BAB 4
HASIL PENELITIAN

4.1 RINGKASAN HASIL PENELITIAN

Penelitian ini telah berhasil mengembangkan sebuah Sistem Pendukung Keputusan (SPK) berbasis web yang bertujuan untuk membantu calon mahasiswa Unjaya dalam memilih jurusan kuliah yang sesuai dengan minat dan kepribadian mereka. Sistem ini dibangun menggunakan *framework* Django, salah satu *framework* web berbasis Python yang mendukung pengembangan aplikasi yang cepat, terstruktur, dan aman. Sistem memanfaatkan integrasi antara teori kepribadian RIASEC dan algoritma klasifikasi K-Nearest Neighbor (KNN) sebagai metode utama dalam proses rekomendasi jurusan. Pengguna sistem akan mengerjakan tes berbasis RIASEC yang terdiri dari 42 pernyataan. Setiap jawaban "Ya" dikonversi menjadi skor untuk aspek kepribadian RIASEC: *realistic*, *investigative*, *artistic*, *social*, *enterprising*, dan *conventional*. Skor ini kemudian disusun menjadi sebuah vektor yang akan digunakan dalam proses klasifikasi.

Dataset klasifikasi yang digunakan dalam sistem ini diperoleh melalui penyebaran kuesioner RIASEC kepada mahasiswa Unjaya. Setiap responden mewakili satu vektor RIASEC dan dikaitkan dengan jurusan yang sedang mereka tempuh. Data ini selanjutnya digunakan sebagai basis pembelajaran sistem untuk mengenali pola minat dan jurusan. Dalam proses klasifikasi, sistem menggunakan algoritma KNN untuk mencari 10 tetangga terdekat berdasarkan Euclidean Distance, kemudian dari hasil tersebut dipilih 3 jurusan berbeda yang memiliki kemiripan tertinggi terhadap vektor minat pengguna. Pendekatan ini memungkinkan sistem memberikan rekomendasi yang lebih beragam dan relevan karena satu jurusan direpresentasikan oleh dua vektor RIASEC, sehingga meningkatkan akurasi pencocokan minat.

Sistem juga menyajikan hasil dalam bentuk visualisasi grafik radar (radar *chart*) yang menggambarkan skor dari masing-masing aspek RIASEC. Selain itu, sistem menampilkan aspek minat tertinggi lengkap dengan deskripsi kepribadian,

serta memungkinkan pengguna untuk mencetak hasil rekomendasi ke dalam format pdf. Dengan demikian, sistem ini tidak hanya memberikan rekomendasi jurusan berdasarkan data empiris mahasiswa Unjaya, tetapi juga mendukung pemahaman diri pengguna melalui tampilan data yang informatif dan mudah dipahami. Sistem ini telah berjalan sesuai dengan tujuan penelitian dan berpotensi untuk dikembangkan lebih lanjut dalam skala yang lebih luas.

4.2 IMPLEMENTASI DESAIN ANTARMUKA

Desain antarmuka pada sistem ini dirancang dengan memperhatikan prinsip *user friendly*, kemudahan navigasi, dan tampilan yang responsif. Implementasi antarmuka dilakukan menggunakan kombinasi teknologi HTML, CSS, JavaScript, serta *framework* Bootstrap untuk memastikan tampilan antar perangkat tetap optimal dan konsisten.

4.2.1 Implementasi *Landing Page*

Implementasi dari *landing page* pada sistem ini dilakukan dengan mengacu langsung pada desain awal yang telah dirancang pada bagian 3.3.3.1 Desain *Landing Page*. Secara keseluruhan, bentuk dan susunan elemen antarmuka yang diimplementasikan hampir 100% menyerupai desain awal, baik dari segi tata letak, pemilihan warna, ikon, hingga tipografi. Namun, terdapat beberapa penyesuaian kecil yang dilakukan pada tahap implementasi guna meningkatkan tampilan dan kualitas informasi yang disajikan kepada pengguna. Adapun perbedaan tersebut antara lain:

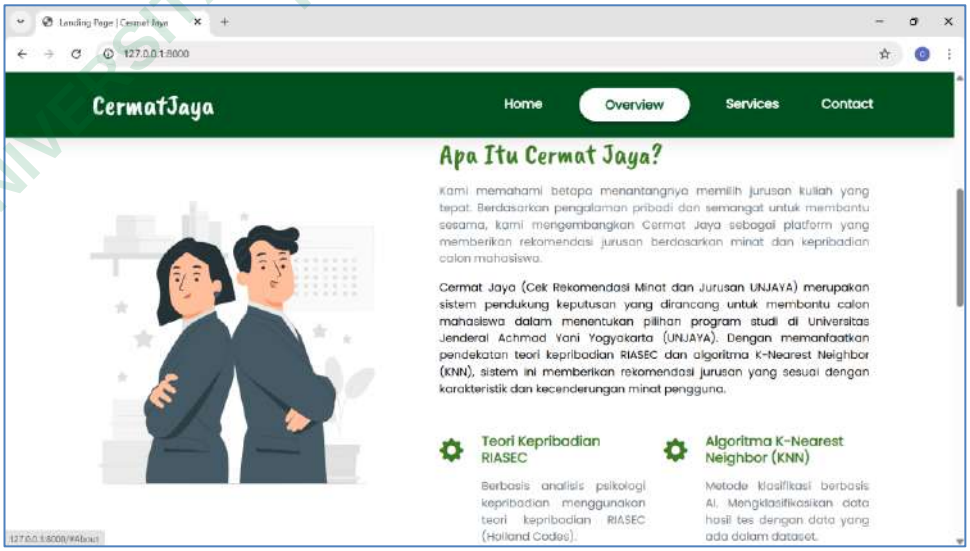
1. Penambahan ikon pada tombol *login* di *section home*

   Hal ini berfungsi sebagai representasi visual untuk memperjelas fungsi tombol tersebut. Ikon ini tidak dirancang pada tahap awal namun ditambahkan saat implementasi guna meningkatkan daya tarik visual dan intuitivitas bagi pengguna, khususnya pengguna baru. Implementasi desain *landing page* di *section home* pada sistem ini ditunjukkan pada Gambar 4.1 berikut.

Gambar 4.1 Implementasi Desain *Landing Page* di *Section Home*

2. *Section overview*

Section ini mengalami sedikit penambahan elemen berupa ilustrasi grafis serta penyesuaian kalimat deskripsi agar terlihat lebih komunikatif dan informatif. Tujuannya adalah untuk memperkuat pemahaman pengguna terhadap metode rekomendasi jurusan yang ditawarkan oleh sistem ini. Implementasi desain *landing page* di *section overview* pada sistem ini ditunjukkan pada Gambar 4.2 berikut.



Gambar 4.2 Implementasi Desain *Landing Page* di *Section Overview*

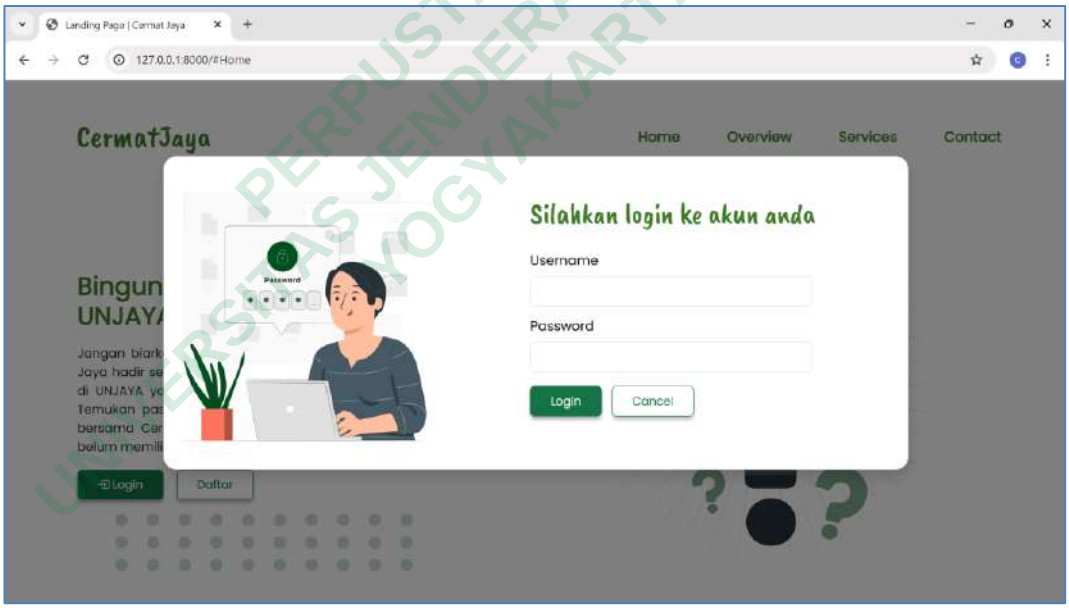
3. Penambahan *section footer*

   Section ini tidak ditampilkan pada desain awal. *Footer* ini berisi informasi tambahan seperti *copyright*, tautan sosial media, dan identitas sistem. Implementasi desain *landing page* di *section footer* pada sistem ini ditunjukkan pada Gambar 4.3 berikut.

Gambar 4.3 Implementasi Desain *Landing Page* di *Section Footer*

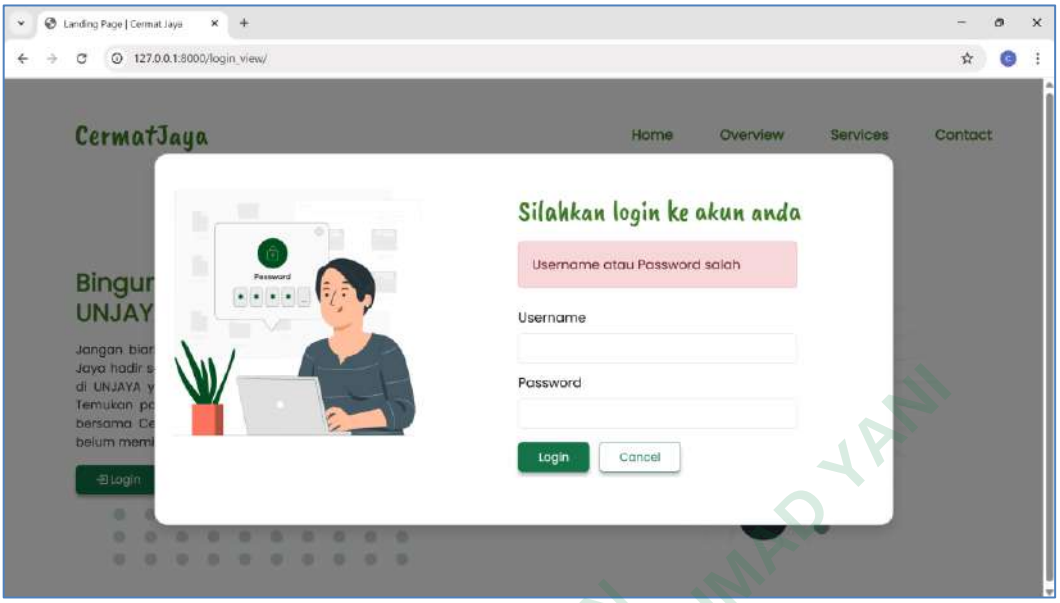
4.2.2 Implementasi *Pop-Up Login*

Implementasi *pop-up login* dalam sistem ini telah direalisasikan sesuai dengan rancangan yang telah ditentukan pada bagian 3.3.3.2 Desain *Pop-Up Login*. Tampilan dan fungsionalitas dari *pop-up login* dirancang menggunakan komponen Bootstrap modal. Implementasi desain *pop-up login* pada sistem ini ditunjukkan pada Gambar 4.4 berikut.



