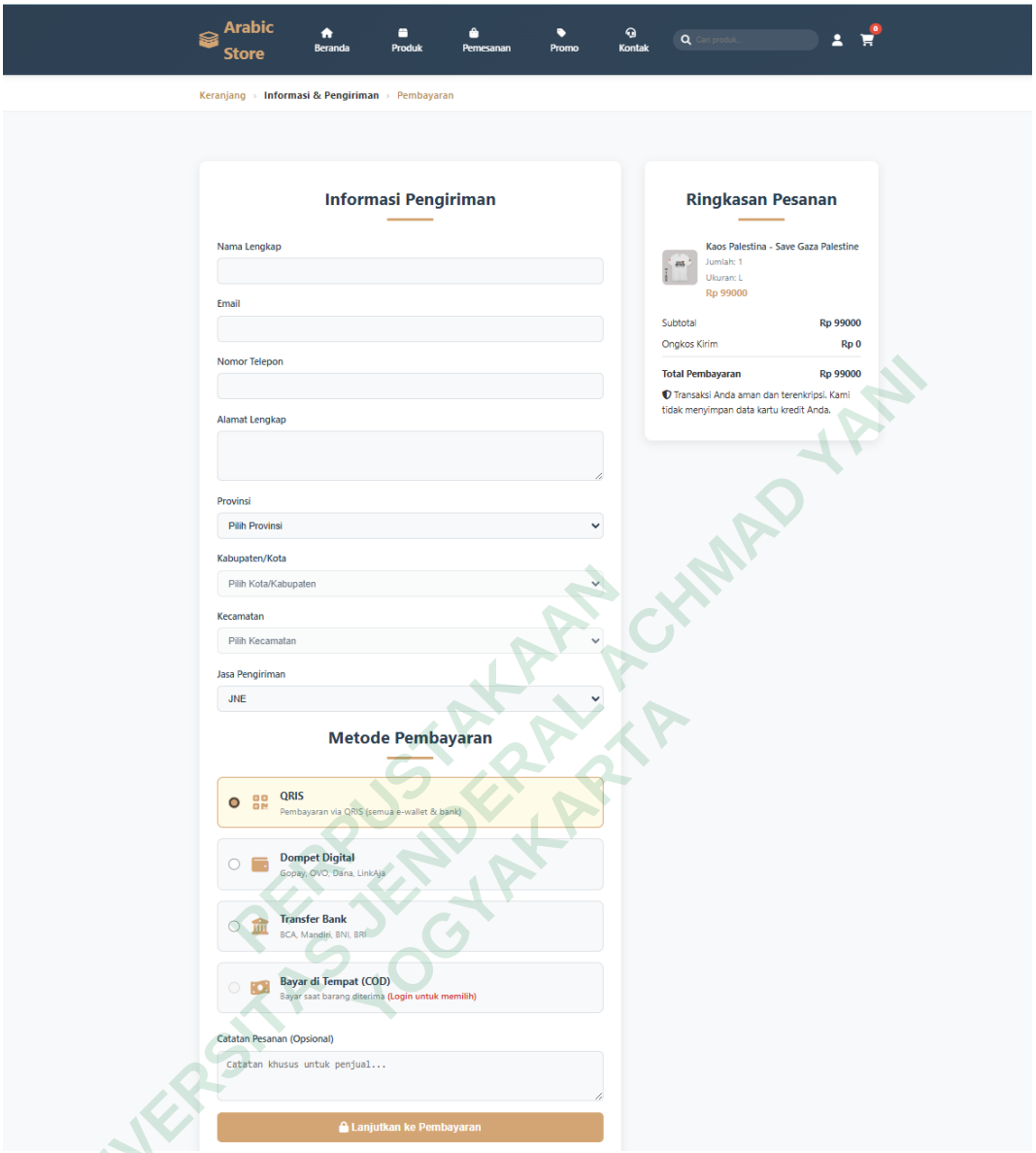
4.2.7 Halaman *Checkout*

Halaman *Checkout* merupakan tahapan akhir dalam proses pemesanan di *website* Arabic Store. Pada halaman ini menampilkan ringkasan pesanan yang akan dibeli oleh pengguna, sistem pada halaman *checkout* Arabic Store dibuat semudah mungkin untuk mempermudah tipe pembeli yang berbeda, jika sudah *login* akan langsung melihat informasi pengiriman secara otomatis, termasuk nama, nomor telepon, dan alamat lengkap yang telah diisi saat registrasi. Hal ini memudahkan pengguna karena tidak perlu mengisi ulang data. Selain itu, pengguna dapat memilih jasa pengiriman seperti JNE dan menentukan metode pembayaran, seperti QRIS, dompet digital, transfer bank, maupun COD. Namun, jika pengguna belum *login*, maka halaman *checkout* akan menampilkan form alamat yang harus diisi secara manual, dan fitur pembayaran COD akan dinonaktifkan sebagai bentuk validasi keamanan transaksi. Sistem ini memastikan proses *checkout* tetap mudah namun aman dan terverifikasi. Gambar 4.9 memperlihatkan tampilan halaman *checkout* ketika sudah *login*.



Gambar 0.9 Tampilan *Chekout User Login*

Jika pengguna belum *login*, halaman *checkout* akan menampilkan form untuk mengisi data pengiriman seperti nama, alamat, dan nomor telepon secara manual. Selain itu, metode pembayaran COD akan dinonaktifkan, sehingga hanya tersedia metode lain seperti QRIS, transfer bank, atau dompet digital. Hal ini dilakukan untuk menjaga keamanan dan memastikan identitas pengguna saat memilih pembayaran di tempat. Gambar 4.10 menampilkan tampilan *checkout* untuk pengguna yang belum *login*.



Gambar 0.10 Tampilan Halaman *Checkout* Tanpa *Login*

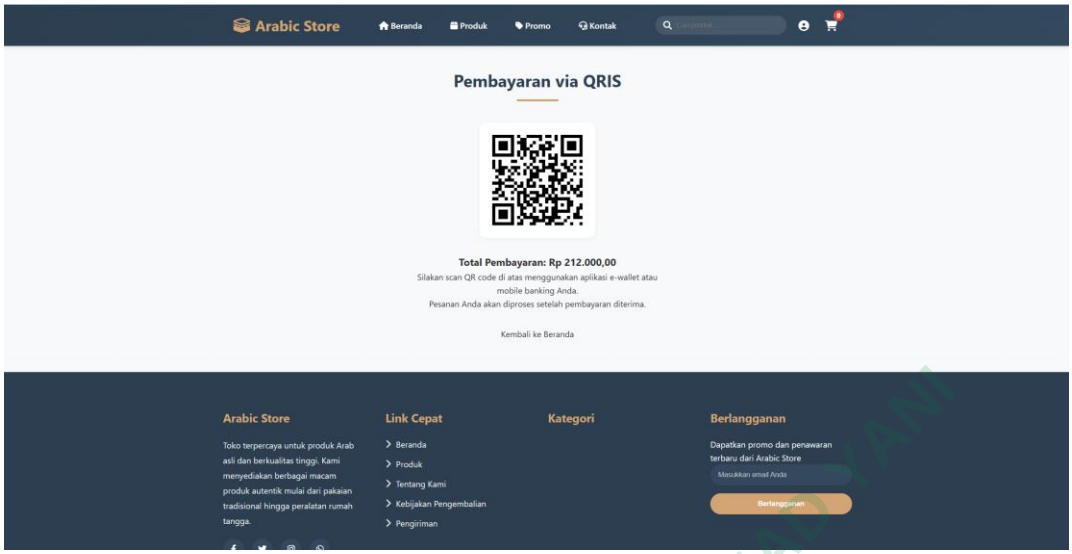
```
def checkout(request):
    product_id = request.GET.get('product_id')
    qty = request.GET.get('qty', 1)
    size = request.GET.get('size') or None
    single_product = None

    if product_id:
        product = get_object_or_404(Product, id=product_id)
        try:
            qty = int(qty)
        except ValueError:
            qty = 1
```

```
        price = product.discount_price if product.discount_price
else product.price
        total_price = price * qty
        single_product = {
            'product': product,
            'quantity': qty,
            'size': size,
            'price': price,
            'total_price': total_price,
        }
        cart_items = []
        cart_total = total_price
    else:
        cart = Cart(request)
        cart_items = list(cart)
        cart_total = cart.get_total_price()
        single_product = None
    shipping_cost = 0
    tax = 0
     t total = (single_product['total_price'] if single_product
else cart_total) + shipping_cost + tax
```

Fungsi *checkout* digunakan untuk mengelola proses akhir transaksi. Jika pengguna memilih satu produk saja, sistem akan menampilkan detail produk tersebut. Jika pengguna melalui keranjang, maka seluruh produk yang di *checkout* akan ditampilkan. Data seperti nama lengkap, alamat, metode pembayaran, dan jumlah pembayaran akan dikumpulkan di halaman ini sebelum dikirim ke *database*. Tujuannya adalah untuk memastikan seluruh data transaksi valid sebelum proses konfirmasi.

