BAB 4
HASIL PENELITIAN

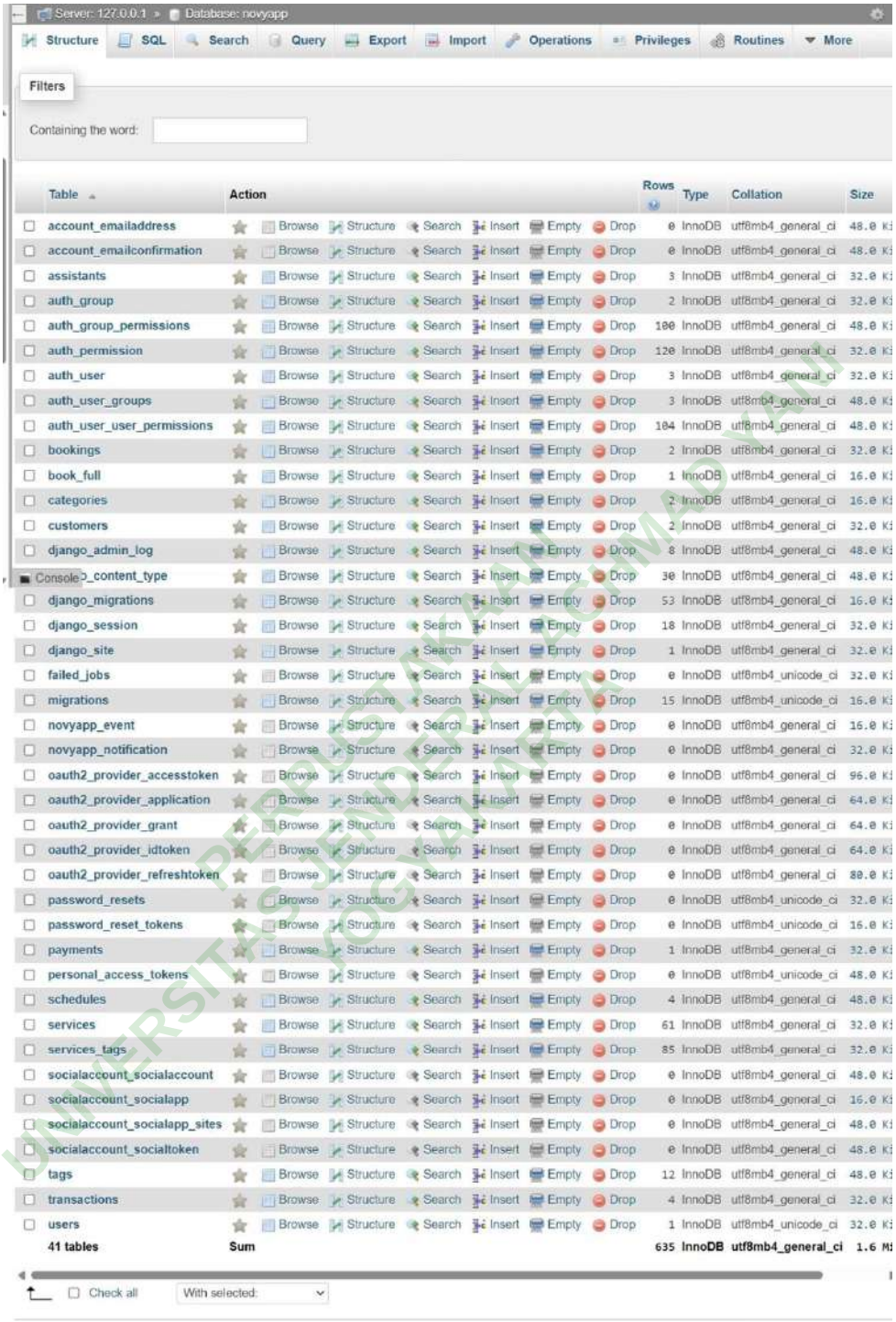
4.1 RINGKASAN HASIL PENELITIAN

Aplikasi pemesanan layanan tata rias Novy salon dan rekapan transaksi dikembangkan dengan metode pengembangan Agile dan bahasa pemrograman Python yaitu *framework* Django serta pengelolaan *database* menggunakan MySQL. Aplikasi ini merupakan implementasi proses bisnis manual menjadi digital. Aplikasi ini mampu menjadi tempat penyimpanan data transaksi dan pengelolaan reservasi jadwal tata rias yang memungkinkan lebih terstruktur dan menghindari kehilangan data atau rusak. Admin dapat lebih mudah memantau performa setiap asisten salon dan melakukan perhitungan total pendapatan berdasarkan data yang dimasukkan setiap harinya.

Asisten salon dapat melakukan pencatatan data transaksi setiap harinya, melakukan *filtering* penghasilan perbulan atau rentang waktu tertentu. Pelanggan dapat melihat jadwal yang tersedia di Novy salon tanpa perlu mendatangi salon dan bertanya terlebih dahulu. Semua informasi layanan salon, alamat salon, jadwal operasi salon dapat diakses dengan mudah. Pelanggan juga dapat melakukan reservasi jadwal yang diinginkan.

4.2 IMPLEMENTASI BASIS DATA

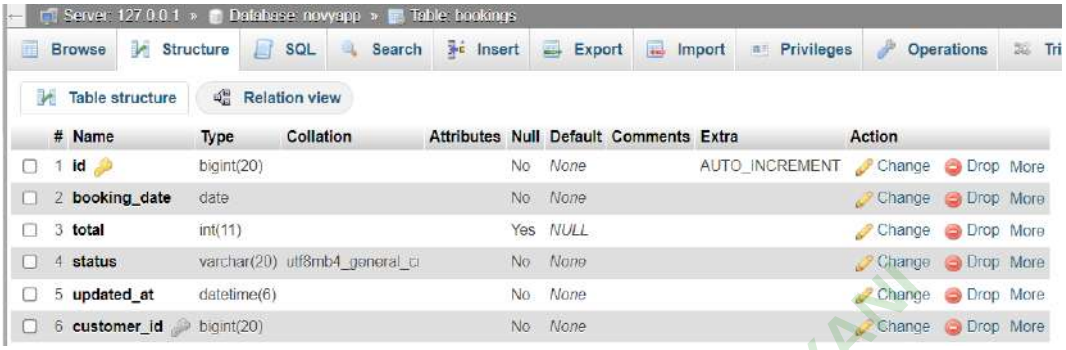
Dalam mengembangkan aplikasi pemesanan layanan ini menggunakan *database* MySQL sebagai penghubung antara kode program dan tampilan pengguna. MySQL mendukung berbagai jenis tipe data yang akan dikembangkan dalam pengelolaan *database* pada server meliputi *string, varchar, float, decimal, date, text* dan sebagainya. Basis data pada aplikasi ini terdiri dari Tabel Categories, Tags, Services, Bookings, Shedules, Customers, Transactions, Assistants, Notification. Daftar Tabel basis data yang telah diimplementasikan terlihat pada Gambar 4.1.



Gambar 4.1 Daftar tabel basis data

Berdasarkan daftar tabel tersebut berikut dilampirkan salah satu struktur tabel yaitu tabel *booking*. Tabel *booking* adalah tabel yang digunakan untuk menampung data *booking* pelanggan. Seperti terlihat pada Gambar 4.2 tabel ini

memiliki id, tanggal *booking* (*booking_date*), total, status, *updated_at* dan *customer_id* (relasi dengan tabel *customer*).



Gambar 4.2 Tabel *booking*

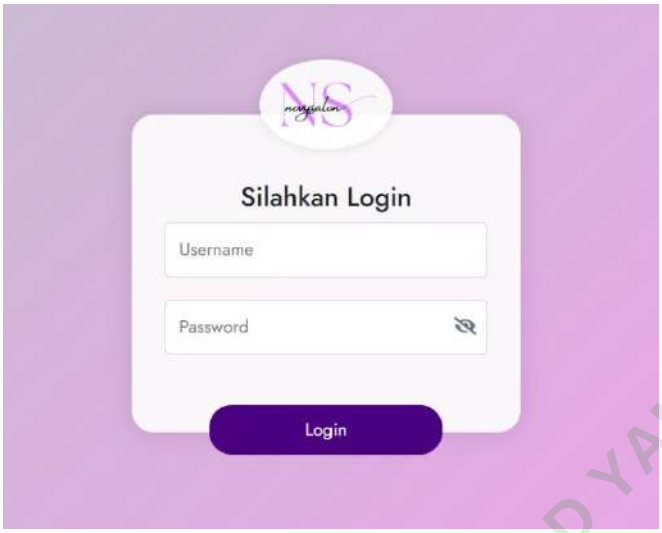
4.3 IMPLEMENTASI DESAIN ANTARMUKA

Agar tampilan aplikasi lebih menarik dan mudah dipahami oleh pengguna maka seluruh rancangan desain antarmuka yang sebelumnnya dijabarkan diimplimentasikan menjadi tampilan aplikasi. Penerapan desain antarmuka merupakan langkah agar pengelolaan data lebih mudah dan detail. Desain antarmuka diterjemahkan menjadi tampilan yang ramah pengguna serta fitur berfungsi dengan baik.

4.3.1 Halaman Login

Halaman *login* yaitu halaman autentikasi admin serta asisten salon sebelum diberikan akses ke dalam fitur aplikasi. Dalam halaman ini, admin dan asisten harus mengisikan kredensial seperti *username* serta *password* yang terdaftar di *database.* Setelah memasukkan informasi yang benar, pengguna dapat masuk (*login*) dan mengakses aplikasi.

Apabila pengguna salah saat memasukkan *username* dan *password* maka tidak dapat melakukan *login* dan akan menampilkan halaman *login* kembali. Halaman *login* terlihat di Gambar 4.3.