Gambar 4.4 Implementasi Desain *Pop-Up Login*

Kemudian implementasi desain antarmuka *pop-up login* ketika terjadi kesalahan autentikasi (salah memasukkan *username* atau *password*) ditunjukkan pada Gambar 4.5 berikut.



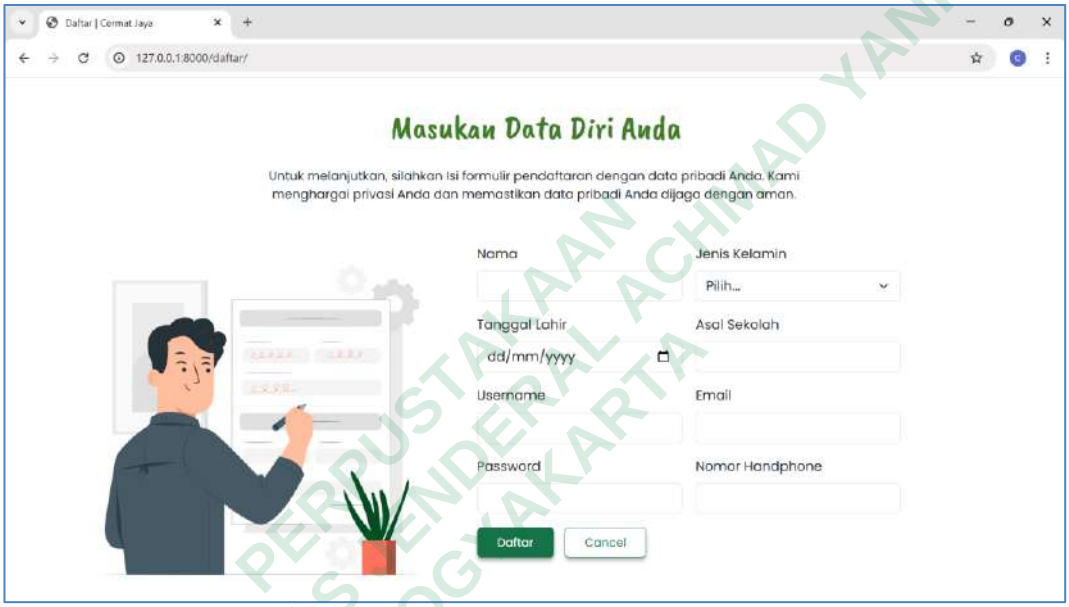
Gambar 4.5 Implementasi Desain *Pop-Up Login* (Kesalahan Autentikasi)

Potongan kode program pada fungsi *login_view* berikut berfungsi untuk menangani proses autentikasi pengguna saat melakukan *login* melalui *pop-up login* yang telah diimplementasikan. Jika autentikasi gagal, maka variabel msg akan berisi pesan kesalahan "*Username* atau *Password* salah", yang kemudian dikirimkan ke *template* index.html untuk ditampilkan di dalam *pop-up login*. Potongan kode ini terdapat pada *file* views.py.

```
def login_view(request):
    msg = None
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']
        user    =    authenticate(request,    username=username,
password=password)
        if user is not None:
            login(request, user)
            return redirect('dashboard')
        else:
            msg = "Username atau Password salah"
    return render(request, 'index.html', {'msg': msg})
```

4.2.3 Implementasi Halaman Pendaftaran

Halaman pendaftaran pada sistem ini telah diimplementasikan sesuai dengan desain antarmuka yang telah dirancang sebelumnya pada bagian 3.3.3.3 Desain Halaman Pendaftaran. Secara keseluruhan, implementasi halaman ini 100% mengikuti desain awal tanpa ada perubahan baik dari segi struktur tampilan, susunan elemen, warna, maupun posisi *field* input. Implementasi desain halaman pendaftaran pada sistem ini ditunjukkan pada Gambar 4.6 berikut.



Gambar 4.6 Implementasi Desain Halaman Pendaftaran

Adapun implementasi proses pendaftaran pada sisi backend dilakukan melalui fungsi daftar pada *file* views.py. Fungsi ini menangani pengambilan data dari formulir, validasi ketersediaan *username*, pembuatan akun baru, serta memberikan respons berupa notifikasi SweetAlert. Berikut kode dari fungsi daftar.

```
def daftar(request):
    if request.method == 'POST':
        # Ambil data dari formulir pendaftaran
        username = request.POST['username']
        password = request.POST['password']
        nama = request.POST['nama']
        jenis_kelamin = request.POST['jenis_kelamin']
        tanggal_lahir = request.POST['tanggal_lahir']
        asal_sekolah = request.POST['asal_sekolah']
        no_hp = request.POST['no_hp']
        email = request.POST['email']
```

```
        # Cek apakah username sudah digunakan
        if User.objects.filter(username=username).exists():
            # Tampilkan SweetAlert jika username sudah digunakan
            response_data = {
                'width': 600,
                'title': 'Username Tidak Tersedia',
                'text': 'Username sudah digunakan. Silahkan pilih
                        username lain.',
                'icon': 'error',
                'confirmButtonColor': "#157347",
            }
            return HttpResponse(json.dumps(response_data),
                   content_type='application/json')

        # Buat objek pengguna baru
        user = User.objects.create_user(username=username,
                   password=password, first_name=nama,
                   jenis_kelamin=jenis_kelamin,
                   tanggal_lahir=tanggal_lahir,
                   asal_sekolah=asal_sekolah,
                   no_hp=no_hp, email=email)

        # Tampilkan SweetAlert jika pendaftaran berhasil
        response_data = {
            'width': 600,
            'title': 'Pendaftaran Berhasil',
            'text': 'Silahkan login untuk melanjutkan.',
            'icon': 'success',
            'confirmButtonColor': "#157347",
        }
        return HttpResponse(json.dumps(response_data),
            content_type='application/json')

    return render(request, 'daftar.html')
```

4.2.4 Implementasi Halaman *Dashboard*

Implementasi halaman *dashboard* secara keseluruhan mengikuti rancangan yang telah disusun pada bagian 3.3.3.4 Desain Halaman *Dashboard*. Desain dan implementasi memiliki kesamaan hingga 100%, baik dari sisi struktur tampilan maupun fungsionalitas. Implementasi desain halaman *dashboard* pada sistem ini ditunjukkan pada Gambar 4.7 berikut.

Gambar 4.7 Implementasi Desain Halaman *Dashboard*

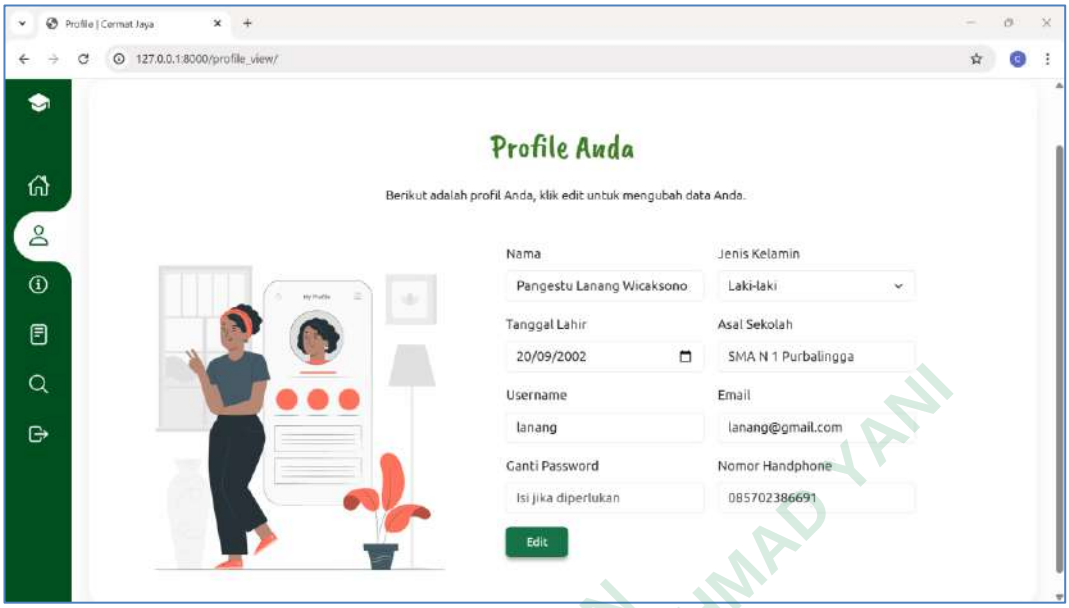
Potongan kode berikut merupakan bagian dari fungsi *dashboard* yang terdapat pada *file* views.py. Fungsi ini bertanggung jawab untuk mengelola data yang akan ditampilkan pada halaman dashboard, baik data statistik umum maupun visualisasi hasil tes RIASEC dari pengguna yang sedang *login*.

```
@login_required
def dashboard(request):
    # Mengambil data statistik
    total_pengguna = User.objects.count()
    total_tes = Tes.objects.count()
    total_jurusan = Jurusan.objects.count()
    total_soal = SoalMinat.objects.count()

    # Mengambil 5 pengguna terbaru
    pengguna_terbaru = User.objects.order_by('-date_joined')[:9]

    user = User.objects.get(username=request.user.username)

    # Menghitung jawaban per minat
    realistic_count = JawabanMinat.objects.filter(
    tes__user=user,
    jawaban='Ya',
    id_soal__minat__nama='Realistic'
    ).count()

    investigative_count = JawabanMinat.objects.filter(
        tes__user=user,
        jawaban='Ya',
        id_soal__minat__nama='Investigative'
    ).count()
```

```
    artistic_count = JawabanMinat.objects.filter(
        tes__user=user,
        jawaban='Ya',
        id_soal__minat__nama='Artistic'
    ).count()

    social_count = JawabanMinat.objects.filter(
        tes__user=user,
        jawaban='Ya',
        id_soal__minat__nama='Social'
    ).count()

    enterprising_count = JawabanMinat.objects.filter(
        tes__user=user,
        jawaban='Ya',
        id_soal__minat__nama='Enterprising'
    ).count()

    conventional_count = JawabanMinat.objects.filter(
        tes__user=user,
        jawaban='Ya',
        id_soal__minat__nama='Conventional'
    ).count()

    context = {
        'total_pengguna': total_pengguna,
        'total_tes': total_tes,
        'total_jurusan': total_jurusan,
        'total_soal': total_soal,
        'pengguna_terbaru': pengguna_terbaru,
        'user': user,
        'realistic_count': realistic_count,
        'investigative_count': investigative_count,
        'artistic_count': artistic_count,
        'social_count': social_count,
        'enterprising_count': enterprising_count,
        'conventional_count': conventional_count,
    }

    return render(request, 'dashboard.html', context)
```


4.2.5 Implementasi Halaman Profil

Implementasi halaman profil secara keseluruhan mengikuti rancangan yang telah disusun pada bagian 3.3.3.5 Desain Halaman Profil. Desain dan implementasi memiliki kesamaan hingga 100%, baik dari sisi struktur tampilan maupun fungsionalitas. Implementasi desain halaman profil pada sistem ini ditunjukkan pada Gambar 4.8 berikut.