4.2.8 Konfirmasi Pembayaran

Halaman konfirmasi pembayaran merupakan bagian penting dari proses transaksi pada *website Arabic Store*. Setelah pengguna menyelesaikan pemesanan, halaman ini digunakan untuk mengunggah bukti transfer pembayaran. Pengguna diminta mengisi data seperti nomor pesanan, nama pengirim, metode pembayaran, serta mengunggah foto bukti pembayaran dalam format gambar. Informasi ini kemudian diteruskan ke *admin* untuk dilakukan proses verifikasi. Halaman ini bertujuan memastikan pembayaran dari pengguna telah diterima dan memvalidasi pesanan sebelum proses pengemasan dan pengiriman dilakukan. Dengan adanya fitur ini, proses pembayaran menjadi lebih transparan dan terstruktur. Berikut menampilkan kofirmasi pembayaran pada Gambar 4.11.



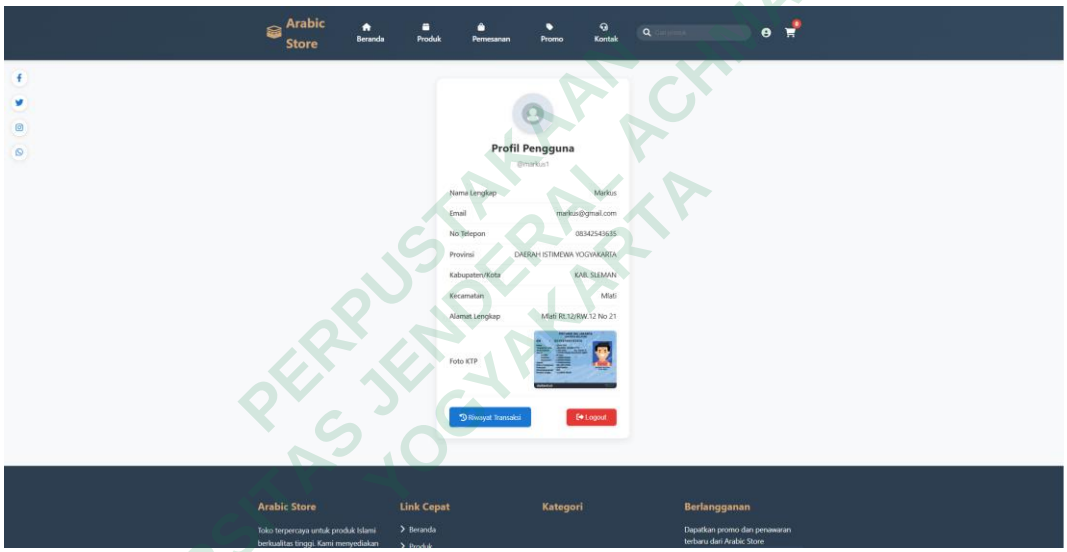
Gambar 0.11 Tampilan Konfirmasi Pembayaran

```
if request.method == 'POST':
        form = OrderCreateForm(request.POST)
        if form.is_valid():
            order = form.save(commit=False)
             # Isi first_name dan last_name dari full_name jika
ada
              full_name = getattr(order, 'full_name', None) or
request.POST.get('full_name', '')
            if full_name:
               parts = full_name.strip().split(' ', 1)
               order.first_name = parts[0]
                order.last_name = parts[1] if len(parts) > 1 else
''
            if request.user.is_authenticated:
                order.user = request.user
            if hasattr(order, 'shipping_cost'):
                                        order.shipping_cost    =
form.cleaned_data.get('shipping_cost', 0)
            payment_method = request.POST.get('payment', 'qris')
            if hasattr(order, 'payment_method'):
                order.payment_method = payment_method
            order.save()
            form.save_m2m()
```

Kode ini memproses form dari halaman *checkout*. Jika form valid, maka data seperti nama lengkap, *user login* (jika ada), metode pembayaran, total harga dan ongkir akan disimpan sebagai data *order* di *database*. Informasi ini digunakan sebagai bukti transaksi dan dapat ditampilkan kembali dalam riwayat pesanan pengguna.

4.2.9 Halaman *Profile*

Halaman profil pengguna menampilkan informasi pribadi pengguna yang telah *login* ke sistem. Pada halaman ini, pengguna dapat melihat data seperti *username* dan alamat email data Alamat dan KTP yang digunakan saat registrasi. Selain itu, tombol riwayat transaksi dan tombol untuk melakukan *logout* dari akun. Fitur ini berfungsi sebagai pusat kontrol data akun pengguna, dan memberikan pengalaman yang lebih personal saat menggunakan layanan. Sistem secara otomatis hanya memperbolehkan akses ke halaman ini bagi pengguna yang telah *login* (terautentikasi), sehingga menjaga keamanan data pengguna. Gambar 4.12 memperlihatkan halaman *profile*.



Gambar 0.12 Tampilan Halaman *Profile*

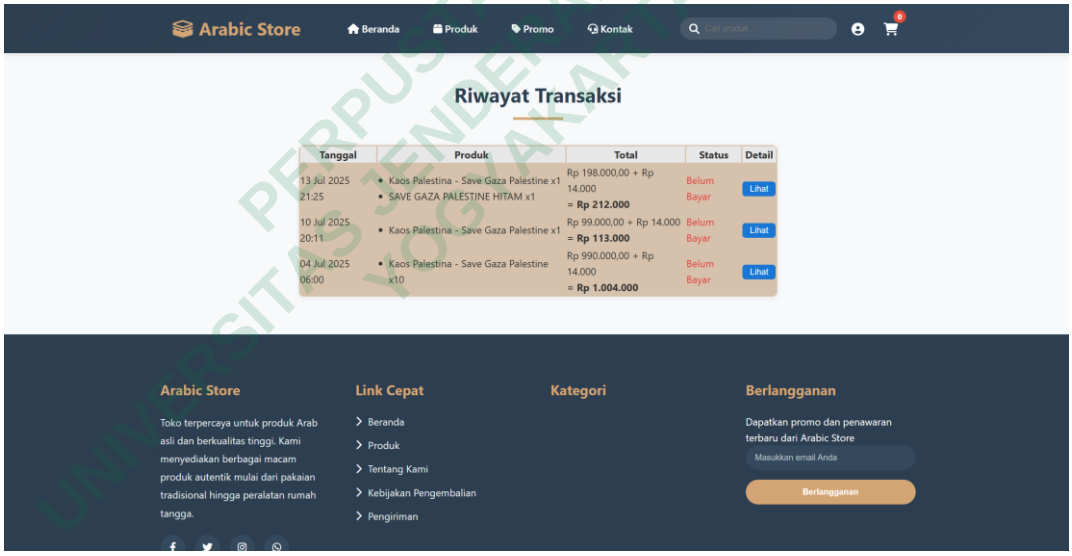
```
@login_required
def profile_view(request):
    from .models import UserProfile
    userprofile = None
    try:
        userprofile = UserProfile.objects.get(user=request.user)
    except UserProfile.DoesNotExist:
        userprofile = None
    return render(request, 'store/profile.html', {'userprofile':
userprofile})
```

Halaman ini menampilkan profil pengguna yang telah *login*. Fungsi *@login_required* memastikan hanya pengguna yang telah *login* yang bisa

mengakses halaman ini. Profil dapat dikembangkan lebih lanjut dengan fitur seperti *logout* dan melihat riwayat transaksi.