Gambar 4.3 Halaman login

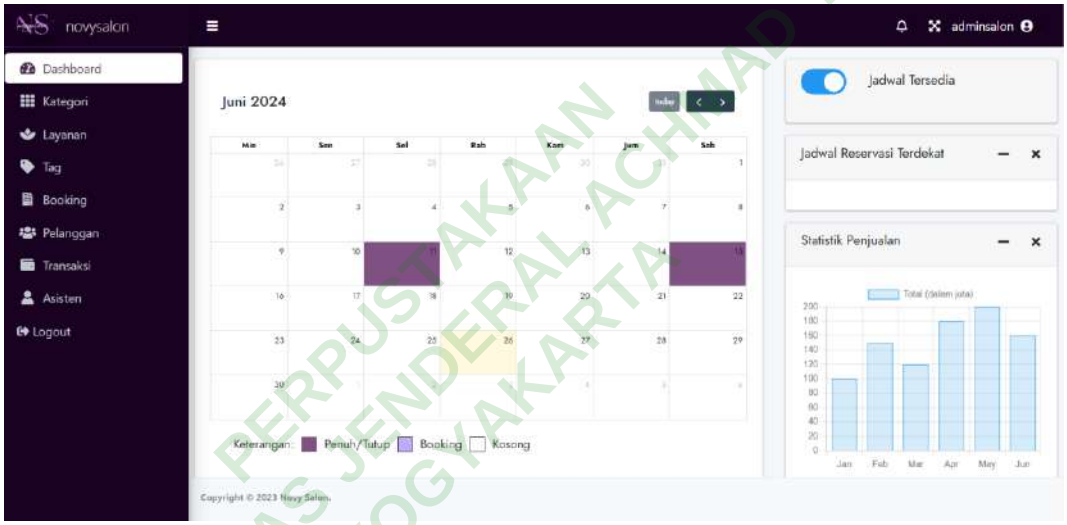
Berikut merupakan kode untuk menampilkan halaman *login* agar dapat mengakses aplikasi.

```
class CustomLoginView(LoginView):
    authentication_form = LoginForm
    template_name = 'admin/login.html'

    def get_success_url(self):
        user = self.request.user
        if user.groups.filter(name='Admin').exists():
            return reverse_lazy('dashboard')
        elif user.groups.filter(name='Asisten').exists():
            return reverse_lazy('transaksi')
        else:
            return reverse_lazy('index')

    def form_invalid(self, form):
        messages.error(self.request,   "Username   atau   password
salah!")
        return super().form_invalid(form)
```

Pada baris kode diatas dijelaskan bahwa terdapat 2 pembagian hak *login* yaitu admin dan asisten. Saat admin berhasil *login* maka akan menampilkan halaman *dashboard* dan jika asisten berhasil *login* maka akan ditampilkan halaman transaksi. Pada saat proses input *username* dan *password*, jika salah satu input atau keduanya salah maka akan ditampilkan pesan kesalahan dan tidak dapat mengakses halaman sistem.

4.3.2 Halaman Dashboard

Halaman dashboard bertujuan untuk mengelola jadwal reservasi dimana tampil sebuah kalender yang jika tanggal tertentu diklik maka akan menampilkan detail reservasi pada tanggal tersebut. Halaman ini menjadi halaman pertama yang tampil saat admin melakukan login pada aplikasi. Dashboard juga bertujuan untuk mengubah tanggal reservasi pada tanggal tertentu yang akan otomatis tampil pada halaman pelanggan. Seperti terlihat dalam Gambar 4.4 merupakan tampilan halaman dashboard yang juga menampilkan halaman lainnya pada *sidebar* menu.



Gambar 4.4 Halaman dashboard

Berikut ini merupakan potongan kode program untuk menampilkan halaman dashboard.

```
@login_required
@user_passes_test(is_admin)
def dashboard(request):
    google_calendar = get_google_calendar_events()

    context = {
        'google_calendar': google_calendar,
        'google_api_calendar': settings.GOOGLE_CALENDAR_API,
        'google_calendar_id': settings.GOOGLE_CALENDAR_ID,
    }
    return render(request, 'dashboard.html', context)
```

Baris kode diatas digunakan untuk menampilkan halaman template pada dashboard dan juga untuk melakukan pemanggilan terhadap Google Calendar API dan Google Calendar ID dari file settings pada aplikasi.

4.3.3 Halaman Kategori

Halaman kategori menampilkan Tabel kategori layanan yang ada di Novy salon dengan beberapa list data yang disimpan.

Gambar 4.5 Halaman kategori

Seperti terlihat pada Gambar 4.5 halaman kategori memiliki tombol tambah data, dan aksi yang ada dalam Tabel list data yaitu edit data kategori dan hapus data kategori. Dalam Tabel terlihat bahwa terdapat beberapa kolom yaitu nomor, nama kategori, deskripsi, status *booking* yaitu apakah dapat di *booking* atau tidak, dan juga tombol pencarian kategori berdasarkan nama. Berikut kode pada halaman kategori agar dapat melakukan tambah data kategori.

```
@user_passes_test(is_admin)
def add_category(request):
    if request.method == 'POST':
        form = CategoryForm(request.POST)
        if form.is_valid():
            is_book_value = int(request.POST.get('is_book', 0))
            form.instance.is_book = is_book_value
            form.save()
            messages.success(request,      "Kategori      berhasil
ditambahkan!")
            return redirect('kategori')
    else:
        form = CategoryForm()
    return render(request, 'kategori/index.html', {'form': form})
```

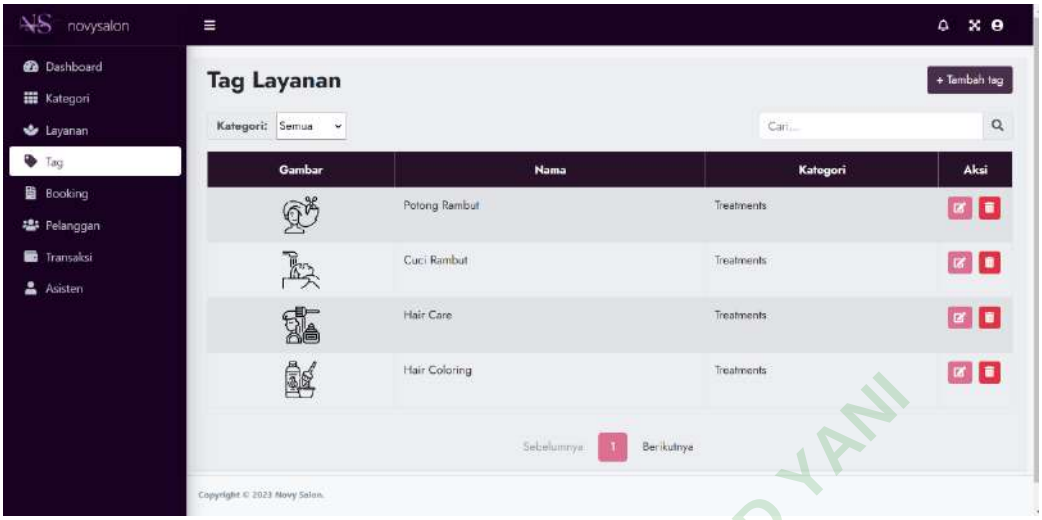
Berikut kode pada halaman kategori agar dapat melakukan edit data kategori.

```
@user_passes_test(is_admin)
def category_edit(request, pk):
    category = get_object_or_404(Category, pk=pk)

    if request.method == 'POST':
        form = CategoryForm(request.POST, instance=category)
        if form.is_valid():
            form.save()
            messages.success(request,      "Kategori      berhasil
diperbaharui!")
            return redirect('kategori')
        else:
            messages.error(request,      "Gagal      memperbaharui
kategori, silahkan periksa kembali!")
    else:
        form = CategoryForm(instance=category)
        return   render(request,   'kategori/edit.html',   {'form':
form})
```

Pada baris kode diatas dijelaskan bahwa bahwa halaman kategori hanya dapat diakses oleh admin. Proses tambah, edit dan hapus data akan menyimpan perubahan kedalam database. Saat proses berhasil ataupun gagal akan menampilkan pesan. Formulir untuk mengatur kolom pada kategori diambil dari CategoryForm.

4.3.4 Halaman Tag

Halaman tag menampilkan tabel tag layanan yang ada di Novy salon dengan beberapa list data yang disimpan. Seperti terlihat pada Gambar 4.6 halaman tag memiliki tombol tambah data, dan aksi yang ada dalam Tabel list data yaitu edit data dan hapus data tag.



Gambar 4.6 Halaman tag

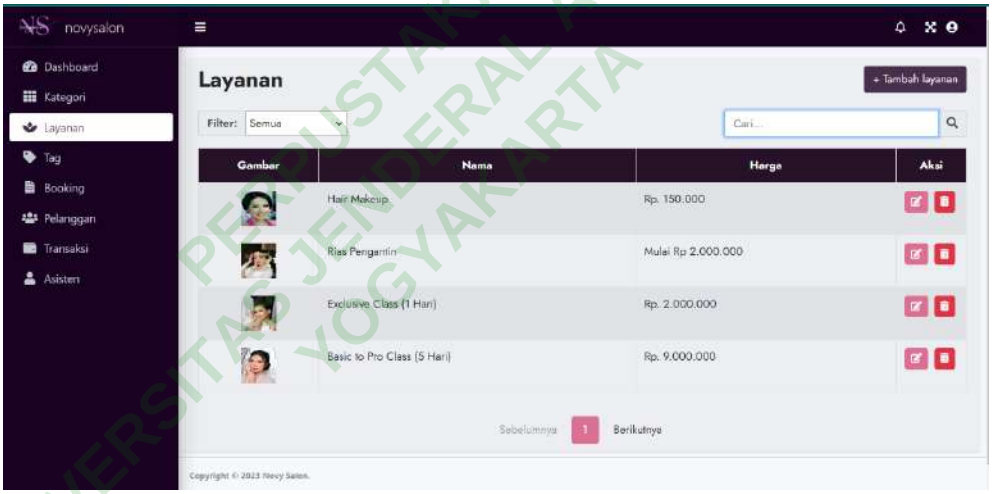
Dalam tabel terlihat bahwa terdapat beberapa kolom yaitu gambar, nama tag, kategori, tombol pencarian tag berdasarkan nama dan *filter* kategori tag. Berikut salah satu fungsi kode program untuk menampilkan halaman tambah tag dan proses simpan data.

```
@user_passes_test(is_admin)
def add_tag(request):
    if request.method == 'POST':
        category_id = request.POST.get('category', None)
        category = Category.objects.get(pk=category_id)
        name = request.POST['name']
        description = request.POST['description']

        if 'image' in request.FILES:
            image = request.FILES['image']
            image_path = os.path.join(settings.MEDIA_ROOT,
'images', 'tag', image.name)
            with open(image_path, 'wb+') as destination:
                for chunk in image.chunks():
                    destination.write(chunk)
            image_url = os.path.join('images', 'tag', image.name)
        else:
            image_url = None

        Tag.objects.create(category=category, name=name,
image=image_url, description=description)
        messages.success(request, "Tag berhasil ditambahkan!")
        return redirect('tag')

    else:
        return render(request, 'tag/index.html')
```

Pada baris kode diatas dijelaskan bahwa halaman tag hanya dapat diakses oleh admin. Terdapat kolom yang perlu diisi yaitu category_id, nama, deksripsi dan gambar. Jika pengisian kolom benar dilakukan maka akan menampilkan pesan berhasil dan kembali ke halaman tag.