Gambar 4.8 Implementasi Desain Halaman Profil

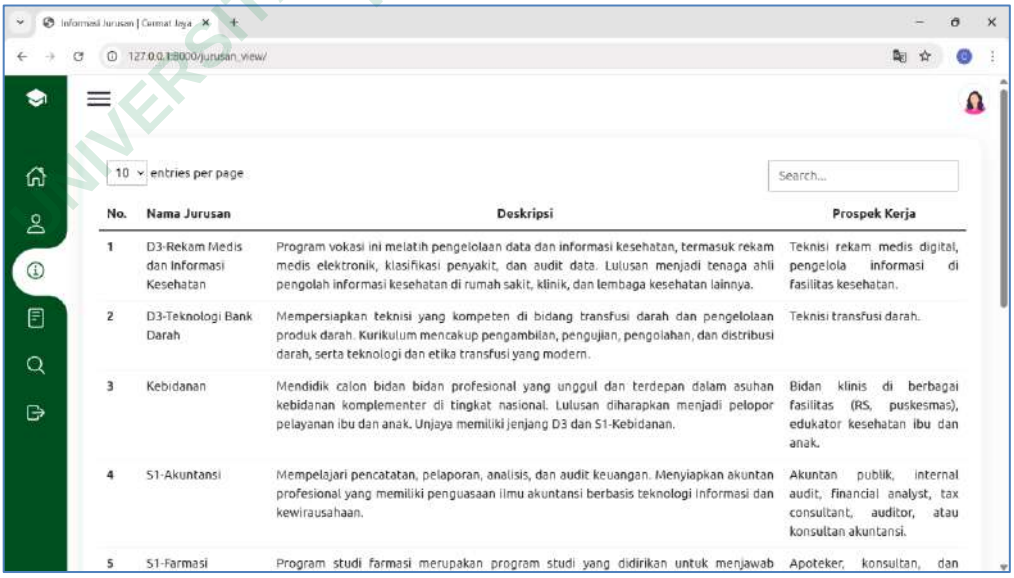
Potongan kode berikut merupakan bagian dari proses implementasi halaman profil, yang terdiri dari dua fungsi utama, yaitu *profile_view* dan *edit_profile*. Fungsi *profile_view* digunakan untuk menampilkan halaman profil kepada pengguna, sementara edit_profile berfungsi untuk memproses pembaruan data profil yang dilakukan oleh pengguna.

```
@login_required
def profile_view(request):
    user = request.user
    return render(request, 'profile.html', {'user': user})

@login_required
def edit_profile(request):
    if request.method == 'POST':
        user = request.user
        new_username = request.POST.get('username')

        # Cek apakah username sudah digunakan
        if User.objects.filter(username=new_username)
            .exclude(pk=user.pk).exists():
            # Tampilkan SweetAlert jika username sudah digunakan
            response_data = {
                'width': 600,
                'title': 'Username Tidak Tersedia',
                'text': 'Username sudah digunakan. Silahkan pilih
                         username lain.',
                'icon': 'error',
                'confirmButtonColor': "#157347",
            }
```

```
            return JsonResponse(response_data, status=400)

        user.first_name = request.POST.get('first_name')
        user.jenis_kelamin = request.POST.get('jenis_kelamin')
        user.tanggal_lahir = request.POST.get('tanggal_lahir')
        user.asal_sekolah = request.POST.get('asal_sekolah')
        user.username = request.POST.get('username')
        user.email = request.POST.get('email')
        user.no_hp = request.POST.get('no_hp')

        # Mengelola password
        password = request.POST.get('password')
        if password:
            user.set_password(password)
            # Perbarui sesi autentikasi setelah mengubah password
            update_session_auth_hash(request, user)
        user.save()

        return JsonResponse({
            'redirect_url': reverse('profile_view')
        }
```

4.2.6 Implementasi Halaman Informasi Jurusan

Implementasi halaman informasi jurusan secara keseluruhan mengikuti rancangan yang telah disusun pada bagian 3.3.3.6 Desain Halaman Informasi Jurusan. Desain dan implementasi memiliki kesamaan hingga 100%, baik dari sisi struktur tampilan maupun fungsionalitas. Implementasi desain halaman informasi jurusan pada sistem ini ditunjukkan pada Gambar 4.9 berikut.



Gambar 4.9 Implementasi Desain Halaman Informasi Jurusan

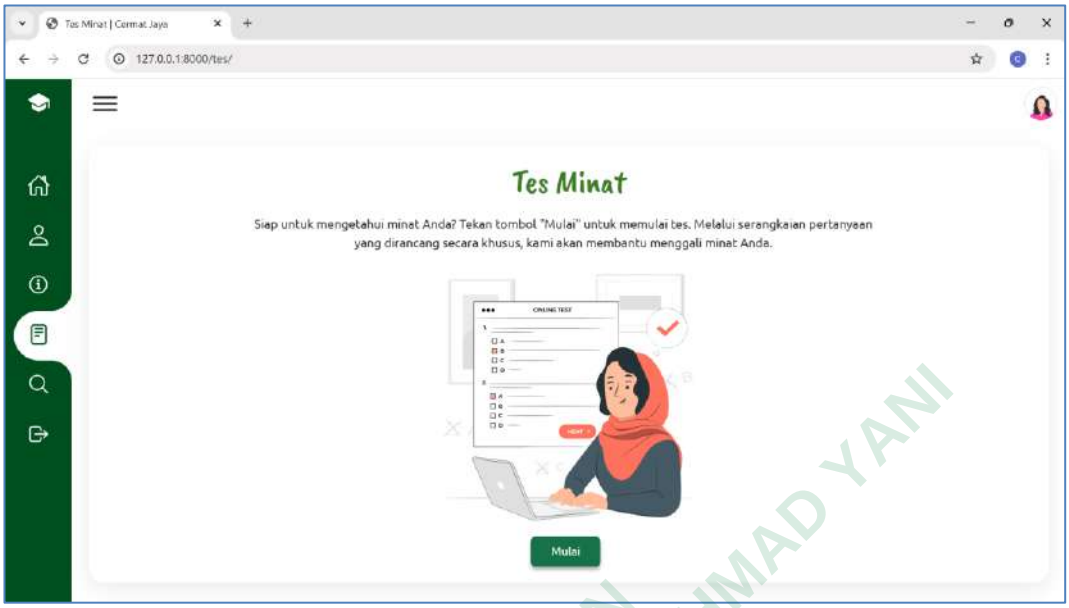
Potongan kode berikut merupakan bagian dari *file* views.py yang bertanggung jawab untuk menangani permintaan (*request*) pengguna ketika mengakses halaman informasi jurusan. Fungsi *jurusan_view* menggunakan dekorator *@login_required* yang berarti hanya pengguna yang telah *login* yang dapat mengakses halaman ini.

```
@login_required
def jurusan_view(request):
    return render(request,'jurusan.html', {
      'jurusan': Jurusan.objects.all().order_by('nama_jurusan'),
    })
```

Dalam fungsi ini, data dari tabel jurusan diambil secara keseluruhan menggunakan metode "*objects.all*()" dan diurutkan berdasarkan nama jurusan "(*order_by*('nama_jurusan'))". Data tersebut kemudian dikirim ke *template* jurusan.html untuk ditampilkan kepada pengguna. Dengan pendekatan ini, halaman informasi jurusan akan menampilkan daftar seluruh jurusan yang telah dimasukkan ke dalam sistem secara terstruktur dan terurut alfabetis.

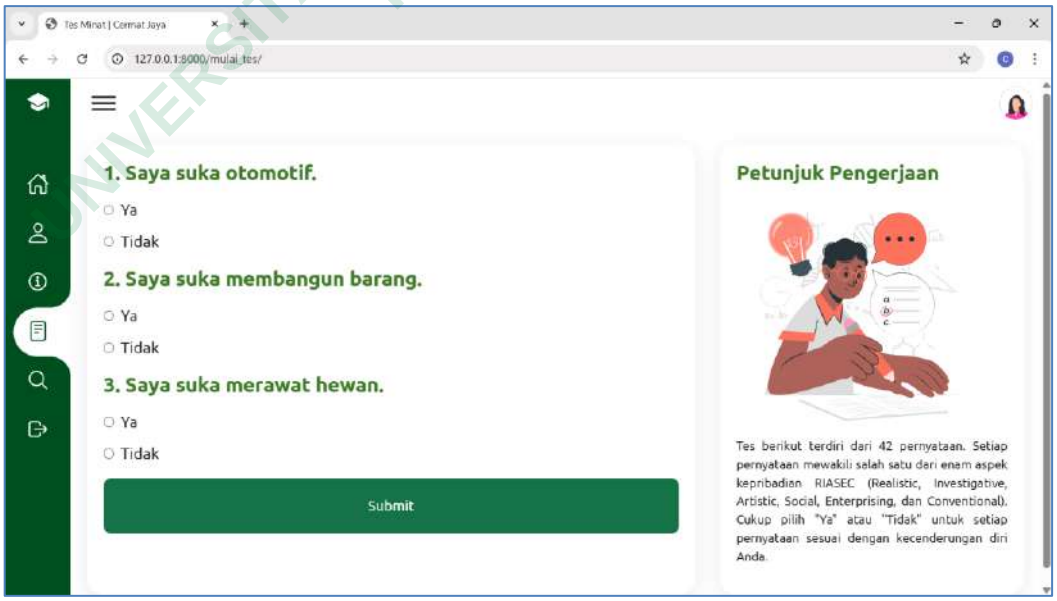
4.2.7 Implementasi Halaman Tes Minat

Implementasi halaman tes minat secara keseluruhan mengikuti rancangan yang telah disusun pada bagian 3.3.3.7 Desain Halaman Tes Minat. Implementasi halaman ini secara umum telah mengikuti desain dengan akurasi hampir 100%. Implementasi desain halaman awal tes minat pada sistem ini ditunjukkan pada Gambar 4.10 berikut.



Gambar 4.10 Implementasi Desain Halaman Tes Minat

Namun, terdapat satu perbedaan penting pada tahap implementasi dibandingkan dengan desain awal, yaitu penambahan elemen petunjuk pengerjaan pada halaman tes minat dimulai. Pada sisi kanan halaman, ditambahkan panel yang berisi ilustrasi dan petunjuk singkat mengenai jumlah pertanyaan serta cara menjawabnya. Implementasi desain halaman tes minat dimulai pada sistem ini ditunjukkan pada Gambar 4.11 berikut.



Gambar 4.11 Implementasi Desain Halaman Tes Minat Dimulai

Pada tahap implementasi, logika sistem dalam mengatur alur pengerjaan tes minat dikendalikan oleh dua fungsi utama yang ditulis pada *file* views.py, yaitu fungsi tes dan mulai_tes. Berikut adalah potongan kode yang mengatur jalannya proses pengerjaan tes minat.