4.2.10 Halaman Riwayat Transaksi

Halaman riwayat transaksi memungkinkan pengguna untuk melihat seluruh histori pemesanan yang telah dilakukan. Pada halaman ini ditampilkan daftar pesanan pengguna, termasuk informasi nomor pesanan, tanggal pemesanan, status pembayaran, dan status pengiriman. Fitur ini memberikan transparansi dan kemudahan dalam pelacakan pesanan yang telah dilakukan oleh pengguna. Dengan adanya halaman ini, pengguna tidak perlu menghubungi *admin* untuk menanyakan status transaksi karena sudah tersaji secara otomatis di akun mereka masing-masing. Hal ini juga meningkatkan kepercayaan pengguna terhadap layanan *e-commerce Arabic Store*. Berikut menampilkan Riwayat transaksi pada Gambar 4.13.



Gambar 0.13 Tampilan Riwayat Transaksi

```
@login_required
def order_history(request):
    orders = Order.objects.filter(user=request.user).order_by('-
created').prefetch_related('items__product')
    return render(request, 'store/order_history.html', {'orders':
orders})
```

Fungsi ini menampilkan daftar pesanan yang telah dilakukan oleh pengguna, diurutkan dari yang terbaru. Dengan menambahkan *prefetch_related*,

sistem mengoptimalkan pengambilan data produk terkait untuk mempercepat load halaman. Halaman ini memudahkan pengguna untuk meninjau kembali pesanan sebelumnya.

4.2.11 *Chatbot*

Fitur *chatbot* pada sistem ini berfungsi sebagai asisten virtual untuk membantu pengguna dalam memberikan informasi umum mengenai produk, jam operasional, dan bantuan dasar lainnya. *Chatbot* ini diintegrasikan ke dalam halaman utama dan dapat diakses oleh semua pengunjung, baik yang sudah *login* maupun belum. *Chatbot* ini memanfaatkan *Natural Language Processing* (NLP) dengan menggunakan pustaka Python seperti NLTK untuk memahami maksud dari input pengguna. Proses utama dalam *chatbot* melibatkan pembersihan teks, tokenisasi, dan pencocokan pola berbasis kata kunci agar dapat memberikan jawaban yang relevan. Gambar 4.14 memperlihatkan tampilan *chatbot*.

Gambar 0.14 Tampilan *Chatbot*

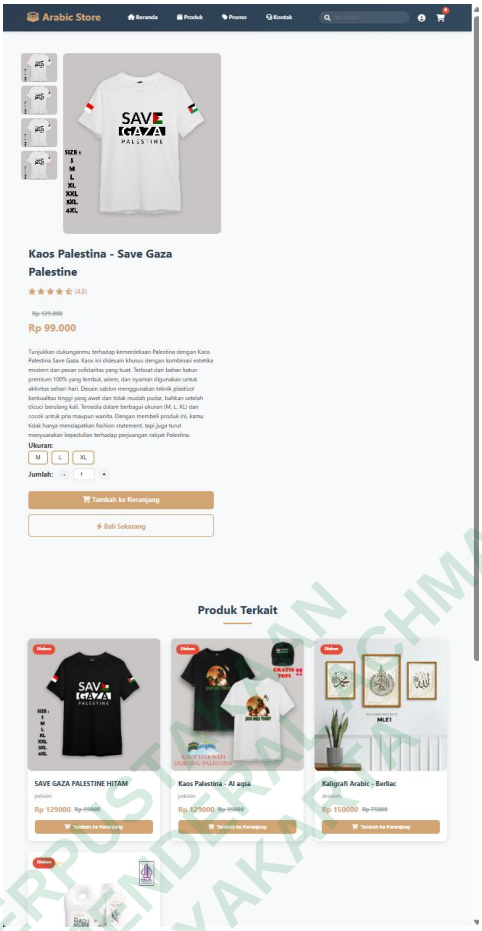
```
@method_decorator(csrf_exempt, name='dispatch')
class ChatbotAPIView(View):
    def post(self, request):
        import json
        try:
            data = json.loads(request.body)
            user_message = data.get('question', '').lower()
            if not user_message:
```

```
                    return JsonResponse({'answer': 'Silakan ketik
pertanyaan Anda.'})
            # Tokenisasi pertanyaan user
            try:
                tokens = word_tokenize(user_message)
            except Exception as e:
               return JsonResponse({'answer': f'Error tokenizing:
{str(e)}'})
            # Cari jawaban di FAQ dengan token
            for keywords, answer in FAQ_LIST:
               if any(any(k.lower() in token for token in tokens)
for k in keywords):
                    return JsonResponse({'answer': answer})
            # Default jika tidak ditemukan
           return JsonResponse({'answer': 'Maaf, pertanyaan Anda
belum tersedia di FAQ. Silakan hubungi admin untuk info lebih
lanjut.'})
        except Exception as e:
             return JsonResponse({'answer': 'Terjadi kesalahan.
Silakan coba lagi.'})
```

Kode ini adalah implementasi API *chatbot* menggunakan Django *View*, yang menerima pertanyaan dari pengguna (via HTTP POST) dan merespon dengan jawaban berdasarkan daftar FAQ yang disiapkan.

4.2.12 Sistem Rekomendasi

Pada produk terkait, menampilkan sistem rekomendasi yang dirancang untuk membantu pengguna menemukan produk relevan berdasarkan deskripsi produk yang sedang dibuka. Sistem ini menggunakan pendekatan *Content-Based Filtering*, yaitu dengan membandingkan kesamaan atribut antar produk—khususnya deskripsi produk. Agar komputer dapat memahami deskripsi dalam format teks, sistem ini memanfaatkan *Natural Language Processing* (NLP). Dalam proyek ini, digunakan metode TF-IDF (*Term Frequency-Inverse Document Frequency*) untuk mengubah teks deskripsi produk menjadi vektor numerik yang bisa dianalisis oleh komputer. Kemudian, dihitung cosine similarity antar vektor tersebut untuk mendapatkan produk yang mirip. Gamabar 4.15 menampilkan system rekomendasi produk.



Gambar 0.15 Tampilan Rekomendasi Produk

```
def get_recommendations(product, top_n=4):
    import nltk
    from nltk.corpus import stopwords
    products = list(Product.objects.filter(available=True))
    if len(products) <= 1:
        return []
    product_ids = []
    corpus = []
    for p in products:
        product_ids.append(p.id)
        # Gabungkan deskripsi, kategori, dan harga sebagai fitur
teks
          text = f"{p.name} {p.description} {p.category.name}
{p.price}"
        corpus.append(text)
    # TF-IDF vectorization
    tfidf = TfidfVectorizer(stop_words='english')
    tfidf_matrix = tfidf.fit_transform(corpus)
    # Cari index produk yang diminta
    try:
        idx = product_ids.index(product.id)
    except ValueError:
```

```
        return []
    # Hitung similarity
        cosine_sim   =   linear_kernel(tfidf_matrix[idx:idx+1],
tfidf_matrix).flatten()
    # Urutkan berdasarkan skor similarity (kecuali diri sendiri)
    similar_indices = np.argsort(cosine_sim)[::-1]
    recommendations = []
    for i in similar_indices:
        if product_ids[i] != product.id:
            recommendations.append(products[int(i)])
        if len(recommendations) >= top_n:
            break
    return recommendations
```

Fungsi *get_recommendations* (*product*, top_n=4) bertujuan memberikan rekomendasi produk yang mirip berdasarkan konten (nama, deskripsi, kategori, dan harga) dari suatu produk tertentu, menggunakan teknik TF-IDF *vectorization* dan *cosine similarity*.

4.2.13 Halaman *Dashboard Admin*

Halaman *dashboard admin* merupakan antarmuka yang hanya dapat diakses oleh pengguna yang memiliki hak akses sebagai *admin*istrator. *Dashboard* ini bertujuan untuk memberikan gambaran umum mengenai aktivitas dalam sistem, serta menyediakan menu navigasi untuk mengelola data penting seperti produk, pesanan, pelanggan, kategori, dan fitur lainnya. Tampilan *dashboard admin* dapat dilihat pada Gambar 4.16.