4.3.5 Halaman Layanan

Halaman layanan menampilkan tabel layanan yang ada di Novy salon dengan beberapa list data yang disimpan. Seperti terlihat pada Gambar 4.7 halaman layanan memiliki tombol tambah data, dan aksi yang ada dalam Tabel list data yaitu edit data layanan dan hapus data layanan. Dalam Tabel terlihat bahwa terdapat beberapa kolom yaitu gambar, nama layanan, harga, tombol pencarian layanan berdasarkan nama dan *filter by tag*.



Gambar 4.7 Halaman layanan

Berikut salah satu fungsi kode program yang digunakan untuk menampilkan data layanan pada halaman layanan.

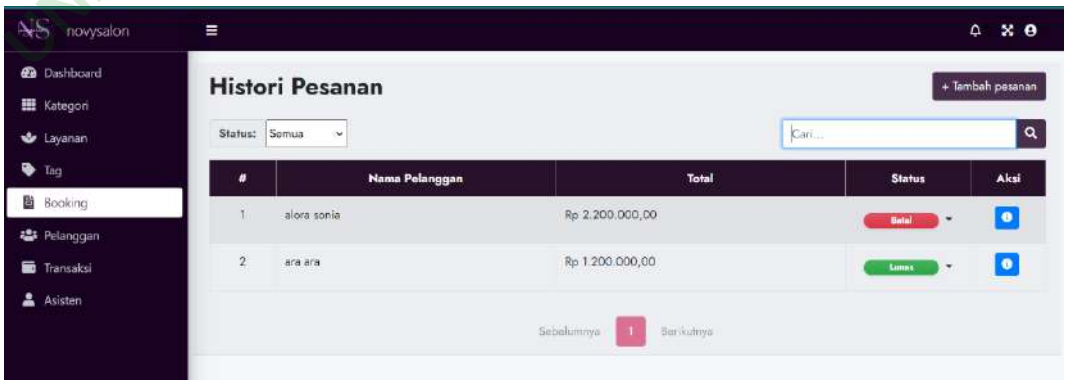
```
@login_required
@user_passes_test(is_admin)

def service(request):
    tags = Tag.objects.all()
    query = request.GET.get('search')
    selected_tag_id = request.GET.get('tag')
    services = Service.objects.all()
```

```
    if query:
        services = services.filter(
            Q(name__icontains=query) |
            Q(description__icontains=query)
        )

    if selected_tag_id:
        services = services.filter(tags__id=selected_tag_id)

    paginator = Paginator(services, 4)
    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)

    for service in page_obj:
        service.image = settings.MEDIA_URL + service.image

    return render(request, 'layanan/index.html', {'page_obj':
page_obj, 'tags': tags, 'selected_tag_id': selected_tag_id})
```

Pada baris kode diatas dijelaskan bahwa halaman layanan hanya dapat diakses oleh admin. Halaman layanan akan menampilkan daftar tag, menu pencarian, berdasarkan nama layanan dan deskripsi layanan serta gambar layanan.

4.3.6 Halaman *Booking*

Halaman *booking* merupakan halaman yang menampilkan data riwayat *booking* pelanggan. Seperti terlihat pada Gambar 4.8 halaman *booking* memiliki tombol tambah data, tombol aksi lihat detail *booking*, tombol pencarian data *booking* dan *filter booking by* status *booking*. Pada tabel *booking* terdapat beberapa kolom yaitu nomor, nama pelanggan, total, jumlah bayar, status dan aksi.

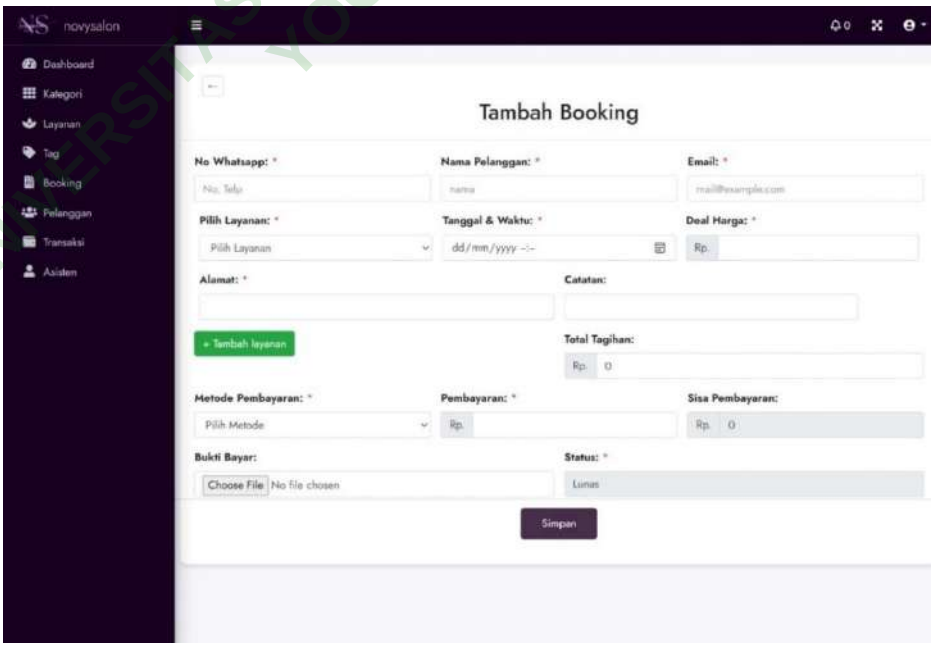


Gambar 4.8 Halaman *booking*

Berikut salah satu fungsi kode program untuk menampilkan halaman *booking*.

```
@login_required
@user_passes_test(is_admin)
def booking(request):
    bookings = Booking.objects.all()
    customer = Customer.objects.all()
    status_filter = request.GET.get('status')

    if status_filter:
        bookings = Booking.objects.filter(status=status_filter)

    paginator = Paginator(bookings, 4)
    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)
return render(request, 'booking/index.html',
    {'bookings': bookings, 'page_obj':page_obj})
```

Pada baris kode diatas dapat dijelaskan bahwa halaman *booking* hanya dapat diakses oleh admin. Pada halaman *booking* terdapat kolom *filter by* status *booking* yang digunakan untuk menampilkan *booking* berdasarkan status tertentu. Proses pengelolaan data *booking* seperti menambah data *booking* dicermati pada Gambar 4.9. Proses tambah *booking* memerlukan data *customer* seperti nama, nomor Whatsapp, alamat email serta kolom lainnya yang akan digunakan dalam pendataan.



Gambar 4.9 Halaman tambah booking

Berikut merupakan potongan kode program untuk menampilkan halaman tambah booking.

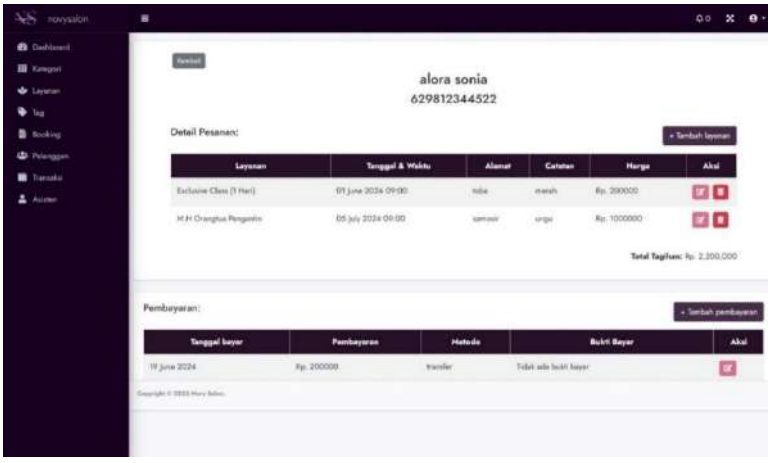
```
def store(request):
    current_date = timezone.now().date()
    if request.method == 'POST':
        customer_name = request.POST.get('customer_name')
        service_ids = request.POST.getlist('services[]')
        datetime_strs = request.POST.getlist('datetime[]')
        addresses = request.POST.getlist('address[]')

        service = get_calendar_service()

        events_created = []

        services_by_date = {}
        for service_id, datetime_str, address in zip(service_ids,
datetime_strs, addresses):
            try:
                datetime_obj =
datetime.fromisoformat(datetime_str)
            except ValueError:
                return    HttpResponseBadRequest(f"Invalid    ISO
format datetime string: {datetime_str}")

            date_key = datetime_obj.date()
            if date_key not in services_by_date:
                services_by_date[date_key]   =   {'services':   [],
'datetime': datetime_obj, 'address': address}

            try:
                service_obj = Service.objects.get(id=service_id)

services_by_date[date_key]['services'].append(service_obj.name)
            except Service.DoesNotExist:
                return  HttpResponseBadRequest(f"Service  with  id
{service_id} does not exist")

        for date_key, data in services_by_date.items():
            service_names = ', '.join(data['services'])
            datetime_obj = data['datetime']
            address = data['address']

            event = {
                'summary': f'Booking {customer_name}',
                'description': service_names,
                'location': address,
                'start': {
                    'dateTime':      datetime_obj.strftime('%Y-%m-
%dT%H:%M:%S'),
                    'timeZone': 'Asia/Jakarta',
                },
                'end': {
```

```
                    'dateTime': (datetime_obj +
timedelta(hours=1)).strftime('%Y-%m-%dT%H:%M:%S'),
                    'timeZone': 'Asia/Jakarta',
                },
            }

            created_event =
service.events().insert(calendarId=settings.GOOGLE_CALENDAR_ID,
body=event).execute()
            events_created.append(created_event)

        # Simpan data pelanggan
        customer_phone = request.POST.get('customer_phone')
        customer_email = request.POST.get('customer_email')

        existing_customer =
Customer.objects.filter(phone=customer_phone).first()
        if existing_customer:
            customer = existing_customer
            customer.name = customer_name
            customer.email = customer_email
            customer.save()
        else:
            customer =
Customer.objects.create(phone=customer_phone, name=customer_name,
email=customer_email)

        # Simpan data booking
        booking = Booking.objects.create(
            customer=customer,
            booking_date=current_date,
            status=request.POST.get('status')
        )

        services = request.POST.getlist('services[]')
        datetimes = request.POST.getlist('datetime[]')
        addresses = request.POST.getlist('address[]')
        notes = request.POST.getlist('note[]')
        deal_prices = request.POST.getlist('deal_price[]')

        for service_id, datetime_str, address, note, deal_price
in zip(services, datetimes, addresses, notes, deal_prices):
            schedule = Schedule.objects.create(
                datetime=datetime.fromisoformat(datetime_str),
                address=address,
                note=note,
                deal_price=deal_price,
                service_id=service_id,
                booking=booking
            )

        # Simpan data payment
        payment = Payment.objects.create(
            booking=booking,
```

```
                payment_method=request.POST.get('payment_method'),
                amount=request.POST.get('amount'),
                payment_date=current_date
            )

            if 'image' in request.FILES:
                image = request.FILES['image']
                payment.image = 'images/payment/' + image.name
                image_path     =     os.path.join(settings.MEDIA_ROOT,
'images/payment', image.name)
                with open(image_path, 'wb') as f:
                    for chunk in image.chunks():
                        f.write(chunk)
            else:
                payment.image = 'default.jpg'

            payment.save()

            return redirect('booking')

        return HttpResponseBadRequest('Invalid request method')
```