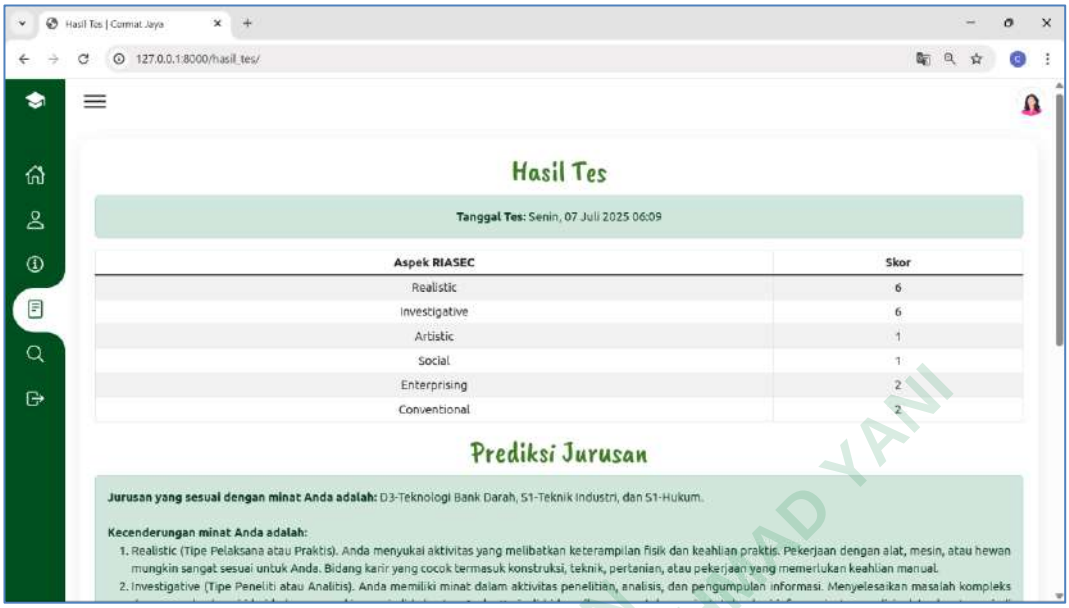
```
@login_required
def tes(request):
    return render(request, 'tes.html')

@login_required
def mulai_tes(request):
    if request.method == 'POST':
        tes_id = request.session.get('tes_id')
        if not tes_id:
            return redirect('tes')

        tes = Tes.objects.get(pk=tes_id)
        question_ids = request.POST.getlist('question_ids')

        for qid in question_ids:
            answer = request.POST.get(f'answer_{qid}')
            if answer:
                soal = SoalMinat.objects.get(pk=qid)
                JawabanMinat.objects.create(
                    tes=tes,
                    id_soal=soal,
                    jawaban=answer
                )

        # Ambil soal yang belum dijawab
        next_questions = SoalMinat.objects.exclude(
            id__in=JawabanMinat.objects.filter(tes=tes)
            .values_list('id_soal', flat=True)
        )[:3]

        if next_questions:
            answered_count = JawabanMinat.objects
                             .filter(tes=tes).count()
            context = {
                'current_questions': next_questions,
                'total_soal': SoalMinat.objects.count(),
                'start_number': answered_count + 1,
            }
            return render(request, 'mulai_tes.html', context)
        else:
            return redirect('selesai_tes')

    else:
        # Pertama kali mulai tes
        user = request.user
        tes = Tes.objects.create(user=user,
              tanggal_tes=timezone.now())
        request.session['tes_id'] = tes.id
```

```
    first_questions = SoalMinat.objects.all()[:3]
    context = {
        'current_questions': first_questions,
        'total_soal': SoalMinat.objects.count(),
        'start_number': 1,
    }
    return render(request, 'mulai_tes.html', context)
```

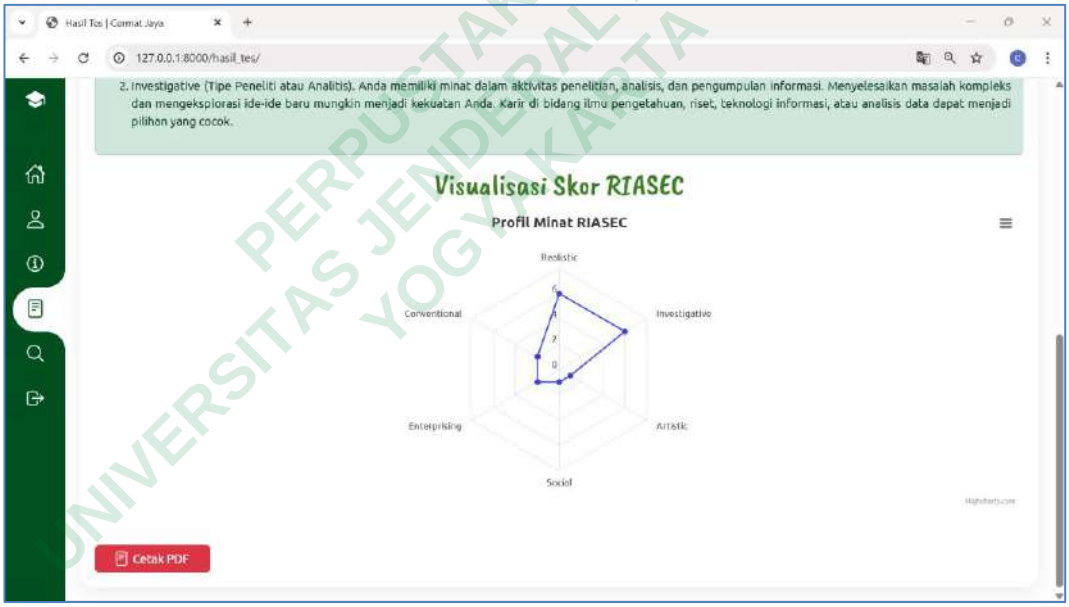
Fungsi tes bertugas untuk menampilkan halaman awal tes minat yang berisi tombol untuk memulai tes. Fungsi ini hanya merender halaman tes.html tanpa melakukan proses logika lainnya. Fungsi mulai_tes mengatur keseluruhan proses pengerjaan tes. Saat pengguna pertama kali membuka halaman tes, sistem akan secara otomatis membuat entri baru pada model "Tes" dan menyimpan id-nya di *session*. Setiap kali pengguna mengirimkan jawaban (*POST*), sistem akan mencatat jawaban untuk tiap soal dan menampilkan 3 soal berikutnya hingga seluruh soal habis.

4.2.8 Implementasi Halaman Hasil Tes

Implementasi halaman hasil tes secara keseluruhan mengikuti rancangan yang telah disusun pada bagian 3.3.3.8 Desain Halaman Hasil Tes. Implementasi halaman ini secara umum telah mengikuti desain dengan akurasi hampir 100%. Hanya terdapat penambahan grafik radar *chart* (*hexagon*) menggunakan *library* Highcharts, serta penyempurnaan tampilan visual dengan menambahkan *background* berbentuk kotak berwarna hijau (menggunakan warna dari skema Bootstrap *alert-success*) pada bagian prediksi jurusan, kecenderungan minat, dan tanggal tes. Implementasi desain halaman hasil tes pada sistem ini ditunjukkan pada Gambar 4.12 dan Gambar 4.13 berikut.



Gambar 4.12 Implementasi Desain Halaman Hasil Tes Bagian Satu



Gambar 4.13 Implementasi Desain Halaman Hasil Tes Bagian Satu

4.2.9 Implementasi Halaman Tentang Tes

Implementasi halaman tentang tes secara keseluruhan mengikuti rancangan yang telah disusun pada bagian 3.3.3.9 Desain Halaman Tentang Tes. Implementasi halaman ini secara umum telah mengikuti desain dengan akurasi hampir 100%. Hanya terdapat perbedaan pada icon yang terdapat pada bagian *sidebar*.

Implementasi desain halaman tentang tes pada sistem ini ditunjukkan pada Gambar 4.14 berikut.

Gambar 4.14 Implementasi Desain Halaman Tentang Tes

4.2.10 Implementasi Halaman *Logout*

Implementasi halaman *logout* secara keseluruhan mengikuti rancangan yang telah disusun pada bagian 3.3.3.10 Desain Halaman *Logout*. Implementasi halaman ini secara umum telah mengikuti desain dengan akurasi hampir 100%. Hanya terdapat penambahan icon yang terdapat pada bagian tombol *sign out*. Implementasi desain halaman *logout* pada sistem ini ditunjukkan pada Gambar 4.15 berikut.

Gambar 4.15 Implementasi Desain Halaman *Logout*

Adapun implementasi fungsionalitas *logout* pada sistem ini ditangani melalui dua fungsi pada *file* views.py, yaitu *logout_view* dan *logout_session*. Berikut adalah potongan kode yang digunakan.

```
@login_required
def logout_view(request):
    return render(request, 'logout.html')

def logout_session(request):
    logout(request)
    return redirect('index')
```

Fungsi *logout_view* bertanggung jawab untuk menampilkan halaman *logout*. Sementara itu, fungsi *logout_session* menangani proses penghapusan sesi pengguna yang sedang aktif menggunakan fungsi bawaan Django *logout*(*request*). Setelah proses *logout* selesai, pengguna akan langsung diarahkan (*redirect*) ke halaman utama (index) sebagai tanda bahwa sesi telah berakhir dan pengguna berhasil keluar dari sistem.