Gambar 0.16 Tampilan *Dashboard Admin*

```
@admin.register(Product)
class ProductAdmin(nested_admin.NestedModelAdmin):
      list_display = ['name', 'price', 'stock', 'available',
'display_image']
    list_editable = ['price', 'stock', 'available']
    prepopulated_fields = {'slug': ('name',)}
    readonly_fields = ['display_image']
    inlines = [ProductVariationTypeInline]
    def display_image(self, obj):
        if obj.image:
            return format_html('<img src="{}" width="100" />',
obj.image.url)
        return "No Image"
    display_image.short_description = 'Preview Gambar'
@admin.register(Category)
class CategoryAdmin(admin.ModelAdmin):
    list_display = ['name', 'slug']
    prepopulated_fields = {'slug': ('name',)}
@admin.register(Order)
class OrderAdmin(admin.ModelAdmin):
     list_display = ['id', 'first_name', 'last_name', 'email',
'paid']
    list_filter = ['paid', 'created', 'updated']
    search_fields = ['first_name', 'last_name', 'email']
@admin.register(OrderItem)
class OrderItemAdmin(admin.ModelAdmin):
    list_display = ['order', 'product', 'price', 'quantity']
    s earch_fields = ['order__id', 'product__name']
```

Kode ini berfungsi untuk mendaftarkan model ke dalam panel *admin* Django. Dengan mendaftarkan model *Product, Order, Customer, dan Category*, maka data-data tersebut dapat dikelola melalui *dashboard admin* secara visual tanpa harus mengakses *database* secara langsung.

4.3 BASIS DATA

Sistem ini menggunakan PostgreSQL sebagai sistem manajemen basis data relasional (RDBMS) yang handal dan *open-source*. PostgreSQL dipilih karena memiliki dukungan terhadap integritas data, relasi antar tabel, dan kompatibilitas yang tinggi dengan Django. *Database* yang digunakan bernama arabicstore, yang berisi berbagai tabel yang dihasilkan otomatis oleh *framework* Django berdasarkan model yang didefinisikan dalam aplikasi. Seluruh skema dan relasi dikelola melalui fitur ORM (*Object-Relational Mapping*) dari Django, namun penyimpanan dan eksekusi dilakukan pada PostgreSQL.

4.3.1 Struktur *Database*

Basis data pada sistem ini menggunakan PostgreSQL sebagai sistem manajemen basis data relasional. Nama *database* yang digunakan adalah arabicstore dengan schema default public. Sistem juga mengaktifkan ekstensi plpgsql untuk mendukung pemrosesan prosedural di dalam PostgreSQL.

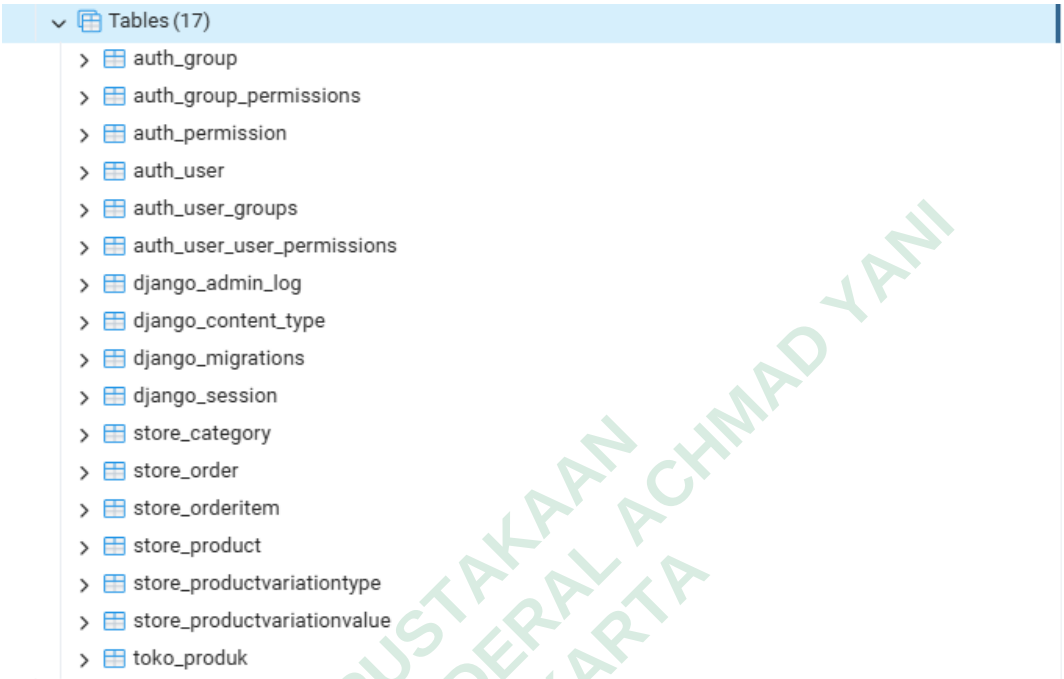
Struktur *database* ini dibentuk secara otomatis oleh Django saat perintah python manage.py *migrate* dijalankan. Django membuat struktur tabel berdasarkan model yang telah didefinisikan dalam aplikasi.

Terdapat lima tabel utama dalam sistem ini, yaitu:

1. *store_product*: Menyimpan data produk seperti nama, harga, deskripsi, gambar, dan kategori.
2. *auth_user*: Menyimpan data pengguna untuk kebutuhan *login* dan autentikasi.
3. *store_order*: Menyimpan data pesanan yang dilakukan oleh pengguna.
4. *store_cart*: Menyimpan data keranjang sementara pengguna sebelum *checkout*.

5. django_session: Tabel otomatis dari Django untuk menyimpan sesi pengguna saat mengakses *website*.

Berikut struktur lengkap tabel ditampilkan pada Gambar 4.17.



Gambar 0.17 Tampilan Tabel *Database*

4.4 IMPLEMENTASI *WEBSITE* PADA STRATEGI *DIGITAL MARKETING*

Implementasi website Arabic Store dirancang untuk mendukung strategi digital marketing dengan mengintegrasikan fitur-fitur yang mampu meningkatkan jangkauan, kenyamanan pengguna, serta konversi penjualan. Faktor penting dalam desain dan fungsionalitas website Arabic Store yang memengaruhi konversi penjualan antara lain: tampilan katalog produk yang menarik, sistem checkout yang efisien, fitur chatbot interaktif, dan sistem rekomendasi yang personal. Kemudahan navigasi dan kecepatan akses juga turut meningkatkan kemungkinan transaksi berhasil. Salah satu strategi yang digunakan adalah pemanfaatan sistem rekomendasi berbasis AI dan Chatbot berbasis NLP yang mempercepat respon terhadap pelanggan secara otomatis, sehingga meningkatkan interaksi dan kepercayaan pelanggan.

Website ini juga dilengkapi dengan katalog produk yang terstruktur, copywriting yang telah disesuaikan dengan hasil riset produk dan kompetitor, serta

fitur checkout yang efisien. Integrasi website dengan kampanye iklan digital seperti Meta Ads mempermudah proses promosi secara tertarget berdasarkan demografi dan minat pengguna yang telah dianalisis menggunakan Google Trends dan Keyword Planner.

Dengan adanya website, Arabic Store tidak lagi bergantung pada platform pihak ketiga seperti marketplace atau media sosial, namun memiliki saluran penjualan mandiri yang mampu memperkuat identitas brand dan mengurangi ketergantungan terhadap algoritma eksternal. Hal ini mendukung strategi digital marketing secara menyeluruh, mulai dari peningkatan visibilitas, efektivitas komunikasi, hingga konversi penjualan

4.4.1 Analisis Target *Audiens*

Hasil riset dari Google Trends dengan kata kunci "baju muslim", yang menunjukkan tren minat masyarakat terhadap produk fashion islami di Indonesia dalam 90 hari terakhir. Grafik bagian atas menunjukkan bahwa minat pencarian mengalami fluktuasi namun tetap stabil dengan rata-rata pencarian konsisten di angka 30–60 poin pada skala popularitas Google Trends.