Baris kode diatas digunakan untuk menyimpan formulir *booking*. Pada saat menyimpan formulir *booking* maka data *booking* akan tersimpan ke dalam *google calendar* terkait. Berikut potongan kode untuk menyimpan halaman *booking* yang otomatis kesimpan *ke event google calendar*.

Proses pengelolaan data detail *booking* terlihat pada Gambar 4.10. Detail halaman *booking* akan menampilkan data *booking* pelanggan yang memiliki beberapa tombol aksi yaitu tambah layanan, edit layanan, hapus layanan, tambah pembayaran dan edit pembayaran.



Gambar 4.10 Halaman detail booking

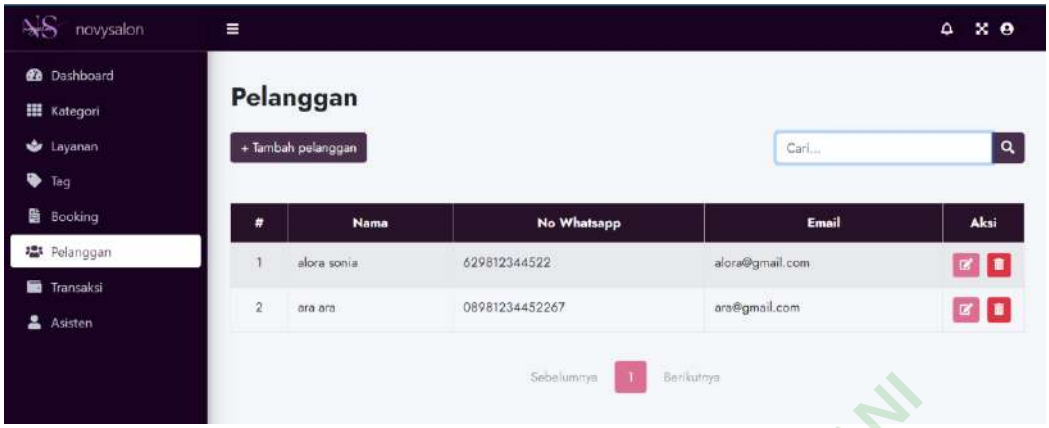
Berikut ini merupakan potongan kode program untuk menampilkan halaman detail booking.

```
def showBooking(request, booking_id):
    booking = get_object_or_404(Booking, pk=booking_id)
    customers = Customer.objects.filter(booking=booking)
    schedules = Schedule.objects.filter(booking=booking)
    services = Service.objects.filter(tags__category__is_book=1)
    payments = Payment.objects.filter(booking=booking)

    for payment in payments:
        if payment.image:
            payment.image = settings.MEDIA_URL + 
str(payment.image)

    return render(request, 'booking/show.html', {'booking': 
booking, 'customers':customers, 'schedules':schedules, 
'payments':payments, 'services':services})
```

Kode diatas digunakan untuk menampilkan halaman detai *booking* yang berelasi dengan beberapa tabel yaitu tabel *customers*, *services* dan *payment*. Terdapat pula logika kode program untuk menampilkan gambar bukti pembayaran. Halaman detail *booking* merupakan halaman yang menampilkan semua jenis layanan dan pembayaran yang terkait dalam satu *booking*. Pada halaman ini admin dapat melakukan pengeditan layanan atau penambahan layanan serta pengeditan pembayaran atau penambahan pembayaran. Admin juga dapat melakukan penghapusan layanan yang tidak di *booking.*

4.3.7 Halaman Pelanggan

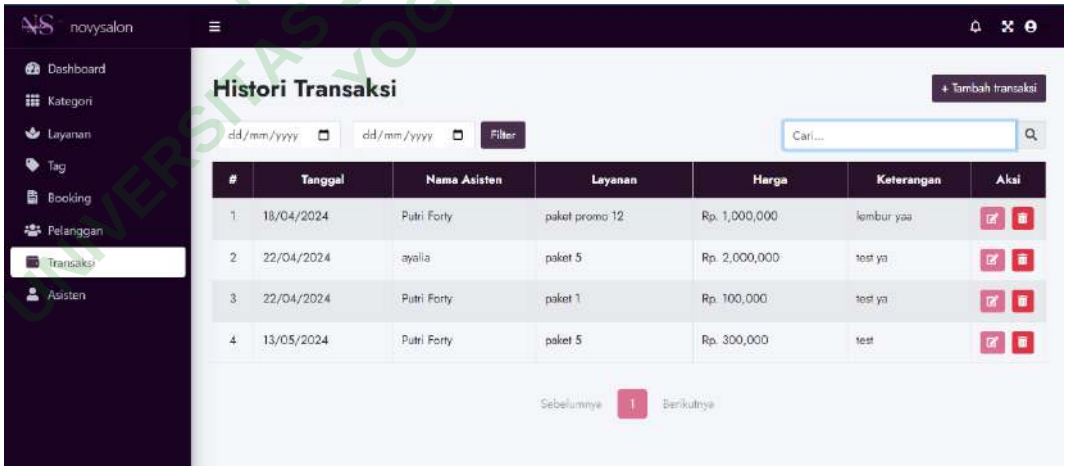
Halaman pelanggan yaitu sebuah halaman yang menampilkan daftar pelanggan yang telah pernah melakukan *booking*. Pada Gambar 4.11 halaman pelanggan terdapat tombol menambahkan data, mengubah data, melakukan penghapusan data serta pencarian data. Tabel pelanggan terdiri dari beberapa kolom yaitu kolom nomor, nama pelanggan, nomor Whatsapp, alamat dan tombol aksi.



Gambar 4.11 Halaman pelanggan

4.3.8 Halaman Transaksi

Halaman transaksi yaitu halaman yang menampilkan informasi daftar transaksi. Transaksi akan digunakan untuk mengukur performa pendapatan yang telah dicapai oleh asisten salon. Pada Gambar 4.12 halaman transaksi memiliki tombol melakukan penambahan data, ubah data, penghapusan data, *filter* transaksi sesuai periode waktu tertentu dan pencarian data transaksi. Tabel transaksi terdiri dari beberapa kolom yaitu kolom nomor, nama asisten, layanan, harga, keterangan dan aksi.



Gambar 4.12 Halaman transaksi

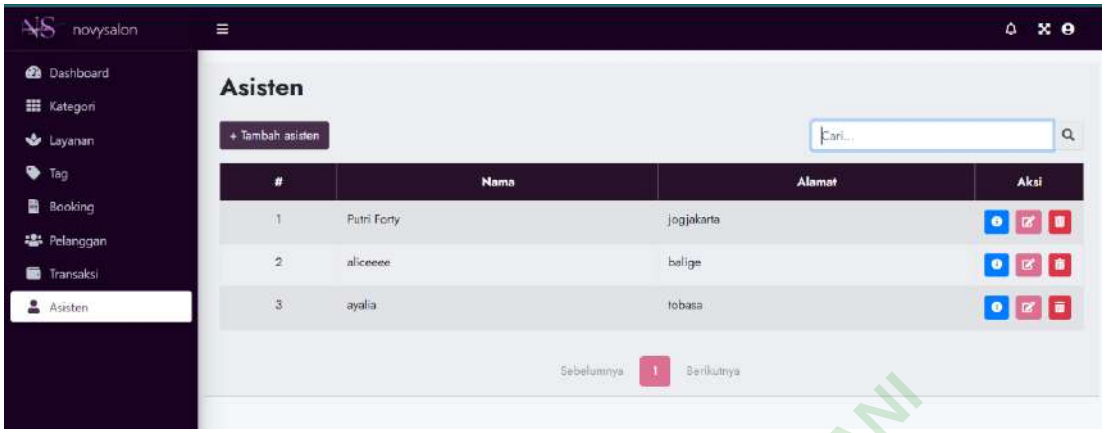
Berikut salah satu fungsi kode program untuk menampilkan data pada halaman data transaksi.

```
    @user_passes_test(is_admin_or_asisten)
    def transaction(request):
        assistants = Assistant.objects.all()
        transactions = Transaction.objects.all()
        query = request.GET.get('search')

        if query:
            transactions                                                =
    Transaction.objects.filter(assistant__name__icontains=query)
        else:
            transactions = Transaction.objects.all()

        if 'start_date' in request.GET and 'end_date' in request.GET:
            start_date_str = request.GET['start_date']
            end_date_str = request.GET['end_date']

            start_date = datetime.strptime(start_date_str,'%Y-%m-%d')
            end_date = datetime.strptime(end_date_str, '%Y-%m-%d')

            transactions =
    transactions.filter(created_at__range=[start_date, end_date])

        paginator = Paginator(transactions, 4)
        page_number = request.GET.get('page')
        page_obj = paginator.get_page(page_number)

    return render(request, 'transaksi/index.html',
            {'transactions':transactions,'assistants':assistants,
            'page_obj':page_obj})
```

Pada baris kode diatas dijelaskan bahwa halaman transaksi dapat diakses oleh admin dan asisten. Kode tersebut digunakan untuk menangani *filter* dalam periode waktu tertentu pada halaman transaksi, *query* kolom pencarian dan juga membuat *pagination* halaman yang menampilkan 4 data per halaman.