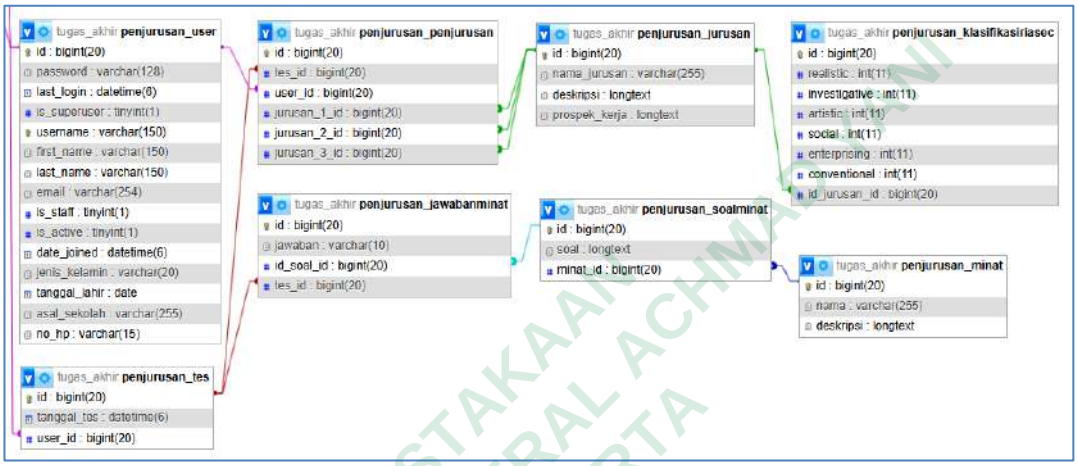
4.3 IMPLEMENTASI BASIS DATA

Implementasi basis data pada sistem ini dirancang dan dibangun dengan mengacu pada perancangan basis data yang telah dijelaskan sebelumnya pada bagian 3.3.2 Perancangan Basis Data. Seluruh entitas dan relasi antar tabel telah diimplementasikan secara konsisten dalam kode program menggunakan *framework* Django. Berikut adalah kode implementasi basis data pada sistem yang dituliskan dalam *file* models.py menggunakan Django ORM (Object-Relational Mapping).

```
from django.contrib.auth.models import AbstractUser
from django.db import models

# Create your models here.
class User(AbstractUser):
    jenis_kelamin = models.CharField(
        max_length=20,
        choices=[('Laki-laki', 'Laki-laki'),
                 ('Perempuan', 'Perempuan')]
    )
    tanggal_lahir = models.DateField()
    asal_sekolah = models.CharField(max_length=255)
    no_hp = models.CharField(max_length=15)

    REQUIRED_FIELDS = ['tanggal_lahir']

    def save(self, *args, **kwargs):
        # Enkripsi password hanya jika password berubah
        super().save(*args, **kwargs)

    def __str__(self):
        return self.username

class Minat(models.Model):
    nama = models.CharField(max_length=255)
    deskripsi = models.TextField()

    def __str__(self):
        return self.nama

class SoalMinat(models.Model):
    soal = models.TextField()
    minat = models.ForeignKey(Minat, on_delete=models.CASCADE)

    def __str__(self):
        return f"Soal {self.id}"

class Tes(models.Model):
    user = models.ForeignKey(User,
           on_delete=models.CASCADE, related_name='tes_set')
    tanggal_tes = models.DateTimeField()

    def __str__(self):
```

```
        return f"{self.user.username} - {self.tanggal_tes.date()}"

class JawabanMinat(models.Model):
    tes = models.ForeignKey(Tes, on_delete=models.CASCADE)
    id_soal = models.ForeignKey(SoalMinat,
            on_delete=models.CASCADE)
    jawaban = models.CharField(max_length=10)

    def __str__(self):
        return f"Tes {self.tes.id} - Soal {self.id_soal.id}
                        - Jawaban: {self.jawaban}"

class Jurusan(models.Model):
    nama_jurusan = models.CharField(max_length=255)
    deskripsi = models.TextField()
    prospek_kerja = models.TextField()

    def __str__(self):
        return self.nama_jurusan

class KlasifikasiRIASEC(models.Model):
    realistic = models.IntegerField()
    investigative = models.IntegerField()
    artistic = models.IntegerField()
    social = models.IntegerField()
    enterprising = models.IntegerField()
    conventional = models.IntegerField()
    id_jurusan = models.ForeignKey(Jurusan,
                 on_delete=models.CASCADE,
                 related_name='klasifikasi_riasec')

    def __str__(self):
        return f"Kriteria untuk {self.id_jurusan.nama_jurusan}"

class Penjurusan(models.Model):
    user = models.ForeignKey(User,
           on_delete=models.CASCADE,
           related_name='penjurusan_set')
    tes = models.ForeignKey(Tes, on_delete=models.CASCADE)
    jurusan_1 = models.ForeignKey(Jurusan,
                on_delete=models.CASCADE,
                related_name='jurusan_utama',
                null=True, blank=True)
    jurusan_2 = models.ForeignKey(Jurusan,
                on_delete=models.CASCADE,
                related_name='jurusan_alternatif_1',
                null=True, blank=True)
    jurusan_3 = models.ForeignKey(Jurusan,
                on_delete=models.CASCADE,
                related_name='jurusan_alternatif_2',
                null=True, blank=True)

    def tanggal_tes(self):
        return self.tes.tanggal_tes
```

Dalam pengembangan sistem ini, model pengguna (*User*) dibuat dengan mewarisi kelas *AbstractUser* dari Django. Alasan penggunaan *AbstractUser* adalah karena sistem memerlukan penambahan atribut khusus yang tidak tersedia pada model *User* bawaan Django, seperti jenis_kelamin, tanggal_lahir, asal_sekolah, dan no_hp. Hasil implementasi basis data pada sistem ini ditunjukkan pada Gambar 4.16 berikut.



Gambar 4.16 Hasil Implementasi Basis Data

4.4 IMPLEMENTASI ALGORITMA K-NEAREST NEIGHBOR (KNN)

Pada sistem ini, proses klasifikasi jurusan dilakukan dengan memanfaatkan algoritma K-Nearest Neighbor (KNN). Algoritma KNN bekerja dengan mencari sejumlah k tetangga terdekat dari data input baru (vektor RIASEC pengguna) berdasarkan jarak kemiripan dengan dataset yang sudah ada. Setiap data dalam dataset berisi enam atribut yang mewakili skor dari masing-masing aspek RIASEC, yaitu: *realistic*, *investigative*, *artistic*, *social*, *enterprising*, dan *conventional*.

Implementasi dilakukan pada fungsi hasil_tes di views.py. Vektor hasil tes pengguna dihitung berdasarkan jumlah jawaban "Ya" untuk setiap aspek minat. Data klasifikasi RIASEC yang telah tersimpan di database akan diolah menjadi dataset pembanding. Selanjutnya, vektor pengguna dibandingkan dengan data klasifikasi menggunakan algoritma KNN dari *library* scikit-learn.

Sistem menetapkan nilai k sebagai 10 atau kurang, tergantung dari jumlah data yang tersedia. Hal ini dilakukan untuk memastikan KNN tetap dapat berjalan meskipun jumlah data pelatihan terbatas. Kemudian, dari hasil prediksi tetangga

terdekat tersebut, sistem hanya mengambil tiga jurusan yang berbeda. Hal ini dilakukan dengan menyaring hasil berdasarkan nama jurusan agar tidak ada duplikasi, meskipun satu jurusan dapat memiliki lebih dari satu data vektor. Berikut adalah kode fungsi hasil_tes pada *file* views.py yang mengimplementasikan proses klasifikasi menggunakan algoritma KNN:

```
@login_required
def hasil_tes(request):
    if request.method == 'GET':
        user = request.user

        # Ambil tes terakhir user
        tes = Tes.objects.filter(user=user)
                 .order_by('-tanggal_tes').first()
        if not tes:
            return render(request, 'hasil_tes.html', {
                'error': 'Anda belum melakukan tes RIASEC.',
            })

        # Hitung jumlah jawaban "Ya" untuk setiap minat
        skor_minat = {}
        for minat in Minat.objects.all():
            skor = JawabanMinat.objects.filter(
                tes=tes,
                id_soal__minat=minat,
                jawaban='Ya'
            ).count()
            skor_minat[minat.nama.lower()] = skor

        # Pastikan semua kategori RIASEC terisi, walaupun 0
        all_keys = ['realistic', 'investigative', 'artistic',
                    'social', 'enterprising', 'conventional']
        for k in all_keys:
            skor_minat.setdefault(k, 0)

        print(skor_minat)  # Print untuk debugging

        # Buat vektor RIASEC
        riasec_vector = np.array([
            skor_minat['realistic'],
            skor_minat['investigative'],
            skor_minat['artistic'],
            skor_minat['social'],
            skor_minat['enterprising'],
            skor_minat['conventional'],
        ])

        # Ambil data klasifikasi dari database
        klasifikasi_data = KlasifikasiRIASEC.objects.all()
        if not klasifikasi_data.exists():
            return render(request, 'hasil_tes.html', {
              'error': 'Data klasifikasi RIASEC belum tersedia.',
```

```
        })

    X = []
    y = []
    for data in klasifikasi_data:
        X.append([
            data.realistic,
            data.investigative,
            data.artistic,
            data.social,
            data.enterprising,
            data.conventional
        ])
        y.append(data.id_jurusan.nama_jurusan)

    # Jalankan KNN
    k = min(10, len(X))
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X, y)

    # Ambil indeks tetangga terdekat
    _, indices = knn.kneighbors([riasec_vector])

    # Kumpulkan jurusan unik dari hasil tetangga
    jurusan_terdekat_unik = []
    jurusan_nama_set = set()

    for i in indices[0]:
        nama_jurusan = y[i]
        if nama_jurusan not in jurusan_nama_set:
            jurusan_obj = Jurusan.objects.filter(
                nama_jurusan=nama_jurusan
            ).first()
            if jurusan_obj:
                jurusan_terdekat_unik.append(jurusan_obj)
                jurusan_nama_set.add(nama_jurusan)
            if len(jurusan_terdekat_unik) == 3:
                break

    # Validasi apakah kita dapat 3 jurusan berbeda
    if len(jurusan_terdekat_unik) < 3:
        return render(request, 'hasil_tes.html', {
            'error': 'Tidak ditemukan cukup jurusan berbeda
                      dari hasil prediksi.',
        })

    # Simpan hasil penjurusan
    Penjurusan.objects.create(
        user=user,
        tes=tes,
        jurusan_1=jurusan_terdekat_unik[0],
        jurusan_2=jurusan_terdekat_unik[1],
        jurusan_3=jurusan_terdekat_unik[2]
    )

    # Cari minat dengan skor tertinggi
```

```
        max_score = max(skor_minat.values())

        minat_tertinggi = []
        for minat in Minat.objects.all():
            nama = minat.nama.lower()
            if skor_minat.get(nama, 0) == max_score:
                minat_tertinggi.append(minat)

        return render(request, 'hasil_tes.html', {
            'tes': tes,
            'riasec_scores': skor_minat,
            'jurusan_1': jurusan_terdekat_unik[0],
            'jurusan_2': jurusan_terdekat_unik[1],
            'jurusan_3': jurusan_terdekat_unik[2],
            'minat_tertinggi': minat_tertinggi,
        })
```

4.5 HASIL KUESIONER RIASEC

Penyebaran kuesioner RIASEC dilakukan pada tanggal 17 Juni sampai dengan 2 Juli 2025 (16 hari). Kuesioner ini bertujuan untuk mengumpulkan data minat dari mahasiswa Unjaya sebagai dasar dalam menyusun dataset klasifikasi minat jurusan berbasis teori kepribadian RIASEC. Kuesioner terdiri dari 42 pernyataan, dimana masing-masing aspek RIASEC terdiri dari 7 pernyataan. Setiap responden diminta menjawab dengan pilihan "Ya" atau "Tidak", dan skor tiap aspek dihitung dari total jawaban "Ya" pada aspek tersebut. Selain itu, kuesioner juga memuat satu pertanyaan validasi yaitu "Apakah Anda merasa salah jurusan?", untuk mendapatkan gambaran umum mengenai ketepatan pilihan jurusan yang saat ini dijalani oleh responden. Dari proses pengumpulan data, diperoleh total 82 responden yang mengisi kuesioner secara lengkap dan valid. Hasil kuesioner RIASEC ditunjukkan pada Tabel 4.1 berikut.

Tabel 4.1 Hasil Kuesioner RIASEC

No.	Program Studi	R	I	A	S	E	C	Validasi	Minat
1.	Akuntansi	5	4	4	6	5	7	Tidak	C dan S
2.	Akuntansi	1	6	1	1	1	7	Tidak	C dan I
3.	Akuntansi	2	6	1	1	2	7	Tidak	C dan I
4.	Akuntansi	3	5	2	3	2	7	Tidak	C dan I
5.	Farmasi	2	7	1	2	2	6	Tidak	I dan C
6.	Farmasi	3	7	2	3	1	5	Tidak	I dan C

7.	Farmasi	2	6	2	2	1	6	Tidak	I dan C
8.	Farmasi	1	6	1	3	1	6	Tidak	I dan C
9.	Farmasi	2	6	2	1	2	6	Tidak	I dan C
10.	Farmasi	4	5	2	7	3	6	Tidak	S dan C
11.	Hukum	2	7	1	1	6	2	Tidak	I dan E
12.	Hukum	3	6	1	3	5	2	Tidak	I dan E
13.	Hukum	4	5	2	7	4	4	Tidak	S dan I
14.	Informatika	5	3	5	4	5	6	Ya	C dan (R atau A atau E)
15.	Informatika	3	4	3	7	7	3	Tidak	S dan E
16.	Informatika	5	7	1	4	6	6	Tidak	I dan (E atau C)
17.	Informatika	4	2	3	4	4	5	Tidak	C dan (R atau S atau E)
18.	Informatika	3	3	4	2	5	6	Tidak	C dan E
19.	Informatika	4	6	1	6	6	7	Tidak	C dan (I atau S atau E)
20.	Informatika	5	3	6	4	6	3	Ya	A dan E
21.	Informatika	2	7	4	4	2	6	Tidak	I dan C
22.	Informatika	5	6	2	5	3	7	Tidak	C dan I
23.	Informatika	4	3	3	6	3	7	Tidak	C dan S
24.	Informatika	6	4	4	6	5	6	Tidak	(R atau S atau C) dan E
25.	Informatika	5	6	4	7	2	7	Ya	S dan C
26.	Informatika	6	7	3	6	5	4	Tidak	I dan (R atau S)
27.	Informatika	4	5	5	5	3	4	Ya	(I atau A atau S) dan (R atau C)
28.	Informatika	4	4	3	5	5	6	Tidak	C dan (S dan E)