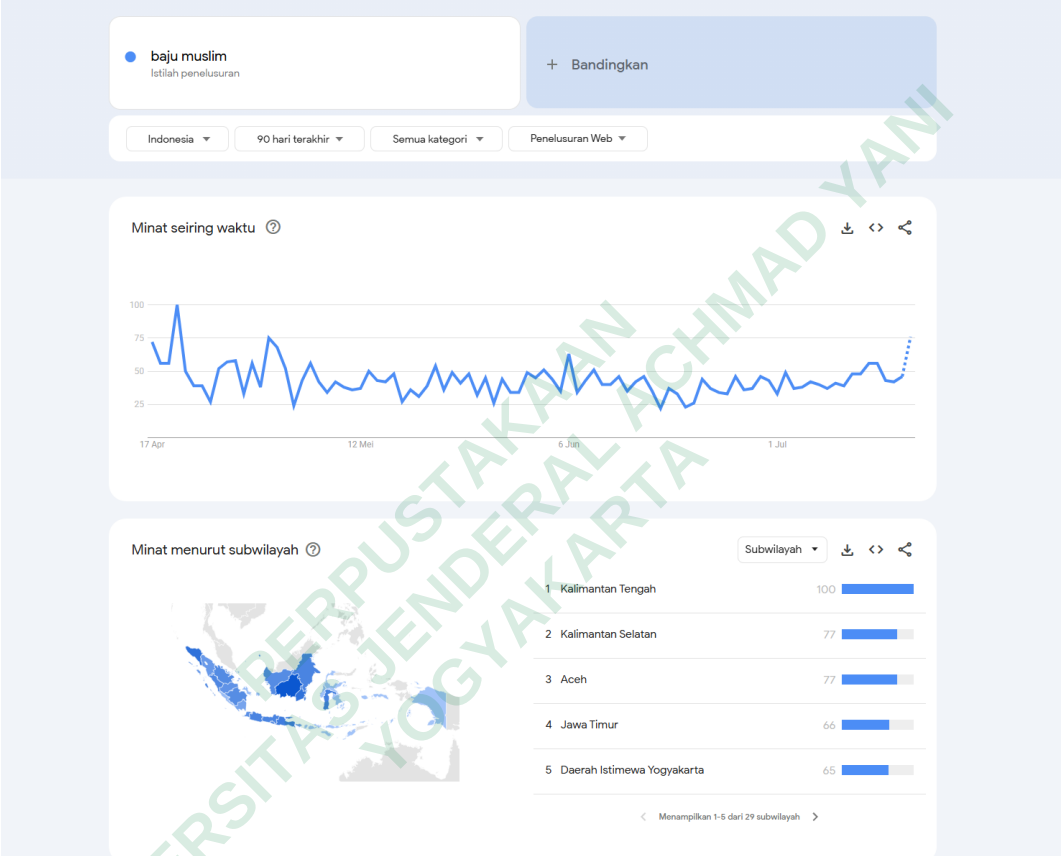
Pada bagian bawah terlihat wilayah dengan minat tertinggi terhadap pencarian "baju muslim". Lima besar daerah dengan minat paling tinggi adalah:

1. Kalimantan Tengah (100 poin – tertinggi),
2. Kalimantan Selatan (77),
3. Aceh (77),
4. Jawa Timur (66),
5. Daerah Istimewa Yogyakarta (65).

Data ini dapat dimanfaatkan oleh Arabic Store untuk menentukan target audiens utama secara geografis dalam strategi digital marketing. Misalnya, Kalimantan Tengah memilik potensi tertingi, lalu Daerah Istimewa Yogyakarta yang berada di posisi ke-5, masih cukup relevan untuk dijadikan target kampanye iklan dan distribusi produk.

Selain itu, tren minat yang stabil mengindikasikan bahwa produk baju muslim memiliki permintaan yang cukup konstan, sehingga mendukung

keberlanjutan pemasaran melalui website Arabic Store secara berkelanjutan. Data ini juga bermanfaat sebagai dasar dalam penentuan keyword iklan, penulisan copywriting, dan pengembangan konten promosi. Gambar 4.18 menampilkan riset pasar menggunakan Google Trends.



Gambar 0.18 Google Trends

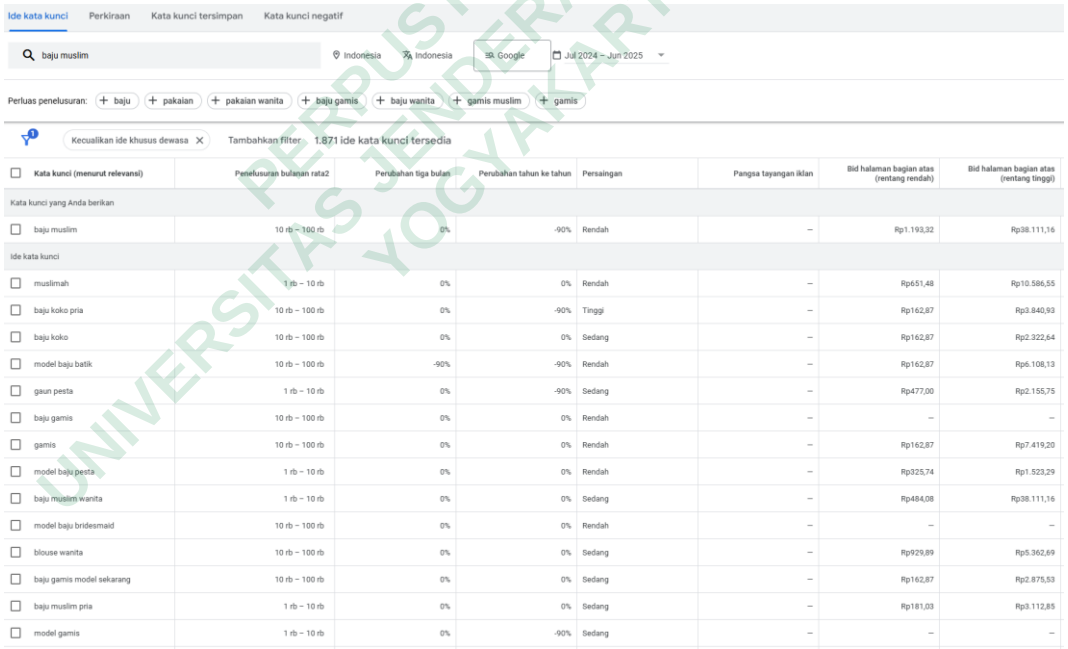
4.4.2 Riset Kata Kunci

Hasil analisis kata kunci menggunakan Google Keyword Planner dengan kata kunci utama “baju muslim”. Alat ini digunakan untuk mengidentifikasi potensi kata kunci yang relevan dalam mendukung strategi digital marketing Arabic Store, terutama untuk kebutuhan periklanan di Google Ads dan penulisan copywriting yang tertarget.

Penjelasan Hasil:

1. Kata kunci utama “baju muslim” memiliki rata-rata pencarian bulanan sebesar 10 ribu hingga 100 ribu, dengan tingkat persaingan rendah, dan biaya iklan (CPC) berkisar antara Rp 1.193,32 hingga Rp 38.111,16.
2. Ide kata kunci turunan yang juga memiliki volume pencarian tinggi dan relevansi kuat, antara lain:
    - “baju koko pria”, “baju koko” – menarget pria muslim
    - “baju gamis”, “gamis”, “muslimah” – relevan untuk target wanita muslim
    - “model baju batik”, “gaun pesta”, “blouse wanita” – menarget fashion wanita muslim yang lebih luas
3. Sebagian besar kata kunci menunjukkan persaingan rendah hingga sedang, yang berarti peluang untuk muncul di hasil pencarian atau iklan masih cukup besar dengan biaya relatif lebih efisien.

Gambar 4.19 menampilkan riset kata kunci dengan Keywoard Planner.