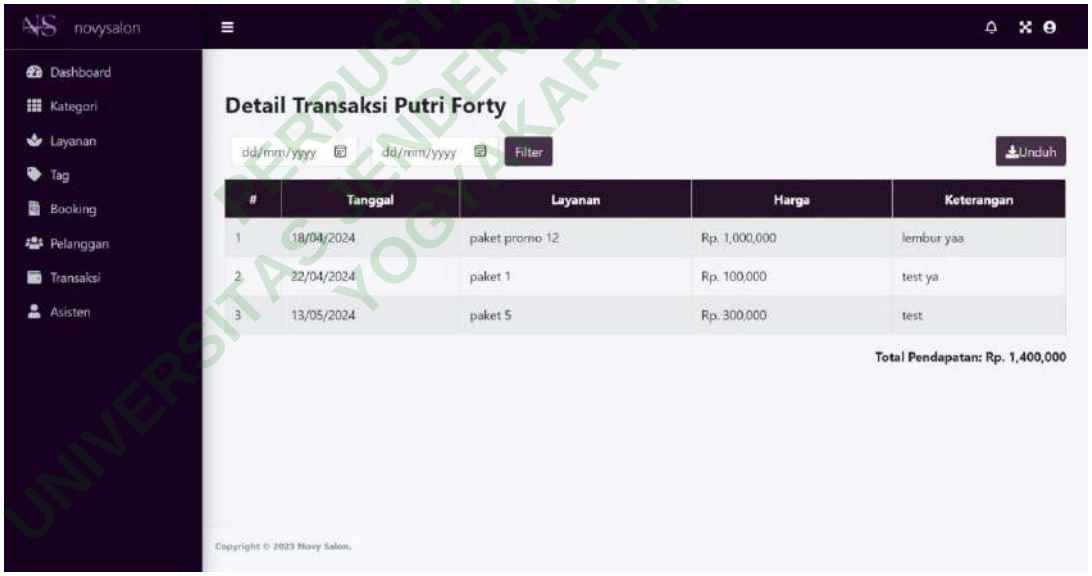
4.3.9 Halaman Asisten

Halaman pelanggan merupakan halaman yang menampilkan infromasi daftar asisten salon. Pada Gambar 4.13 halaman asisten salon memiliki tombol untuk menambah data asisten, edit data asisten, hapus data asisten dan pencarian data asisten serta detail data asisten. Detail data asisten akan menampilkan seluruh daftar transaksi terkait yang telah dilakukan seorang asisten salon. Tabel asisten terdiri dari beberapa kolom yaitu kolom nomor, nama asisten, alamat dan aksi.



Gambar 4.13 Halaman asisten

Halaman asisten memiliki tombol aksi detail yang akan menampilkan detail transaksi asisten salon dan terdapat *filter* tanggal untuk menampilkan data transaksi terkait asisten salon dalam periode waktu tertentu seperti pada Gambar 4.14.



Gambar 4.14 Halaman detail asisten

Dalam halaman detail asisten salon terdapat tombol unduh untuk mengunduh data transaksi asisten salon dalam periode waktu yang dipilih. Hal ini bertujuan untuk menghitung performa asisten salon serta lebih mudah mengelola data transaksi terkait asisten salon yang akan digunakan untuk proses

penghitungan gaji asisten salon. Berikut kode program untuk mengelola proses unduh detail transaksi asisten.

```
def assistant_pdf(request, assistant_id):
    assistant = Assistant.objects.get(pk=assistant_id)
    transactions =
Transaction.objects.filter(assistant_id=assistant_id)
    total_income =
transactions.aggregate(total_income=Sum('deal_price')).get('total
_income') or 0

    template = get_template('pdf_template.html')
    html = template.render({'assistant': assistant,
'transactions': transactions, 'total_income': total_income})

    response = HttpResponse(content_type='application/pdf')
    response['Content-Disposition'] =
f'filename="{assistant.name}_transactions.pdf"'

    from xhtml2pdf import pisa
    pisa_status = pisa.CreatePDF(html, dest=response)

    if pisa_status.err:
        return HttpResponse('We had some errors <pre>' + html +
'</pre>')
return response
```

Dalam penanganan logika untuk unduh data transaksi terlihat seperti potongan kode diatas. Tampilan halaman yang digunakan yaitu dari pdf_template yang telah dibuat sebelumnya. Pada kode diatas terdapat pengelolaan nama file dan juga data yang akan ditampilkan. Package yang digunakan yaitu xhtml2pdf untuk melakukan konversi HTML menjadi PDF. Tampilan halaman unduh seperti terlihat pada Gambar 4.15. Setelah melakukan unduh maka berkas file akan tersimpan kedalam direktori yang diinginkan. Admin dapat menggunakan file ini untuk proses penggajian asisten salon atau proses lainnya sesuai dengan kebutuhan salon.

NOVY SALON

Jl. Tugu 42 Siborongborong Tapanuli Utara Sumatera Utara 22474 Indonesia
Telp. (0274) 563046 - 561392 Fax: (0274) 540628 e-mail: novysalon.gmail.com

Performa Asisten Salon Putri Forty

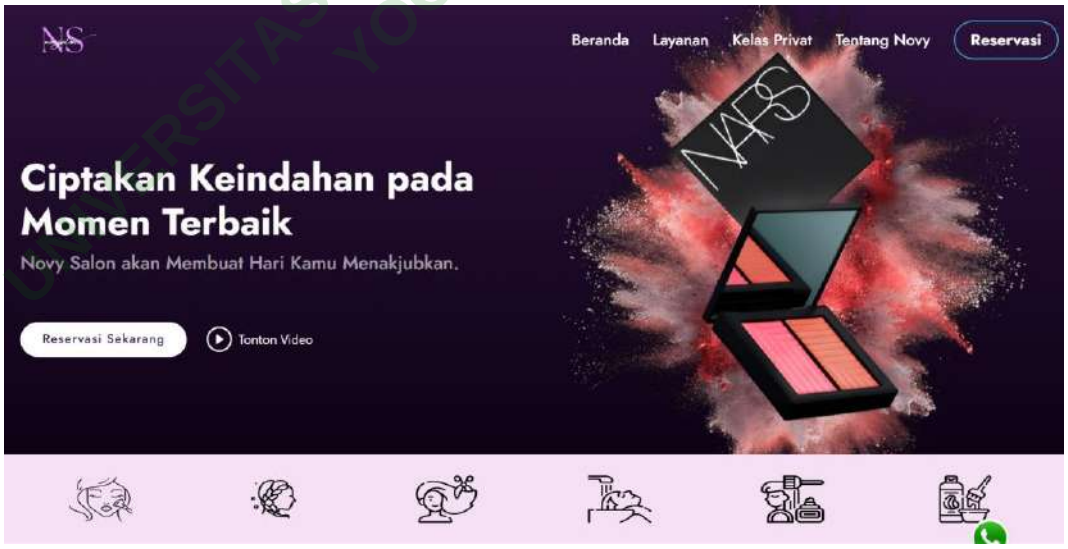
No	Tanggal	Layanan	Harga	Keterangan
1	18 April 2024	paket promo 12	Rp. 1,000,000	lembur yaa
2	22 April 2024	paket 1	Rp. 100,000	test ya
3	13 May 2024	paket 5	Rp. 300,000	test

Total Pendapatan: Rp. 1,400,000

Gambar 4.15 Halaman unduh performa asisten

4.3.10 Halaman Beranda

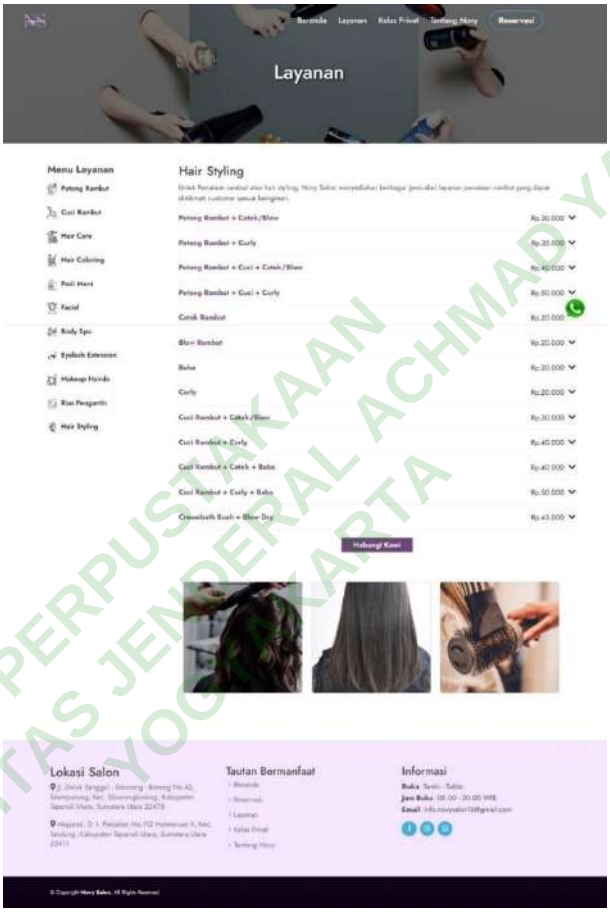
Halaman beranda akan menampilkan informasi yang dapat diakses pengguna yang terdiri dari ringkasan singkat penjelasan tiap menu navigasi. Pada halaman ini, pengguna dapat melihat daftar layanan, jadwal salon, tentang salon, portofolio salon dan layanan kelas privat. Tampilan halaman beranda terlihat seperti Gambar 4.16.