29.	Informatika	2	6	1	3	2	5	Tidak	I dan C
30.	Informatika	3	7	1	4	1	6	Tidak	I dan C
31.	Informatika	3	6	1	2	1	5	Tidak	I dan C
32.	Informatika	2	6	2	4	2	5	Tidak	I dan C
33.	Informatika	3	5	2	2	2	4	Tidak	I dan C
34.	Kebidanan	5	6	2	7	4	5	Ya	S dan I
35.	Kebidanan	4	5	1	7	6	6	Ya	S dan (E atau C)
36.	Kebidanan	6	2	2	6	3	1	Tidak	R dan S
37.	Kebidanan	6	1	1	7	1	1	Tidak	S dan R
38.	Kebidanan	5	3	1	6	2	1	Tidak	S dan R
39.	Kebidanan	7	2	2	7	1	1	Tidak	R dan S
40.	Kebidanan	6	3	2	6	2	1	Tidak	R dan S
41.	Keperawatan	2	3	4	7	4	4	Tidak	S dan (A atau E atau C)
42.	Keperawatan	5	3	2	6	3	1	Tidak	S dan R
43.	Keperawatan	6	2	2	7	3	1	Tidak	S dan R
44.	Keperawatan	5	2	1	7	1	1	Tidak	S dan R
45.	Keperawatan	6	3	3	7	2	2	Tidak	S dan R
46.	Keperawatan	5	3	2	7	3	3	Tidak	S dan R
47.	Manajemen	2	3	2	2	7	6	Tidak	E dan C
48.	Manajemen	3	3	1	2	7	6	Tidak	E dan C
49.	Manajemen	2	3	2	3	6	5	Tidak	E dan C
50.	Psikologi	3	2	5	6	0	7	Tidak	C dan S
51.	Psikologi	4	3	4	7	2	3	Tidak	S dan (R atau A)
52.	Psikologi	5	6	5	4	1	7	Tidak	C dan I
53.	Psikologi	2	6	3	7	1	3	Tidak	S dan I
54.	Psikologi	1	6	3	7	1	1	Tidak	S dan I
55.	Psikologi	1	5	1	6	3	1	Tidak	S dan I

56.	Rekam Medis dan Informasi Kesehatan	2	6	1	2	2	7	Tidak	C dan I
57.	Rekam Medis dan Informasi Kesehatan	2	5	1	3	1	7	Tidak	C dan I
58.	Rekam Medis dan Informasi Kesehatan	2	5	1	1	1	7	Tidak	C dan I
59.	Rekam Medis dan Informasi Kesehatan	3	5	2	2	1	7	Tidak	C dan I
60.	Rekam Medis dan Informasi Kesehatan	2	6	1	3	1	7	Tidak	C dan I
61.	Rekam Medis dan Informasi Kesehatan	4	5	2	7	1	7	Tidak	S dan C
62.	Sistem Informasi	7	6	7	6	6	7	Ya	(R atau A atau C) dan (I atau S atau E)
63.	Sistem Informasi	4	3	2	5	3	6	Tidak	C dan S
64.	Sistem Informasi	4	6	2	7	5	7	Tidak	S dan C
65.	Sistem Informasi	2	4	2	1	6	7	Tidak	C dan E
66.	Sistem Informasi	1	3	2	2	6	7	Tidak	C dan E
67.	Teknik Industri	7	6	4	6	4	6	Tidak	R dan (I atau S atau C)
68.	Teknik Industri	4	7	3	7	2	7	Tidak	(I atau S atau C) dan R
69.	Teknik Industri	5	7	3	7	2	7	Tidak	(I atau S atau C) dan R

70.	Teknik Industri	7	3	1	1	1	6	Tidak	R dan C
71.	Teknik Industri	7	3	2	1	2	5	Tidak	R dan C
72.	Teknik Industri	6	3	1	1	2	5	Tidak	R dan C
73.	Teknologi Bank Darah	6	6	1	1	1	2	Tidak	R dan I
74.	Teknologi Bank Darah	5	6	2	2	1	3	Tidak	I dan R
75.	Teknologi Bank Darah	6	6	1	1	2	2	Tidak	R dan I
76.	Teknologi Bank Darah	5	7	2	2	1	2	Tidak	I dan R
77.	Teknologi Bank Darah	6	6	1	3	2	3	Tidak	R dan I
78.	Teknologi Bank Darah	6	5	5	7	4	7	Tidak	S dan C
79.	Teknologi Informasi	3	7	4	7	5	7	Ya	(I atau S atau C) dan E
80.	Teknologi Informasi	3	6	2	2	1	6	Tidak	I dan C
81.	Teknologi Informasi	2	6	1	1	2	7	Tidak	C dan I
82.	Teknologi Informasi	3	5	1	3	2	7	Tidak	C dan I

Tabel 4.1 menyajikan hasil pengolahan data dari kuesioner RIASEC yang telah disebarkan kepada mahasiswa Unjaya. Data yang ditampilkan dalam tabel ini mencakup beberapa informasi penting, antara lain jurusan yang saat ini diambil oleh masing-masing responden, serta skor masing-masing aspek RIASEC yang terdiri dari *realistic* (R), *investigative* (I), *artistic* (A), *social* (S), *enterprising* (E), dan *conventional* (C). Selain itu, tabel ini juga memuat hasil validasi berupa tanggapan responden terhadap pertanyaan "Apakah Anda merasa salah jurusan?", yang bertujuan untuk mengetahui kesesuaian antara minat dan jurusan yang

diambil. Di bagian akhir tabel, ditampilkan dua minat tertinggi dari masing-masing responden yang diperoleh berdasarkan perbandingan skor keenam aspek RIASEC. Dari hasil kuesioner tersebut dilakukan analisis untuk mengetahui minat dominan pada tiap-tiap jurusan. Hasil analisis tersebut ditunjukkan pada Tabel 4.2 berikut.

Tabel 4.2 Analisis Minat Jurusan

No.	Program Studi	Minat Dominan
1.	Akuntansi	C dan I
2.	Farmasi	I dan C
3.	Hukum	I dan E
4.	Informatika	I dan C
5.	Kebidanan	R dan S
6.	Keperawatan	S dan R
7.	Manajemen	E dan C
8.	Psikologi	S dan I
9.	Rekam Medis dan Informasi Kesehatan	C dan I
10.	Sistem Informasi	C dan E
11.	Teknik Industri	R dan C
12.	Teknologi Bank Darah	R dan I
13.	Teknologi Informasi	C dan I

Tabel 4.2 menunjukan minat dominan dari masing-masing jurusan. Nantinya data yang diambil dari setiap jurusan untuk keperluan *dataset* harus data yang memiliki minat yang sesuai dengan tabel tersebut. Dikarenakan jurusan hukum hanya memiliki 2 data yang sesuai dengan minat yang ada di Tabel 4.2, maka untuk keseimbangan *dataset*, setiap jurusan hanya akan diambil 2 data. Data hasil kuesioner yang akan dijadikan sebagai *dataset* ditunjukkan pada Tabel 4.3 berikut.

Tabel 4.3 *Dataset* Klasifikasi RIASEC

No.	Program Studi	R	I	A	S	E	C	Validasi	Minat
1.	Akuntansi	2	6	1	1	2	7	Tidak	C dan I
2.	Akuntansi	3	5	2	3	2	7	Tidak	C dan I
3.	Farmasi	3	7	2	3	1	5	Tidak	I dan C
4.	Farmasi	1	6	1	3	1	6	Tidak	I dan C
5.	Hukum	2	7	1	1	6	2	Tidak	I dan E
6.	Hukum	3	6	1	3	5	2	Tidak	I dan E
7.	Informatika	2	7	4	4	2	6	Tidak	I dan C
8.	Informatika	3	7	1	4	1	6	Tidak	I dan C
9.	Kebidanan	7	2	2	7	1	1	Tidak	R dan S
10.	Kebidanan	6	3	2	6	2	1	Tidak	R dan S
11.	Keperawatan	5	3	2	6	3	1	Tidak	S dan R
12.	Keperawatan	6	2	2	7	3	1	Tidak	S dan R
13.	Manajemen	3	3	1	2	7	6	Tidak	E dan C
14.	Manajemen	2	3	2	3	6	5	Tidak	E dan C
15.	Psikologi	2	6	3	7	1	3	Tidak	S dan I
16.	Psikologi	1	6	3	7	1	1	Tidak	S dan I
17.	Rekam Medis dan Informasi Kesehatan	2	5	1	3	1	7	Tidak	C dan I
18.	Rekam Medis dan Informasi Kesehatan	2	6	1	3	1	7	Tidak	C dan I
19.	Sistem Informasi	2	4	2	1	6	7	Tidak	C dan E
20.	Sistem Informasi	1	3	2	2	6	7	Tidak	C dan E
21.	Teknik Industri	7	3	1	1	1	6	Tidak	R dan C
22.	Teknik Industri	7	3	2	1	2	5	Tidak	R dan C
23.	Teknologi Bank Darah	6	6	1	1	2	2	Tidak	R dan I

24.	Teknologi Bank Darah	6	6	1	3	2	3	Tidak	R dan I
25.	Teknologi Informasi	2	6	1	1	2	7	Tidak	C dan I
26.	Teknologi Informasi	3	5	1	3	2	7	Tidak	C dan I

4.6 PENGUJIAN SISTEM

Setelah proses implementasi sistem selesai dilakukan, tahap selanjutnya adalah melakukan pengujian untuk memastikan bahwa sistem berjalan dengan baik dan seluruh fungsionalitas yang dirancang telah berjalan sebagaimana mestinya. Pengujian ini bertujuan untuk mengidentifikasi adanya kesalahan (*bug*), ketidaksesuaian sistem terhadap kebutuhan pengguna, serta menguji kinerja dari algoritma yang diterapkan dalam sistem. Pengujian dilakukan melalui dua pendekatan utama, yaitu pengujian *black box testing* dan pengujian akurasi algoritma KNN.

4.6.1 Pengujian *Black Box Testing*

Metode *black box testing* digunakan untuk menguji fungsi-fungsi utama sistem berdasarkan input dan *output* yang diharapkan, tanpa memperhatikan struktur internal kode program. Pengujian dilakukan dengan cara memberikan berbagai skenario input pada fitur-fitur yang tersedia, lalu memeriksa apakah hasil *output* yang ditampilkan oleh sistem sesuai dengan yang diharapkan. Beberapa fitur utama yang diuji antara lain proses *login*, *custom error page*, pendaftaran pengguna, edit profil pengguna, pencarian jurusan, pengecekan jawaban tes RIASEC, serta cetak hasil tes dalam format pdf. Hasil pengujian dari setiap fitur dicatat dalam bentuk tabel untuk menunjukkan status apakah fitur tersebut berhasil dijalankan (lulus) atau tidak.

4.6.1.1 Pengujian Fitur *Login*

Pengujian fitur *login* dapat dilihat pada Tabel 4.4 berikut.