Ide kata kunci | Perkiraan | Kata kunci tersimpan | Kata kunci negatif

baju muslim | Indonesia | Indonesia | Google | Jul 2024 – Jun 2025

Perluas penelusuran: + baju | + pakaian | + pakaian wanita | + baju gamis | + baju wanita | + gamis muslim | + gamis

Kecualikan ide khusus dewasa | Tambahkan filter | 1.871 ide kata kunci tersedia

Kata kunci (menurut relevansi)	Penelusuran bulanan rata2	Perubahan tiga bulan	Perubahan tahun ke tahun	Persaingan	Pangsa tayangan iklan	Bid halaman bagian atas (rentang rendah)	Bid halaman bagian atas (rentang tinggi)
Kata kunci yang Anda berikan							
baju muslim	10 rb – 100 rb	0%	-90%	Rendah	–	Rp1.193,32	Rp38.111,16
Ide kata kunci							
muslimah	1 rb – 10 rb	0%	0%	Rendah	–	Rp651,48	Rp10.586,55
baju koko pria	10 rb – 100 rb	0%	-90%	Tinggi	–	Rp162,87	Rp3.840,93
baju koko	10 rb – 100 rb	0%	0%	Sedang	–	Rp162,87	Rp2.322,64
model baju batik	10 rb – 100 rb	-90%	-90%	Rendah	–	Rp162,87	Rp6.108,13
gaun pesta	1 rb – 10 rb	0%	-90%	Sedang	–	Rp477,00	Rp2.155,75
baju gamis	10 rb – 100 rb	0%	0%	Rendah	–	–	–
gamis	10 rb – 100 rb	0%	0%	Rendah	–	Rp162,87	Rp7.419,20
model baju pesta	1 rb – 10 rb	0%	0%	Rendah	–	Rp325,74	Rp1.523,29
baju muslim wanita	1 rb – 10 rb	0%	0%	Sedang	–	Rp484,08	Rp38.111,16
model baju bridesmaid	10 rb – 100 rb	0%	0%	Rendah	–	–	–
blouse wanita	10 rb – 100 rb	0%	0%	Sedang	–	Rp929,89	Rp5.362,69
baju gamis model sekarang	10 rb – 100 rb	0%	0%	Sedang	–	Rp162,87	Rp2.875,53
baju muslim pria	1 rb – 10 rb	0%	0%	Sedang	–	Rp181,03	Rp3.112,85
model gamis	1 rb – 10 rb	0%	-90%	Sedang	–	–	–

Gambar 0.19 Riset Kata Kunci

Data ini menunjukkan bahwa kata kunci seputar produk fashion muslim masih sangat potensial untuk ditargetkan dalam kampanye pemasaran. Dengan fokus pada kata kunci seperti “baju muslim”, “baju gamis”, “baju koko”, dan “muslimah”, Arabic Store dapat menyusun strategi konten dan iklan yang lebih

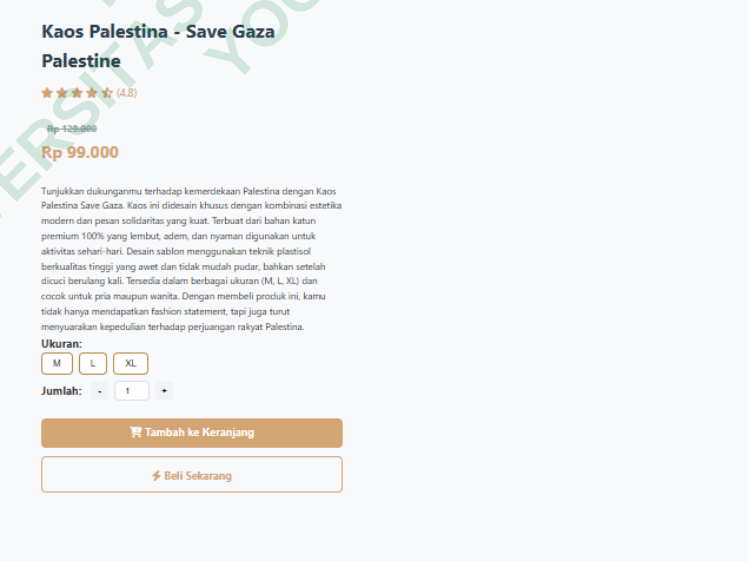
efektif untuk menjangkau pengguna yang benar-benar mencari produk serupa. Keyword Planner ini membantu Arabic Store dalam menentukan kata kunci prioritas yang akan digunakan pada:

1. Judul dan deskripsi produk
2. Copywriting iklan (Google Ads maupun Meta Ads)
3. SEO pada *website e-commerce*

   Sehingga dapat meningkatkan visibilitas di pencarian dan mendatangkan trafik tertarget.

4.4.3 Implementasi *Copywriting*

Implementasi copywriting dalam proyek ini dilakukan dengan menyusun deskripsi produk yang persuasif dan informatif pada setiap halaman detail produk di website Arabic Store. Strategi ini bertujuan untuk menarik perhatian pengunjung, membangun kepercayaan, serta mendorong mereka untuk melakukan pembelian tanpa perlu bertanya lagi ke penjual. Deskripsi produk disusun dengan memperhatikan hasil riset keyword dari Google Keyword Planner, agar tetap relevan dengan kata kunci yang sering dicari pengguna. Gambar 4.20 menampilkan implementasi strategi digital marketing pada deskripsi produk.