Gambar 4.16 Halaman beranda

4.3.11 Halaman Layanan Pelanggan

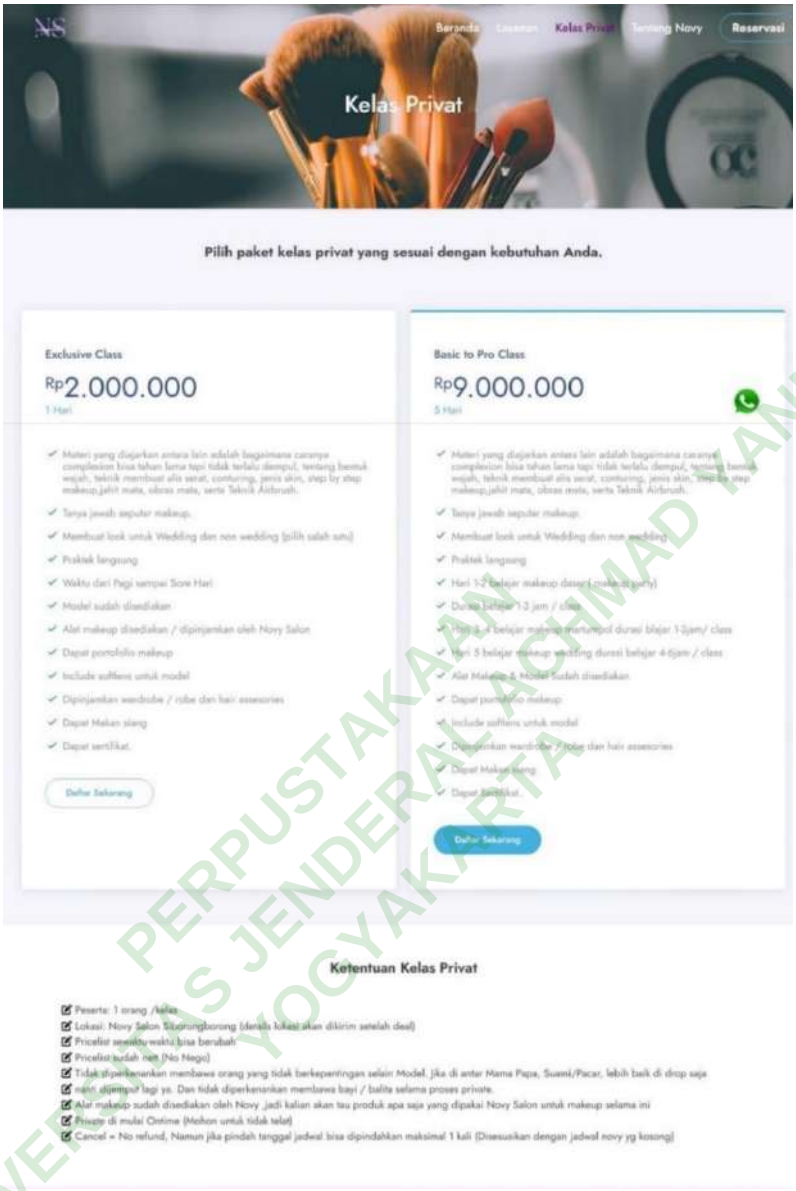
Halaman layanan pelanggan merupakan halaman informasi seluruh layanan salon yang dapat diakses pengguna. Pada halaman ini terdapat tombol hubungi kami jika ingin melakukan diskusi terkait suatu layanan. Halaman layanan pelanggan terlihat seperti Gambar 4.17.



Gambar 4.17 Halaman layanan pelanggan

4.3.12 Halaman Kelas Privat

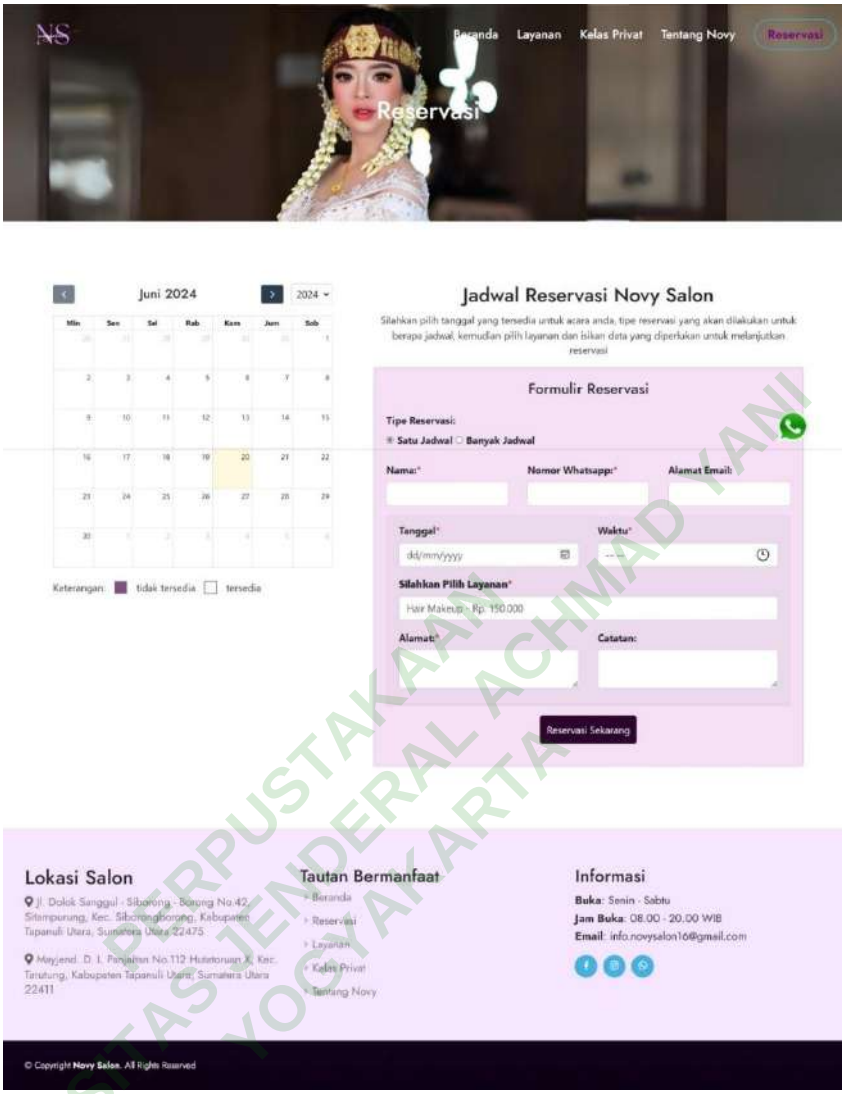
Halaman kelas privat merupakan halaman yang dapat diakses pengguna dan akan menampilkan informasi terkait daftar kelas privat yang ada di Novy salon. Pengguna juga dapat mengakses ketentuan terkait kelas privat. Halaman kelas privat terlihat seperti Gambar 4.18.



Gambar 4.18 Halaman kelas privat

4.3.13 Halaman Reservasi

Halaman reservasi yaitu halaman yang menampilkan informasi terkait jadwal salon dengan keterangan yaitu tersedia atau tidak tersedia. Untuk melakukan reservasi jadwal pelanggan dapat mengisikan formulir reservasi. Halaman reservasi terlihat seperti Gambar 4.19.



Gambar 4.19 Halaman reservasi

Berikut merupakan potongan kode program untuk menyimpan formulir reservasi dari pelanggan.

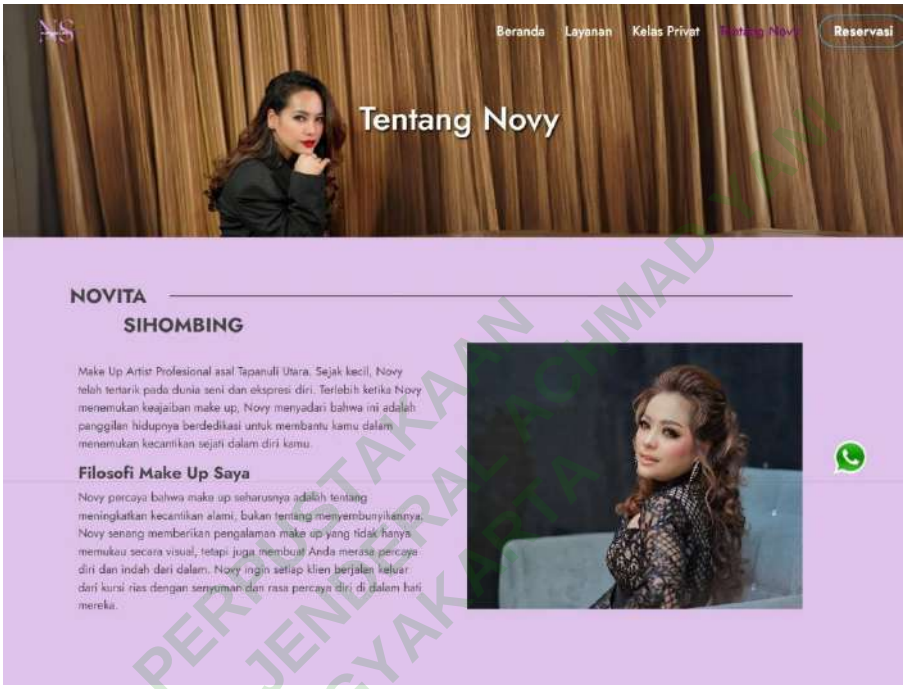
```
def submit_booking(request):
    if request.method == 'POST':
        current_date = timezone.now().date()

        customer_phone = request.POST.get('customer_phone')
        customer_name = request.POST.get('customer_name')
        customer_email = request.POST.get('customer_email')

        existing_customer =
              Customer.objects.filter(phone=customer_phone,
              name=customer_name).first()
        if existing_customer:
```

```
            existing_customer.email = customer_email
            existing_customer.save()
            customer = existing_customer
        else:
            customer =
               Customer.objects.create(phone=customer_phone,
               name=customer_name, email=customer_email)

        status = request.POST.get('status', 'Belum Lunas')

        booking = Booking.objects.create(
            customer=customer,
            booking_date=current_date,
            status=status
        )

        services = request.POST.getlist('services[]')
        addresses = request.POST.getlist('address[]')
        notes = request.POST.getlist('note[]')
        deal_prices = request.POST.getlist('deal_price[]')
        datetimes = request.POST.getlist('datetime[]')

      for service_id, datetime, address, note, deal_price in
      zip(services, datetimes, addresses, notes, deal_prices):
            schedule = Schedule.objects.create(
                datetime=datetime,
                address=address,
                note=note,
                deal_price=deal_price,
                service_id=service_id,
                booking=booking
            )

            Notification.objects.create(
                message=f"Reservasi baru dari {customer_name}.",
                booking_id=booking.id
            )
            messages.success(request, 'Reservasi berhasil
              disimpan.')
            return redirect('reservasi')

    else:
        return redirect('back')
```

Baris kode diatas merupakan fungsi untuk menyimpan data formulir reservasi yang akan menyimpan data *customer*, data *booking* dan data layanan *booking*. Jika sebelumnya *customer* telah melakukan *booking* maka akan data yang akan digunakan yaitu data yang telah tersimpan. Namun, jika belum pernah melakukan *booking* maka *customer* baru akan disimpan. Untuk data layanan *booking* dapat dilakukan dengan lebih dari satu layanan dan tanggal.