Tabel 4.4 Pengujian Fitur *Login*

Skenario	*Output* yang Diharapkan	*Output* Sistem	Kesimpulan
User menginputkan *username* dan *password* yang benar.	Berhasil masuk ke halaman *dashboard.*	Berhasil masuk ke halaman *dashboard.*	Sesuai
User menginputkan *username* yang salah dan *password* yang benar.	Sistem menampilkan pesan "*Username* atau *Password* salah".	Sistem menampilkan pesan "*Username* atau *Password* salah".	Sesuai
User menginputkan *username* yang benar dan *password* yang salah.	Sistem menampilkan pesan "*Username* atau *Password* salah".	Sistem menampilkan pesan "*Username* atau *Password* salah".	Sesuai
User menginputkan *username* yang salah dan *password* yang salah.	Sistem menampilkan pesan "*Username* atau *Password* salah".	Sistem menampilkan pesan "*Username* atau *Password* salah".	Sesuai

4.6.1.2 Pengujian Fitur *Custom Error Page*

Pengujian fitur *custom error page* dapat dilihat pada Tabel 4.5 berikut.

Tabel 4.5 Pengujian Fitur *Custom Error Page*

Skenario	*Output* yang Diharapkan	*Output* Sistem	Kesimpulan
User mencoba mengakses halaman *dashboard* tanpa *login* terlebih dahulu, dengan cara menambahkan "/*dashboard*" pada URL sistem.	Sistem menampilkan halaman *custom error page*.	Sistem menampilkan halaman *custom error page.*	Sesuai

User menginputkan URL yang salah pada sistem.	Sistem menampilkan halaman *custom error page*.	Sistem menampilkan halaman *custom error page*.	Sesuai

4.6.1.3 Pengujian Fitur Pendaftaran

Pengujian fitur pendaftaran dapat dilihat pada Tabel 4.6 berikut.

Tabel 4.6 Pengujian Fitur Pendaftaran

Skenario	*Output* yang Diharapkan	*Output* Sistem	Kesimpulan
User melakukan pendaftaran dengan menginputkan *username* yang telah digunakan.	Sistem menampilkan pesan kesalahan "*Username* Tidak Tersedia".	Sistem menampilkan pesan kesalahan "*Username* Tidak Tersedia".	Sesuai
User melakukan pendaftaran dengan menginputkan email dengan format yang salah.	Sistem menampilkan pesan kesalahan format email.	Sistem menampilkan pesan kesalahan format email.	Sesuai
User melakukan pendaftaran dengan mengosongkan *field* tertentu.	Sistem menampilkan pesan wajib melengkapi *field* tersebut.	Sistem menampilkan pesan wajib melengkapi *field* tersebut.	Sesuai
User melakukan pendaftaran dengan menginputkan semua data dengan format yang sesuai.	Sistem menampilkan pesan "Pendaftaran Berhasil" dan mengarahkan pengguna untuk melakukan *login*.	Sistem menampilkan pesan "Pendaftaran Berhasil" dan mengarahkan pengguna untuk melakukan *login*.	Sesuai

4.6.1.4 Pengujian Fitur Edit Profil

Pengujian fitur edit profil dapat dilihat pada Tabel 4.7 berikut.

Tabel 4.7 Pengujian Fitur Edit Profil

Skenario	*Output* yang Diharapkan	*Output* Sistem	Kesimpulan
User melakukan edit profil, kemudian mengklik tombol edit.	Sistem menampilkan *pop-up* konfirmasi "Apakah Anda yakin ingin menyimpan perubahan?". Pada *pop-up* tersebut terdapat pilihan "Ya, Simpan!" dan "*Cancel*".	Sistem menampilkan *pop-up* konfirmasi "Apakah Anda yakin ingin menyimpan perubahan?". Pada *pop-up* tersebut terdapat pilihan "Ya, Simpan!" dan "*Cancel*".	Sesuai
User melakukan edit profil, kemudian mengklik tombol edit, lalu memilih pilihan "Ya, Simpan!" pada *pop-up* konfirmasi.	Sistem menampilkan pesan "Perubahan berhasil disimpan" dan melakukan *refresh* halaman.	Sistem menampilkan pesan "Perubahan berhasil disimpan" dan melakukan *refresh* halaman.	Sesuai
User melakukan edit profil, kemudian mengklik tombol edit, lalu memilih pilihan "*Cancel*" pada *pop-up* konfirmasi.	Sistem menampilkan pesan "Perubahan yang Anda lakukan tidak disimpan".	Sistem menampilkan pesan "Perubahan yang Anda lakukan tidak disimpan".	Sesuai
User melakukan edit profil dengan mengganti *username* lama dengan *username* baru yang sudah digunakan oleh orang lain, kemudian mengklik tombol edit.	Sistem menampilkan pesan "*Username* Tidak Tersedia".	Sistem menampilkan pesan "*Username* Tidak Tersedia".	Sesuai

4.6.1.5 Pengujian Fitur Pencarian Jurusan

Pengujian fitur pencarian jurusan pada tabel jurusan di halaman informasi jurusan dapat dilihat pada Tabel 4.8 berikut.

Tabel 4.8 Pengujian Fitur Pencarian Jurusan

Skenario	*Output* yang Diharapkan	*Output* Sistem	Kesimpulan
User melakukan pencarian dengan menginputkan nama jurusan yang ingin dicari pada *field* pencarian.	Sistem menampilkan jurusan yang dicari.	Sistem menampilkan jurusan yang dicari.	Sesuai

4.6.1.6 Pengujian Fitur Pengecekan Jawaban

Pengujian fitur pengecekan jawaban pada saat melakukan tes dapat dilihat pada Tabel 4.9 berikut.

Tabel 4.9 Pengujian Fitur Pengecekan Jawaban

Skenario	*Output* yang Diharapkan	*Output* Sistem	Kesimpulan
User tidak mengisi semua jawaban, kemudian mengklik tombol *submit*.	Sistem menampilkan pesan “Pastikan semua pertanyaan sudah dijawab!”.	Sistem menampilkan pesan “Pastikan semua pertanyaan sudah dijawab!”.	Sesuai
User mengisi semua jawaban, kemudian mengklik tombol *submit*.	Sistem menampilkan pertanyaan berikutnya.	Sistem menampilkan pertanyaan berikutnya.	Sesuai

4.6.1.7 Pengujian Fitur Cetak PDF

Pengujian fitur cetak pdf pada halaman hasil tes dapat dilihat pada Tabel 4.10 berikut.

Tabel 4.10 Pengujian Fitur Cetak PDF

Skenario	*Output* yang Diharapkan	*Output* Sistem	Kesimpulan
User mengklik tombol cetak pdf.	Sistem memproses cetak pdf hasil tes.	Sistem memproses cetak pdf hasil tes.	Sesuai

4.6.2 Pengujian Akurasi

Pengujian akurasi dilakukan dengan membandingkan hasil rekomendasi sistem dengan data yang telah terlabel dalam *dataset*. *Dataset* ini telah disajikan sebelumnya pada Tabel 4.3 Dataset Klasifikasi RIASEC, yang memuat data vektor dari hasil kuesioner sebanyak 26 entri. Masing-masing entri dalam tabel tersebut mencantumkan total skor enam aspek RIASEC (Realistic, Investigative, Artistic, Social, Enterprising, dan Conventional) dan label jurusan aktual dari responden.

Untuk memastikan sistem memberikan rekomendasi yang tepat, dilakukan simulasi pengujian dengan cara memilih beberapa data vektor dalam Tabel 4.3, lalu memasukkannya ke dalam sistem sebagai data uji. Hasil rekomendasi dari sistem, yaitu tiga jurusan yang ditampilkan, kemudian dibandingkan dengan hasil perhitungan manual menggunakan metode Euclidean Distance seperti yang telah dijelaskan pada bagian 2.2.2.2. Dengan menghitung secara manual jarak antara vektor uji dan seluruh vektor dalam dataset klasifikasi, maka akan diketahui urutan kedekatan setiap data berdasarkan nilai jarak terkecil.

Jika sistem bekerja dengan benar, maka jurusan pertama (jurusan ke-1) yang direkomendasikan sistem seharusnya identik dengan jurusan dari data yang diuji (karena data tersebut ada dalam dataset dan merupakan jarak 0 terhadap dirinya sendiri). Sementara jurusan kedua dan ketiga seharusnya merupakan jurusan dari dua data lain yang memiliki nilai kedekatan terendah berikutnya terhadap data uji.

Dengan demikian, pengujian ini tidak hanya melihat apakah jurusan pertama cocok dengan label, namun juga mengevaluasi apakah tiga jurusan teratas yang direkomendasikan sistem sesuai dengan tiga jarak terdekat dari hasil perhitungan manual menggunakan Euclidean Distance. Hal ini membantu memastikan bahwa implementasi algoritma KNN pada sistem benar-benar melakukan klasifikasi berdasarkan kedekatan vektor yang valid.

4.6.2.1 Pengujian Pertama

Pengujian pertama dilakukan dengan menggunakan data nomor 14 pada Tabel 4.3, yaitu data vektor [2, 3, 2, 3, 6, 5] yang merepresentasikan program studi manajemen. Vektor ini dimasukkan sebagai data uji ke dalam sistem untuk

mengamati apakah algoritma KNN berhasil mengembalikan jurusan yang sesuai berdasarkan kedekatan jarak terhadap data lainnya. Jarak antar vektor dihitung menggunakan metode Euclidean Distance seperti yang telah dijelaskan pada bagian 2.2.2.2. Jarak ini menunjukkan tingkat kemiripan antara data uji dengan setiap data lainnya dalam dataset. Semakin kecil nilai jarak, semakin mirip data tersebut dengan data uji. Tabel 4.11 berikut ini adalah tabel kedekatan jarak Euclidean antara data uji dan seluruh entri dalam dataset.

Tabel 4.11 Tabel Jarak Euclidean Pengujian Pertama

No.	Program Studi	R	I	A	S	E	C	Jarak Euclidean
14.	Manajemen	2	3	2	3	6	5	0,000
13.	Manajemen	3	3	1	2	7	6	2,236
20.	Sistem Informasi	1	3	2	2	6	7	2,449
19.	Sistem Informasi	2	4	2	1	6	7	3,000
6.	Hukum	3	6	1	3	5	2	4,583
2.	Akuntansi	3	5	2	3	2	7	5,000
26.	Teknologi Informasi	3	5	1	3	2	7	5,099
5.	Hukum	2	7	1	1	6	2	5,477
17.	Rekam Medis dan Informasi Kesehatan	2	5	1	3	1	7	5,831
25.	Teknologi Informasi	2	6	1	1	2	7	5,831
1.	Akuntansi	2	6	1	1	2	7	5,831
4.	Farmasi	1	6	1	3	1	6	6,083
7.	Informatika	2	7	4	4	2	6	6,164
18.	Rekam Medis dan Informasi Kesehatan	2	6	1	3	1	7	6,245
3.	Farmasi	3	7	2	3	1	5	6,481
11.	Keperawatan	5	3	2	6	3	1	6,557
22.	Teknik Industri	7	3	2	1	2	5	6,708

8.	Informatika	3	7	1	4	1	6	6,708
24.	Teknologi Bank Darah	6	6	1	3	2	3	6,782
15.	Psikologi	2	6	3	7	1	3	7,416
23.	Teknologi Bank Darah	6	6	1	1	2	2	7,416
21.	Teknik Industri	7	3	1	1	1	6	7,483
10.	Kebidanan	6	3	2	6	2	1	7,550
12.	Keperawatan	6	2	2	7	3	1	7,616
16.	Psikologi	1	6	3	7	1	1	8,246
9.	Kebidanan	7	2	2	7	1	1	9,110

Dari Tabel 4.11 tersebut, dapat dilihat bahwa 3 vektor terdekat adalah data nomor 14, 13, dan 20. Dari hasil tersebut maka data jurusan terdekat secara berurutan adalah manajemen, manajemen, dan sistem informasi. Akan tetapi, setelah jarak dihitung, sistem tidak hanya mengambil tiga data terdekat, tetapi juga memfilter agar jurusan yang diambil adalah jurusan yang berbeda. Hal ini bertujuan agar sistem tidak memberikan rekomendasi yang redundan (misalnya dua kali jurusan yang sama), tetapi benar-benar menyajikan tiga jurusan alternatif terbaik yang berbeda berdasarkan tingkat kedekatan vektor terhadap hasil tes pengguna.