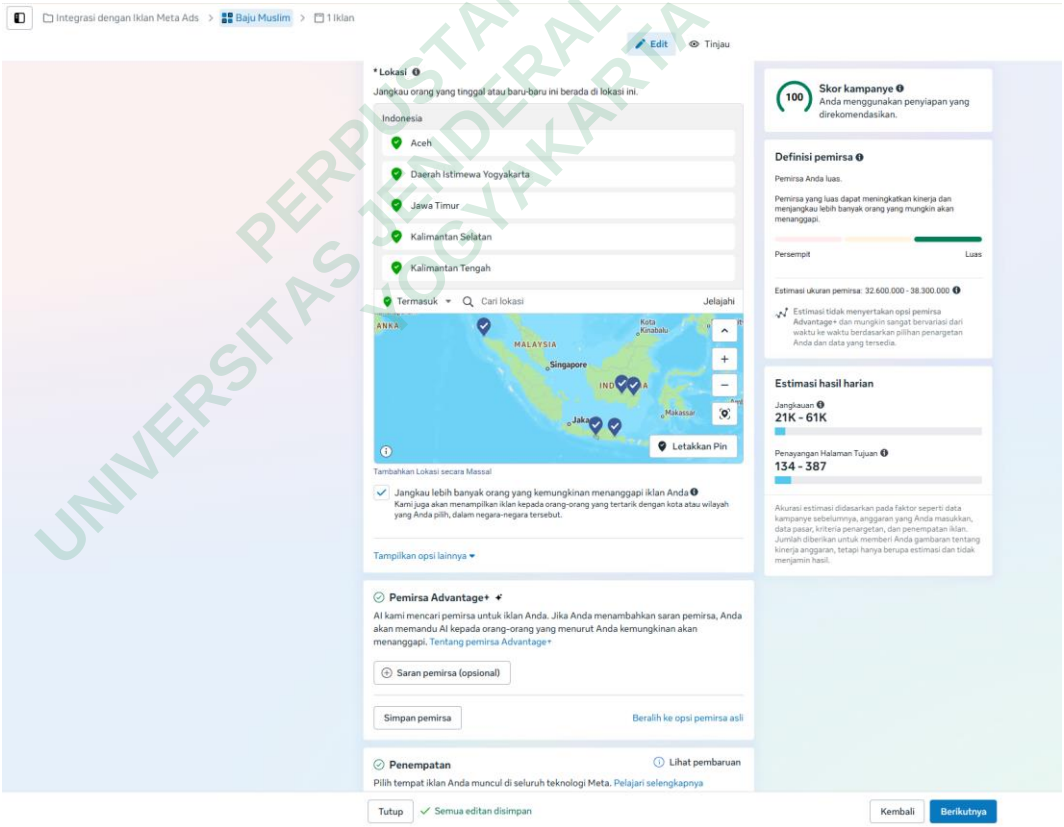


Gambar 0.20 Implementasi *Copywriting*

4.4.4 Integrasi dengan Iklan Meta Ads

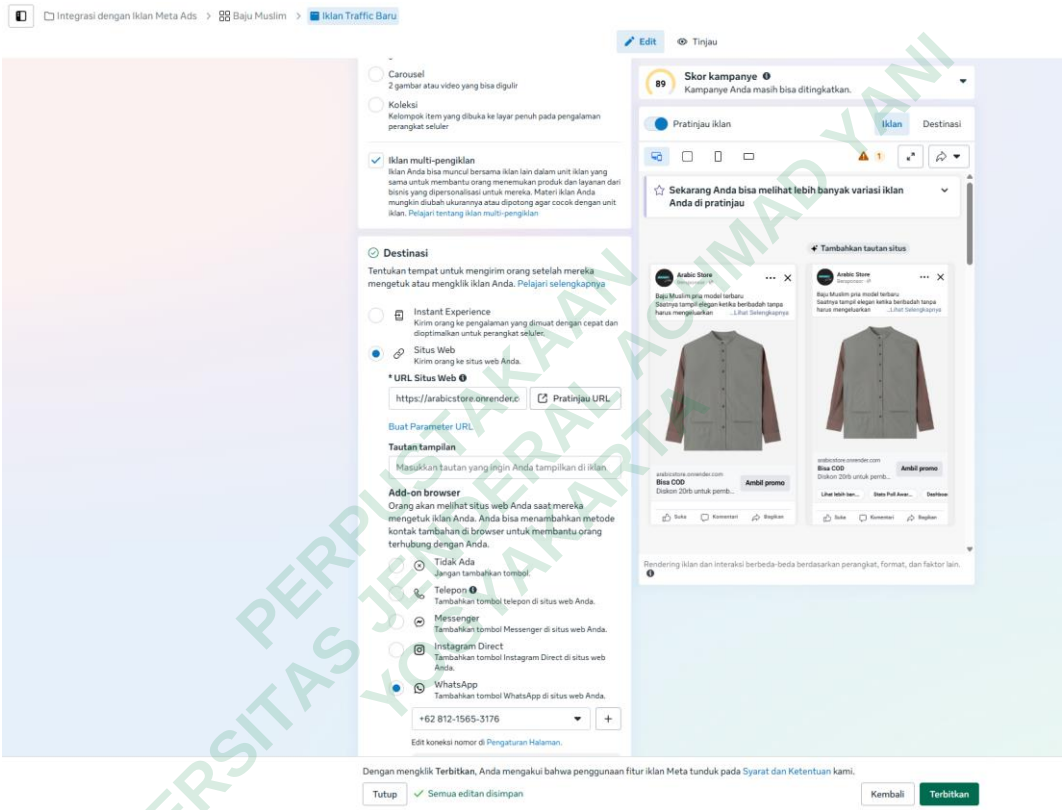
Penargetan lokasi dalam kampanye Meta Ads untuk produk "baju muslim." Lokasi yang dipilih adalah wilayah dengan minat tinggi terhadap kata kunci "baju muslim" berdasarkan Google Trends, yaitu: Aceh, Daerah Istimewa Yogyakarta, Jawa Timur, Kalimantan Selatan, dan Kalimantan Tengah. Ini menunjukkan strategi berbasis data untuk mengarahkan iklan hanya ke wilayah potensial, guna meningkatkan efektivitas anggaran iklan.

Bagian kanan menampilkan skor kampanye 100 yang berarti pengaturan kampanye sesuai rekomendasi terbaik Meta. Estimasi hasil harian iklan menunjukkan jangkauan antara 21 ribu hingga 61 ribu pengguna, dan estimasi tayangan halaman tujuan antara 134 – 387 kali per hari, menandakan potensi visibilitas yang besar terhadap audiens tertarget. Gambar 4.21 menampilkan penerapan wilayah dari Google Trends ke target pada iklan.



Gambar 0.21 Penargetan Lokasi

Pengaturan tujuan klik iklan (*destinasi*) yang diarahkan ke situs web Arabic Store (arabicstore.onrender.com). Pilihan ini penting agar calon pembeli langsung diarahkan ke website toko setelah melihat iklan. Tampilan iklan menggunakan format multi-pengiklanan, yang memungkinkan beberapa varian iklan muncul dalam satu set, meningkatkan peluang keterlibatan pengguna. Gambar 4.22 menampilkan pratinjau pembuatan iklan.



Gambar 0.22 Pratinjau Iklan

4.4.5 Fitur Pendukung Strategi *Digital Marketing*

Strategi *digital marketing* diterapkan melalui pemanfaatan pada riset Google Trends untuk menentukan kata kunci populer, kemudian diteruskan ke dalam pembuatan kampanye iklan melalui Meta Ads. Target pasar ditentukan berdasarkan lokasi, usia, dan minat yang relevan, sehingga meningkatkan efisiensi promosi.

Fitur pendukung telah diimplementasikan dalam sistem website. Fitur-fitur ini tidak hanya menunjang fungsionalitas teknis, tetapi juga dirancang untuk

mengoptimalkan pengalaman pengguna (*user experience*) dan mempercepat proses konversi dari pengunjung menjadi pelanggan. Fitur-fitur pendukung strategi digital marketing pada website Arabic Store meliputi chatbot interaktif untuk menjawab pertanyaan pelanggan secara otomatis, sistem rekomendasi produk berbasis kesamaan deskripsi dan kategori guna mendukung *upselling* dan *cross-selling*, serta fitur checkout cepat yang menyederhanakan proses pembelian. Selain itu, data profil pengguna seperti alamat dan nomor telepon yang diisi saat registrasi akan otomatis ditampilkan saat checkout, sehingga memudahkan pembelian ulang dan meningkatkan efisiensi transaksi. Seluruh fitur ini dirancang untuk meningkatkan kenyamanan pengguna, interaksi, dan tingkat konversi penjualan.

4.5 PENGUJIAN

Pengujian sistem dilakukan untuk memastikan bahwa setiap fitur pada *website* Arabic Store telah berjalan sesuai dengan fungsi yang diharapkan. Metode pengujian yang digunakan adalah *Black-box Testing*, yaitu menguji fungsionalitas sistem tanpa mengetahui struktur internal kodenya. Fokus pengujian ini adalah memastikan bahwa setiap *input* menghasilkan *output* yang sesuai.

4.5.1 Metode Pengujian

Metode pengujian yang digunakan adalah Black Box *Testing*, yaitu metode pengujian perangkat lunak yang menitikberatkan pada pengujian fungsionalitas aplikasi berdasarkan spesifikasi yang telah ditentukan. Selain itu, pengujian *User Acceptance Testing* (UAT) yang dilakukan kepada pengguna menunjukkan bahwa sistem dinilai bermanfaat. Pengujian dilakukan dengan cara memberikan input pada sistem dan mengamati hasil output yang dihasilkan untuk memastikan bahwa sistem telah berjalan sesuai dengan kebutuhan.

4.5.2 Hasil Pengujian *Black Box*

Berikut ini merupakan hasil pengujian terhadap fitur-fitur utama dalam *website* Arabic Store:

Tabel 4.1 Hasil *Black Box Testing*

No	Fitur	Langkah Uji	Input	Output	Status
1	Halaman *Login*	Pengguna mengisi form *login* dengan *username* dan *password*	*username* & *Password* valid	Pengguna diarahkan ke *dashboard*	Berhasil
2	Halaman Katalog Produk	Menampilkan daftar semua produk	Klik menu Produk	Semua produk tampil	Berhasil
3	Detail Produk	Klik pada gambar/nama produk dari katalog	Klik Produk	Halaman detail produk terbuka	Berhasil
4	Tambah ke Keranjang	Klik tombol "Tambah ke Keranjang"	Tombol diklik	Produk masuk ke keranjang	Berhasil
5	*Checkout*	Klik *checkout* setelah isi keranjang	Tombol *Checkout*	Halaman *checkout* tampil	Berhasil
6	Fitur *Chatbot*	Pengguna mengetik pertanyaan pada *chatbot*	“Promo?”	*Chatbot* memberikan jawaban yang sesuai	Berhasil
7	Rekomendasi Produk	Sistem menampilkan produk rekomendasi berdasarkan produk yang dibuka	Buka produk tertentu	Produk serupa ditampilkan di bagian bawah	Berhasil
8	Halaman *Admin*	*Login admin* lalu akses	*Username* &	*Admin dashboard* tampil	Berhasil

		halaman *admin*	*password admin*		

4.5.3 Hasil Pengujian *User Acceptance Testing* (UAT)

User Acceptance Testing dilakukan dengan menyebarkan kuesioner kepada 5 responden guna mengevaluasi kelayakan aplikasi Arabic Store berdasarkan metode USE (*Usefulness, Satisfaction, Ease of Use, Ease of Learning*). Setiap pernyataan dinilai menggunakan skala Likert 1–5. Hasil pengujian ditampilkan pada table berikut:

Tabel 4.2 Penilaian *Usefulness*

Responden	P1	P2	P3	P4	Total
Ilham Mulia	5	5	4	5	19
Khabil Putra	5	4	5	4	18
Aji Nur Cahyo	5	5	4	5	19
Faihal Rasyid	4	4	5	5	18
Devi Ambar K	5	5	5	5	20
Total Skor					**94**
Skor Ideal					**100**
Presentase					**94%**

Tabel 4.2 menjelasnya bahwa hasil dari perhitungan *Usefulness* adalah 94%. Setiap responden memberikan nilai berdasarkan skala Likert 1–5, dengan 5 berarti “sangat setuju”.