4.3.14 Halaman Tentang Novy

Halaman tentang Novy merupakan halaman yang akan menampilkan informasi terkait Novy salon. Pada halaman ini, pelanggan dapat mengakses informasi pemilik salon, lokasi salon dan waktu operasional salon. Halaman tentang Novy terlihat seperti Gambar 4.20.



Gambar 4.20 Halaman tentang Novy

4.4 PENGUJIAN APLIKASI

4.4.1 *Black Box Testing*

Black box testing yaitu pengujian fungsionalitas pada aplikasi. Pengujian dilakukan oleh 2 orang Pakar dan 1 orang admin. Berdasarkan pengujian yang dilakukan, diperoleh hasil pengujiannya yang selengkapnya tercantum di bagian Lampiran 2 serta diperoleh kesimpulan bahwa fitur aplikasi berjalan dengan baik sesuai dengan tujuan awal dibangunnya aplikasi.

4.4.2 *User Acceptance Test*

User Acceptance Test pada aplikasi ini yaitu dengan mengajukan beberapa pertanyaan terkait kelayakan fungsionalitas aplikasi kepada 14 responden yaitu

sembilan orang asisten salon, pemilik salon dan tiga orang pelanggan salon. Rumus dalam menghitung persentase jawaban responden sesuai dengan skor yang akan dijabarkan seperti dibawah ini:

$$P = \frac{S}{\text{Skor Ideal}} \times 100\%$$

Keterangan:

P = Persentase

S = Skor setiap jawaban dikali frekuensi

Skor ideal = Skor tertingi dikali jumlah responden

Skor ideal = 5 x jumlah responden.

4.4.2.1 Pengujian UAT Admin

Hasil persentase pengujian berdasarkan akumulasi jawaban dari setiap pertanyaan dijabarkan sebagai berikut:

1. Apakah aplikasi pemesanan layanan tata rias dapat mempermudah dalam pengelolaan jadwal reservasi di Novy salon?

Tabel 4.1 Tabel Skor Pertanyaan Pertama - Admin

Skala Jawaban	Skor	Frekuensi	Persentase
SS	5	1	5 x 1 = 5
S	4	0	4 x 0 = 0
KS	3	0	3 x 0 = 0
TS	2	0	2 x 0 = 0
STS	1	0	1 x 0 = 0
Jumlah			5

$$P = \frac{5}{5} \times 100\% = 100\%$$

Berdasarkan Tabel skor 4.1 dapat dijelaskan bahwa aplikasi pemesanan layanan tata rias dapat mempermudah dalam pengelolaan jadwal reservasi di Novy salon dengan nilai 100%.

2. Apakah aplikasi dapat dijadikan sebagai *database* penyimpanan data transaksi agar lebih terstruktur?

Tabel 4.2 Tabel Skor Pertanyaan Kedua - Admin

Skala Jawaban	Skor	Frekuensi	Persentase
SS	5	1	5 x 1 = 5
S	4	0	4 x 0 = 0
KS	3	0	3 x 0 = 0
TS	2	0	2 x 0 = 0
STS	1	0	1 x 0 = 0
Jumlah			5

$$P = \frac{5}{5} \times 100\% = 100\%$$

Berdasarkan Tabel skor 4.2 dapat dijelaskan bahwa aplikasi dapat dijadikan sebagai *database* penyimpanan data transaksi agar lebih terstruktur dengan nilai 100 %.

3. Apakah dengan adanya aplikasi mempermudah pengelolaan transaksi yang dilakukan setiap asisten salon?

Tabel 4.3 Tabel Skor Pertanyaan Ketiga - Admin

Skala Jawaban	Skor	Frekuensi	Persentase
SS	5	1	5 x 1 = 5
S	4	0	4 x 0 = 0
KS	3	0	3 x 0 = 0
TS	2	0	2 x 0 = 0
STS	1	0	1 x 0 = 0
Jumlah			5

$$P = \frac{5}{5} \times 100\% = 100\%$$

Berdasarkan Tabel skor 4.3 dapat dijelaskan bahwa dengan adanya aplikasi mempermudah pengelolaan transaksi yang dilakukan setiap asisten salon dengan nilai 100%.

4. Apakah dengan adanya unduh transaksi terkait asisten mempermudah pengelolaan data rekap transaksi asisten?

Tabel 4.4 Tabel Skor Pertanyaan Keempat - Admin

Skala Jawaban	Skor	Frekuensi	Persentase
SS	5	1	5 x 1 = 5
S	4	0	4 x 0 = 0
KS	3	0	3 x 0 = 0
TS	2	0	2 x 0 = 0
STS	1	0	1 x 0 = 0
Jumlah			5

$$P = \frac{5}{5} \times 100\% = 100\%$$

Berdasarkan Tabel skor 4.4 dapat dijelaskan bahwa adanya unduh transaksi terkait asisten mempermudah pengelolaan data rekap transaksi asisten dengan nilai 100 %.

5. Apakah aplikasi mempermudah admin dalam pengelolaan data layanan di salon?

Tabel 4.5 Tabel Skor Pertanyaan Kelima - Admin

Skala Jawaban	Skor	Frekuensi	Persentase
SS	5	1	5 x 1 = 5
S	4	0	4 x 0 = 0
KS	3	0	3 x 0 = 0

TS	2	0	2 x 0 = 0
STS	1	0	1 x 0 = 0
Jumlah			5

$$P = \frac{5}{5} \times 100\% = 100\%$$

Berdasarkan Tabel skor 4.5 dapat dijelaskan bahwa aplikasi mempermudah admin dalam pengelolaan data layanan di salon dengan perolehan nilai 100 %.

4.4.2.2 Pengujian UAT Asisten salon

Hasil persentase pengujian berdasarkan akumulasi jawaban dari setiap pertanyaan dijabarkan sebagai berikut:

1. Apakah fitur pengelolaan data transaksi dapat dipahami dengan baik?

Tabel 4.6 Tabel Skor Pertanyaan Pertama – Asisten Salon

Skala Jawaban	Skor	Frekuensi	Persentase
SS	5	7	5 x 7 = 35
S	4	2	4 x 2 = 8
KS	3	0	3 x 0 = 0
TS	2	0	2 x 0 = 0
STS	1	0	1 x 0 = 0
Jumlah			43

$$P = \frac{43}{45} \times 100\% = 95\%$$

Berdasarkan Tabel skor 4.6 dapat dijelaskan bahwa fitur pengelolaan data transaksi dapat dipahami dengan baik oleh asisten salon dengan nilai 95%.

2. Apakah dengan adanya fitur transaksi memudahkan asisten salon dalam melakukan rekap data transaksi salon?

Tabel 4.7 Tabel Skor Pertanyaan Kedua – Asisten Salon

Skala Jawaban	Skor	Frekuensi	Persentase
SS	5	9	5 x 9 = 45
S	4	0	4 x 0 = 0
KS	3	0	3 x 0 = 0
TS	2	0	2 x 0 = 0
STS	1	0	1 x 0 = 0
Jumlah			45

$$P = \frac{45}{45} \times 100\% = 100\%$$

Berdasarkan Tabel skor 4.7 dapat dijelaskan bahwa adanya fitur transaksi memudahkan asisten salon dalam melakukan rekap data transaksi salon dengan nilai 100%.

3. Apakah fitur transaksi meringankan pekerjaan asisten salon sehingga tidak perlu menulis kembali rekap perbulan?

Tabel 4.8 Tabel Skor Pertanyaan Ketiga – Asisten Salon

Skala Jawaban	Skor	Frekuensi	Persentase
SS	5	7	5 x 7 = 35
S	4	1	4 x 1 = 4
KS	3	0	3 x 0 = 0
TS	2	1	2 x 1 = 2
STS	1	0	1 x 0 = 0
Jumlah			41

$$P = \frac{41}{41} \times 100\% = 91\%$$

Berdasarkan Tabel skor 4.8 dapat dijelaskan bahwa fitur transaksi meringankan pekerjaan asisten salon sehingga tidak perlu menulis kembali rekap perbulan dengan nilai 91%.

4. Apakah asisten salon dapat dengan mudah melakukan penambahan data transaksi?

Tabel 4.9 Tabel Skor Pertanyaan Keempat – Asisten Salon

Skala Jawaban	Skor	Frekuensi	Persentase
SS	5	6	5 x 6 = 30
S	4	3	4 x 3 = 12
KS	3	0	3 x 0 = 0
TS	2	0	2 x 0 = 0
STS	1	0	1 x 0 = 0
Jumlah			42

$$P = \frac{42}{45} \times 100\% = 93\%$$

Berdasarkan Tabel skor 4.9 dapat dijelaskan bahwa asisten salon dapat dengan mudah melakukan penambahan data transaksi dengan nilai 93%.

5. Apakah tabel data transaksi sesuai dengan tabel data transaksi secara manual?

Tabel 4.10 Tabel Skor Pertanyaan Kelima – Asisten Salon

Skala Jawaban	Skor	Frekuensi	Persentase
SS	5	6	5 x 6 = 30
S	4	3	4 x 3 = 12
KS	3	0	3 x 0 = 0
TS	2	0	2 x 0 = 0
STS	1	0	1 x 0 = 0
Jumlah			42

$$P = \frac{42}{45} \times 100\% = 93\%$$

Berdasarkan Tabel skor 4.10 dapat dijelaskan bahwa tabel data transaksi sesuai dengan tabel data transaksi secara manual dengan nilai 93%.

4.4.2.3 Pengujian UAT Pelanggan

Hasil persentase pengujian berdasarkan akumulasi jawaban dari setiap pertanyaan dijabarkan sebagai berikut:

1. Apakah pelanggan dapat dengan mudah mengakses informasi terkait jadwal reservasi Novy salon?

Tabel 4.11 Tabel Skor Pertanyaan Pertama – Pelanggan

Skala Jawaban	Skor	Frekuensi	Persentase
SS	5	2	5 x 2 = 10
S	4	1	4 x 1 = 4
KS	3	0	3 x 0 = 0
TS	2	0	2 x 0 = 0
STS	1	0	1 x 0 = 0
Jumlah			14

$$P = \frac{14}{15} \times 100\% = 93\%$$

Berdasarkan Tabel skor 4.11 dapat dijelaskan bahwa pelanggan dapat dengan mudah mengakses informasi terkait jadwal reservasi Novy salon dengan nilai 93%.