Tabel 4.3 Penilaian *Satisfaction*

Responden	P1	P2	P3	P4	Total
Ilham Mulia	4	4	5	4	17
Khabil Putra	5	5	4	5	19
Aji Nur Cahyo	5	4	4	5	18
Faihal Rasyid	5	5	5	5	20
Devi Ambar K	4	4	5	5	18

Total Skor	**92**
Skor Ideal	**100**
Presentase	**92%**

Tabel 4.3 menjelasnya bahwa hasil dari perhitungan *Satisfaction* adalah 92%. Setiap responden memberikan nilai berdasarkan skala Likert 1–5, dengan 5 berarti “sangat setuju”.

Tabel 4.4 *Ease of Use*

Responden	P1	P2	P3	P4	Total
Ilham Mulia	4	4	5	4	17
Khabil Putra	4	4	4	4	16
Aji Nur Cahyo	5	5	4	5	19
Faihal Rasyid	4	5	5	4	18
Devi Ambar K	5	5	5	5	20
Total Skor					**90**
Skor Ideal					**100**
Presentase					**90%**

Tabel 4.4 menjelasnya bahwa hasil dari perhitungan *Ease of Use* adalah 90%. Setiap responden memberikan nilai berdasarkan skala Likert 1–5, dengan 5 berarti “sangat setuju”.

Tabel 4.5 *Ease of Learning*

Responden	P1	P2	P3	P4	Total
Ilham Mulia	4	4	5	4	17
Khabil Putra	5	4	4	5	18
Aji Nur Cahyo	5	5	5	4	19
Faihal Rasyid	4	5	4	5	18
Devi Ambar K	5	5	4	4	18
Total Skor					**90**
Skor Ideal					**100**
Presentase					**90%**

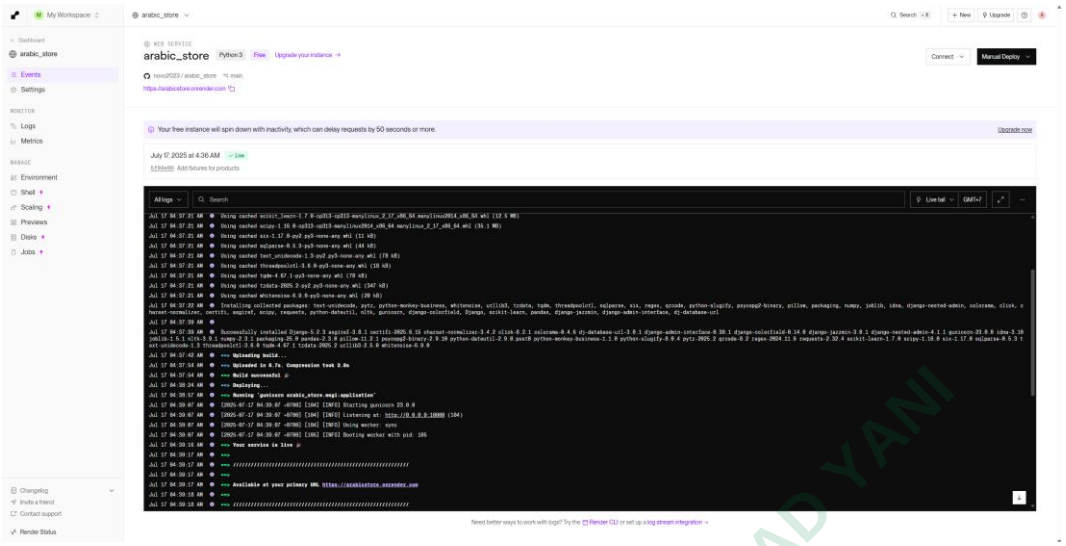
Tabel 4.5 menjelasnya bahwa hasil dari perhitungan *Ease of Learning* adalah 90%. Setiap responden memberikan nilai berdasarkan skala Likert 1–5, dengan 5 berarti “sangat setuju”.

4.6 *DEPLOYMENT*

Deployment merupakan tahap akhir dalam proses pengembangan aplikasi, yaitu proses menempatkan sistem yang telah selesai dibuat ke lingkungan server agar dapat diakses dan digunakan oleh pengguna secara online. Pada proyek ini, aplikasi *Arabic Store* dideploy menggunakan layanan cloud *hosting* Render.com yang menyediakan *platform* gratis dan mudah untuk aplikasi berbasis Django.

Langkah-langkah *deployment* dimulai dengan melakukan push kode proyek ke GitHub, lalu menghubungkan *repository* tersebut ke akun Render.com. Setelah itu dilakukan konfigurasi seperti menentukan *runtime environment, build command, dan start command* menggunakan *Gunicorn* untuk menjalankan server WSGI. Selain itu, dilakukan juga pengaturan variabel *environment* seperti DEBUG=False, ALLOWED_HOSTS, dan *DATABASE*_URL.

Proses *deployment* ini bertujuan agar aplikasi dapat diakses secara publik melalui alamat URL yang diberikan oleh Render, yaitu: https://arabicstore.onrender.com. Dengan *deployment* ini, aplikasi *Arabic Store* tidak hanya dapat diakses di lingkungan lokal developer, tetapi juga bisa digunakan oleh pengguna secara luas melalui jaringan internet. Gambar 4.23 menampilkan proses *deployment.*



Gambar 0.23 *Deployment*

4.7 PEMBAHASAN

Website Arabic Store sudah dikembangkan dari hasil implementasi sistem yang telah dibangun, mulai dari fungsionalitas, tampilan antarmuka, hingga hasil pengujian. Aplikasi *Arabic Store* telah berhasil dikembangkan sebagai *platform e-commerce* yang memfasilitasi pengguna dalam mencari, memesan, dan melakukan pembayaran produk Muslim seperti kaos Palestina dan produk Islami lainnya.

Setiap fitur yang dirancang seperti registrasi pengguna dengan input alamat dan unggah KTP, sistem *login*, keranjang belanja, *checkout* otomatis dengan pengambilan data profil, sistem rekomendasi produk, hingga fitur *chatbot* telah berhasil diimplementasikan dan diuji berjalan dengan baik. Fitur *dropdown* alamat dinamis menggunakan API dari Binderbyte juga mendukung proses input wilayah secara akurat.

Website Arabic Store mengimplementasikan chatbot berbasis NLP untuk merespons pertanyaan pelanggan seperti promo, status pesanan, dan stok produk. Sistem rekomendasi dibuat berdasarkan kemiripan deskripsi produk menggunakan TF-IDF dan cosine similarity, yang mampu menampilkan produk relevan bagi pengguna.

Hasil pengujian menggunakan *Black Box Testing* menunjukkan bahwa seluruh fitur inti dapat berjalan sesuai dengan skenario yang dirancang. Selain itu,

pengujian *User Acceptance Test* (UAT) yang dilakukan kepada lima pengguna menyatakan bahwa sistem ini mudah digunakan, memberikan kepuasan, dan memiliki fungsi yang bermanfaat, dengan persentase skor UAT di atas 90% pada kategori *Usefulness* dan *Satisfaction*.

Secara keseluruhan, sistem yang dikembangkan telah memenuhi kebutuhan pengguna dalam memesan produk secara online serta memberikan pengalaman pengguna yang baik. Namun, masih terdapat potensi pengembangan seperti penambahan fitur pembayaran otomatis, pelacakan pengiriman, dan sistem *admin* yang lebih kompleks.