2. Apakah pelanggan dapat dengan mudah melakukan reservasi jadwal?

Tabel 4.12 Tabel Skor Pertanyaan Kedua – Pelanggan

Skala Jawaban	Skor	Frekuensi	Persentase
SS	5	0	5 x 0 = 0
S	4	3	4 x 3 = 12
KS	3	0	3 x 0 = 0

TS	2	0	2 x 0 = 0
STS	1	0	1 x 0 = 0
Jumlah			12

$$P = \frac{12}{15} \times 100\% = 80\%$$

Berdasarkan Tabel skor 4.12 dapat dijelaskan bahwa pelanggan dapat dengan mudah melakukan reservasi jadwal dengan nilai 80%.

3. Apakah pelanggan dapat dengan mudah mengakses informasi layanan yang ada di Novy salon?

Tabel 4.13 Tabel Skor Pertanyaan Ketiga – Pelanggan

Skala Jawaban	Skor	Frekuensi	Persentase
SS	5	0	5 x 0 = 0
S	4	3	4 x 3 = 12
KS	3	0	3 x 0 = 0
TS	2	0	2 x 0 = 0
STS	1	0	1 x 0 = 0
Jumlah			12

$$P = \frac{12}{15} \times 100\% = 80\%$$

Berdasarkan Tabel skor 4.13 dapat dijelaskan bahwa pelanggan dapat dengan mudah mengakses informasi layanan salon dengan nilai 80%.

4.5 PEMBAHASAN

Aplikasi pemesanan layanan tata rias Novy salon dapat dijadikan solusi dalam pengelolaan data jadwal agar pelanggan lebih mudah mengakses informasi terkait jadwal salon. Hal ini menjadi poin penting sehingga pelanggan tidak perlu mengunjungi salon terlebih dahulu serta pemilik salon dapat lebih efisien dalam

pengelolaan jadwal. Dalam aplikasi ini terdapat beberapa fitur yang diharapkan dapat membantu admin dan asisten.

Selain itu, pengelolaan data transaksi memudahkan asisten salon dalam melakukan pencatatan transaksi harian sehingga risiko data hilang atau rusak dapat diatasi. Sistem penggajian di Novy salon menggunakan sistem persentase dimana setiap transaksi yang dilakukan asisten selama sebulan dijumlahkan dan diakumulasikan dalam rentang persen tertentu. Dalam hal penghitungan gaji karyawan, pemilik salon dapat lebih mudah dalam proses penggajian serta lebih efisien dan akurat.

Berdasarkan pengujian *black box* yang dilakukan diperoleh akumulasi hasil pengujian dalam Tabel 4.14.

Tabel 4.14 Hasil Pengujian Black Box

Pengujian aplikasi pemesanan tata rias Novy salon	Pakar		Admin	Total
	A	B		
Halaman *login*	✓	✓	✓	100%
Halaman dashboard	✓	✓	✓	100%
Halaman kategori	✓	✓	✓	100%
Halaman layanan	✓	✓	✓	100%
Halaman tag	✓	✓	✓	100%
Halaman booking	✓	✓	✓	100%
Halaman pelanggan	✓	✓	✓	100%
Halaman transaksi	✓	✓	✓	100%
Halaman asisten	✓	✓	✓	100%

Tabel 4.14 menunjukkan bahwa fitur yang ada pada aplikasi pemesanan layanan tata rias Novy salon 100% berfungsi dengan baik. Berdasarkan pengujian *black box* yang dilakukan terdapat beberapa saran dari pakar A yang melakukan pengujian yaitu dalam hal pengisian formulir pada menu layanan, tag, kategori, booking agar menampilkan pesan validasi langsung. Pakar B menyarankan agar

penggunaan kolom pencarian lebih ditingkatkan lagi dan prosesnya lebih halus sehingga tidak mengganggu proses.

Berdasarkan hasil pengujian menggunakan *user acceptance test* pada bagian admin diperoleh hasil rata rata yaitu 100% yang berarti sistem dapat digunakan oleh admin dan mempermudah pekerjaan admin. Pada bagian asisten salon diperoleh hasil dengan nilai 94,4% yang berarti asisten salon dapat menggunakan sistem untuk melakukan input transaksi serta mempermudah asisten salon dalam mengelola data terkait transaksi. Pada bagian pelanggan diperoleh hasil dengan nilai 84,3%, yang berarti asisten salon dapat dengan mudah mengakses jadwal reservasi salon serta dapat melakukan reservasi jadwal yang diinginkan. Maka hasil pengujian dengan menggunakan *user acceptance test* memperoleh nilai rata-rata yaitu 92,9%.

Dalam proses pengembangan aplikasi diperoleh beberapa hasil Sprint yang telah dilakukan yaitu:

- Sprint 1: Halaman Admin

Pada Sprint 1, berhasil mengembangkan dan menyelesaikan fitur-fitur admin termasuk dashboard admin. Dashboard ini dirancang untuk memberikan kemudahan bagi admin dalam mengelola berbagai aspek aplikasi, termasuk manajemen jadwal salon, detail pemesanan, dan jadwal buka atau tutup tanggal tertentu. Fitur utama yang dikembangkan meliputi tampilan statistik pemesanan serta manajemen data jadwal yang terintegrasi langsung dari *event google calender*. Selama pengembangan, terdapat beberapa *review* untuk memastikan semua kebutuhan fungsional telah terpenuhi dan tampilan antarmuka sesuai dengan standar *user experience* (UX) yang baik. *Review* sangat membantu dalam menyempurnakan tampilan dan navigasi dashboard, sehingga lebih intuitif dan mudah digunakan oleh admin. Hasil akhir dari Sprint 1 menunjukkan bahwa semua fitur dashboard admin berfungsi dengan baik namun terdapat penyesuaian lebih lanjut terkait detail jadwal yang akan dilanjutkan pada Sprint 2.

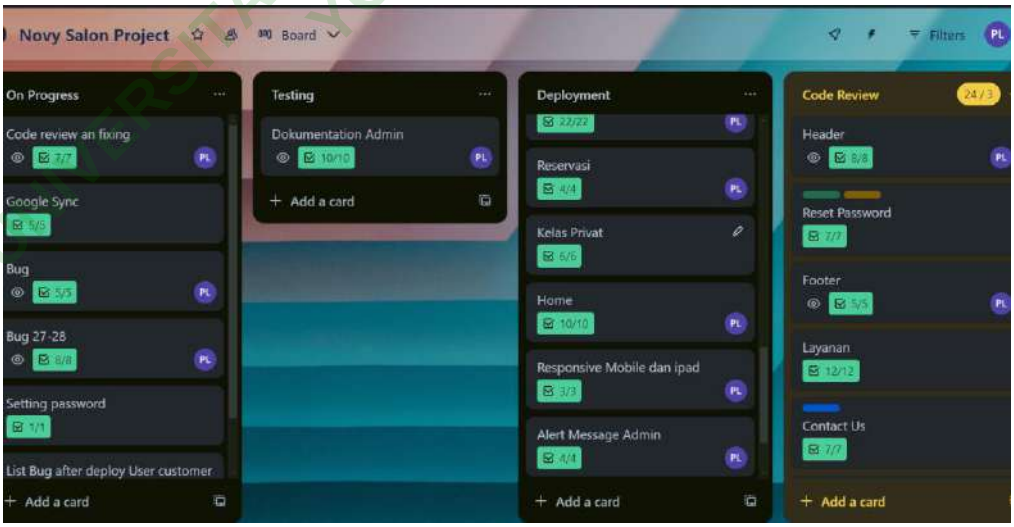
- Sprint 2: Halaman Pelanggan

Pada Sprint 2, fokus utama adalah mengembangkan fitur reservasi pelanggan yang memungkinkan pengguna untuk memesan layanan tata rias

melalui aplikasi. Fitur ini mencakup pemilihan layanan, pemilihan jadwal, dan konfirmasi pemesanan. Pengguna dapat dengan mudah memilih jenis layanan sesuai ketersediaan tanggal, lalu melakukan pemesanan secara *real-time*. Selain itu, sistem reservasi ini juga diintegrasikan dengan notifikasi untuk mengingatkan admin tentang jadwal yang telah pelanggan pesan. Selama sprint ini, saya memastikan bahwa semua proses pemesanan berjalan lancar dan data yang dimasukkan oleh pengguna tervalidasi dengan baik. Beberapa *feedback* selama Sprint 2 membantu memperbaiki beberapa aspek seperti alur pengguna (*user flow*) dan validasi input untuk memastikan tidak ada kesalahan dalam proses pemesanan serta pemilihan untuk melakukan reservasi banyak jadwal. Dengan selesainya Sprint 2, fitur reservasi pelanggan telah berfungsi sesuai dengan yang diharapkan, memberikan pengalaman yang lancar dan intuitif bagi pengguna dalam memesan layanan tata rias di Novy Salon.

Dalam pengembangan aplikasi, backlog adalah daftar tugas atau fitur yang perlu dikerjakan terlihat pada Gambar 4.21. Backlog ini berisi semua item pekerjaan yang belum diselesaikan dan masih perlu diimplementasikan dalam iterasi atau sprint berikutnya.

Gambar 4.21 *Backlog*



Dalam konteks Sprint 1, backlog digunakan untuk merencanakan dan mengatur tugas-tugas yang diperlukan untuk mengembangkan fitur dashboard

admin. Meskipun sebagian besar tugas berhasil diselesaikan, ada beberapa penyesuaian detail terkait manajemen jadwal yang belum terselesaikan dan kemudian dimasukkan ke dalam backlog untuk Sprint 2. Ini memungkinkan untuk fokus pada pekerjaan yang paling penting dan sesuai prioritas, sambil tetap fleksibel dalam menanggapi perubahan atau masukan selama proses pengembangan.

Penggunaan backlog dalam Agile tidak hanya sebagai daftar tugas, tetapi juga sebagai alat untuk mengelola dan memprioritaskan pekerjaan secara efektif. Ini memastikan bahwa setiap iterasi atau sprint menghasilkan nilai tambah yang signifikan dan memungkinkan adaptasi yang cepat terhadap perubahan kebutuhan atau prioritas pengguna. Dengan memanfaatkan backlog dengan baik, pengembangan lebih terstruktur dan efisien, memastikan bahwa pengembangan aplikasi berjalan sesuai dengan tujuan dan kebutuhan yang ditetapkan.