Dengan pendekatan ini, dapat dipastikan bahwa jurusan pertama yang direkomendasikan sistem memiliki jarak paling dekat dengan hasil tes pengguna, sedangkan jurusan kedua dan ketiga adalah jurusan dari data klasifikasi lain yang juga memiliki tingkat kemiripan yang tinggi. Oleh karena itu, hasil 3 jurusan terdekat secara berturut-turut menjadi manajemen, sistem informasi, dan hukum. Hal ini sesuai dengan hasil yang diberikan sistem pada Gambar 4.17 berikut.

Gambar 4.17 Hasil Pengujian Pertama

4.6.2.2 Pengujian Kedua

Pengujian kedua dilakukan dengan menggunakan data nomor 18 pada Tabel 4.3, yaitu data vektor [2, 6, 1, 3, 1, 7] yang merepresentasikan program studi rekam medis dan informasi kesehatan. Vektor ini dimasukkan sebagai data uji ke dalam sistem untuk mengamati apakah algoritma KNN berhasil mengembalikan jurusan yang sesuai berdasarkan kedekatan jarak terhadap data lainnya. Jarak antar vektor dihitung menggunakan metode Euclidean Distance seperti yang telah dijelaskan pada bagian 2.2.2.2. Jarak ini menunjukkan tingkat kemiripan antara data uji dengan setiap data lainnya dalam dataset. Semakin kecil nilai jarak, semakin mirip data tersebut dengan data uji. Tabel 4.12 berikut ini adalah tabel kedekatan jarak Euclidean antara data uji dan seluruh entri dalam dataset.

Tabel 4.12 Tabel Jarak Euclidean Pengujian Kedua

No.	Program Studi	R	I	A	S	E	C	Jarak Euclidean
18.	Rekam Medis dan Informasi Kesehatan	2	6	1	3	1	7	0,000
17.	Rekam Medis dan Informasi Kesehatan	2	5	1	3	1	7	1,000

4.	Farmasi	1	6	1	3	1	6	1,414
26.	Teknologi Informasi	3	5	1	3	2	7	1,732
2.	Akuntansi	3	5	2	3	2	7	2,000
8.	Informatika	3	7	1	4	1	6	2,000
25.	Teknologi Informasi	2	6	1	1	2	7	2,236
1.	Akuntansi	2	6	1	1	2	7	2,236
3.	Farmasi	3	7	2	3	1	5	2,646
7.	Informatika	2	7	4	4	2	6	3,606
24.	Teknologi Bank Darah	6	6	1	3	2	3	5,745
19.	Sistem Informasi	2	4	2	1	6	7	5,831
15.	Psikologi	2	6	3	7	1	3	6,000
20.	Sistem Informasi	1	3	2	2	6	7	6,083
14.	Manajemen	2	3	2	3	6	5	6,245
21.	Teknik Industri	7	3	1	1	1	6	6,245
6.	Hukum	3	6	1	3	5	2	6,481
22.	Teknik Industri	7	3	2	1	2	5	6,633
23.	Teknologi Bank Darah	6	6	1	1	2	2	6,782
13.	Manajemen	3	3	1	2	7	6	6,928
5.	Hukum	2	7	1	1	6	2	7,416
16.	Psikologi	1	6	3	7	1	1	7,550
11.	Keperawatan	5	3	2	6	3	1	8,246
10.	Kebidanan	6	3	2	6	2	1	8,485
12.	Keperawatan	6	2	2	7	3	1	9,434
9.	Kebidanan	7	2	2	7	1	1	9.695

Dari Tabel 4.12 tersebut, dapat dilihat bahwa 3 vektor terdekat adalah data nomor 18, 17, dan 4. Dari hasil tersebut maka data jurusan terdekat secara berurutan adalah rekam medis dan informasi kesehatan, rekam medis dan informasi kesehatan, dan farmasi. Akan tetapi, setelah jarak dihitung, sistem tidak hanya

mengambil tiga data terdekat, tetapi juga memfilter agar jurusan yang diambil adalah jurusan yang berbeda. Hal ini bertujuan agar sistem tidak memberikan rekomendasi yang redundan (misalnya dua kali jurusan yang sama), tetapi benar-benar menyajikan tiga jurusan alternatif terbaik yang berbeda berdasarkan tingkat kedekatan vektor terhadap hasil tes pengguna.

Dengan pendekatan ini, dapat dipastikan bahwa jurusan pertama yang direkomendasikan sistem memiliki jarak paling dekat dengan hasil tes pengguna, sedangkan jurusan kedua dan ketiga adalah jurusan dari data klasifikasi lain yang juga memiliki tingkat kemiripan yang tinggi. Oleh karena itu, hasil 3 jurusan terdekat secara berturut-turut menjadi rekam medis dan informasi kesehatan, farmasi, dan teknologi informasi. Hal ini sesuai dengan hasil yang diberikan sistem pada Gambar 4.18 berikut.

Gambar 4.18 Hasil Pengujian Kedua

4.6.2.3 Pengujian Ketiga

Pengujian ketiga dilakukan dengan menggunakan data nomor 23 pada Tabel 4.3, yaitu data vektor [6, 6, 1, 1, 2, 2] yang merepresentasikan program studi manajemen. Vektor ini dimasukkan sebagai data uji ke dalam sistem untuk mengamati apakah algoritma KNN berhasil mengembalikan jurusan yang sesuai berdasarkan kedekatan jarak terhadap data lainnya. Jarak antar vektor dihitung

menggunakan metode Euclidean Distance seperti yang telah dijelaskan pada bagian 2.2.2.2. Jarak ini menunjukkan tingkat kemiripan antara data uji dengan setiap data lainnya dalam dataset. Semakin kecil nilai jarak, semakin mirip data tersebut dengan data uji. Tabel 4.13 berikut ini adalah tabel kedekatan jarak Euclidean antara data uji dan seluruh entri dalam dataset.

Tabel 4.13 Tabel Jarak Euclidean Pengujian Ketiga

No.	Program Studi	R	I	A	S	E	C	Jarak Euclidean
23.	Teknologi Bank Darah	6	6	1	1	2	2	0,000
24.	Teknologi Bank Darah	6	6	1	3	2	3	2,236
22.	Teknik Industri	7	3	2	1	2	5	4,472
6.	Hukum	3	6	1	3	5	2	4,690
3.	Farmasi	3	7	2	3	1	5	5,000
21.	Teknik Industri	7	3	1	1	1	6	5,196
5.	Hukum	2	7	1	1	6	2	5,745
8.	Informatika	3	7	1	4	1	6	6,000
10.	Kebidanan	6	3	2	6	2	1	6,000
11.	Keperawatan	5	3	2	6	3	1	6,164
26.	Teknologi Informasi	3	5	1	3	2	7	6,245
2.	Akuntansi	3	5	2	3	2	7	6,325
1.	Akuntansi	2	6	1	1	2	7	6,403
25.	Teknologi Informasi	2	6	1	1	2	7	6,403
18.	Rekam Medis dan Informasi Kesehatan	2	6	1	3	1	7	6,782
4.	Farmasi	1	6	1	3	1	6	6,782
17.	Rekam Medis dan Informasi Kesehatan	2	5	1	3	1	7	6,856
7.	Informatika	2	7	4	4	2	6	7,141

14.	Manajemen	2	3	2	3	6	5	7,416
12.	Keperawatan	6	2	2	7	3	1	7,416
9.	Kebidanan	7	2	2	7	1	1	7,483
15.	Psikologi	2	6	3	7	1	3	7,616
13.	Manajemen	3	3	1	2	7	6	7,746
19.	Sistem Informasi	2	4	2	1	6	7	7,874
16.	Psikologi	1	6	3	7	1	1	8,185
20.	Sistem Informasi	1	3	2	2	6	7	8,775

Dari Tabel 4.13 tersebut, dapat dilihat bahwa 3 vektor terdekat adalah data nomor 23, 24, dan 22. Dari hasil tersebut maka data jurusan terdekat secara berurutan adalah teknologi bank darah, teknologi bank darah, dan teknik industri. Akan tetapi, setelah jarak dihitung, sistem tidak hanya mengambil tiga data terdekat, tetapi juga memfilter agar jurusan yang diambil adalah jurusan yang berbeda. Hal ini bertujuan agar sistem tidak memberikan rekomendasi yang redundan (misalnya dua kali jurusan yang sama), tetapi benar-benar menyajikan tiga jurusan alternatif terbaik yang berbeda berdasarkan tingkat kedekatan vektor terhadap hasil tes pengguna.

Dengan pendekatan ini, dapat dipastikan bahwa jurusan pertama yang direkomendasikan sistem memiliki jarak paling dekat dengan hasil tes pengguna, sedangkan jurusan kedua dan ketiga adalah jurusan dari data klasifikasi lain yang juga memiliki tingkat kemiripan yang tinggi. Oleh karena itu, hasil 3 jurusan terdekat secara berturut-turut menjadi teknologi bank darah, teknik industri, dan hukum. Hal ini sesuai dengan hasil yang diberikan sistem pada Gambar 4.19 berikut.

Gambar 4.19 Hasil Pengujian Ketiga

4.7 PEMBAHASAN

Bab ini membahas hasil pengujian sistem berdasarkan pertanyaan penelitian yang telah dirumuskan. Pembahasan difokuskan pada efektivitas sistem dalam membantu pemilihan jurusan berbasis minat dan kepribadian, mekanisme kerja integrasi RIASEC-KNN, serta akurasi sistem dalam memberikan rekomendasi jurusan.

4.7.1 Sistem Berbasis Web dalam Membantu Pemilihan Jurusan

Sistem pendukung keputusan yang dikembangkan berhasil memberikan solusi berbasis web yang interaktif dan mudah diakses untuk membantu calon mahasiswa Unjaya dalam menentukan jurusan yang sesuai. Melalui antarmuka yang sederhana, pengguna dapat:

1. Melihat informasi jurusan yang ada di Unjaya, berupa deskripsi dan prospek kerja dari masing-masing jurusan yang tersedia.
2. Melakukan tes minat berbasis teori kepribadian RIASEC.
3. Melihat hasil klasifikasi skor RIASEC secara visual dalam bentuk tabel maupun grafik radar.

4. Mendapatkan rekomendasi jurusan berdasarkan hasil tes. Penyajian rekomendasi jurusan dilengkapi dengan informasi deskriptif mengenai jurusan dan prospek kerjanya, serta kecenderungan minat pengguna.
5. Mengunduh hasil tes dalam format pdf sebagai arsip pribadi.

Keberadaan sistem ini dapat mengurangi ketergantungan pada intuisi pribadi atau informasi umum dalam pemilihan jurusan. Dengan menyajikan rekomendasi berdasarkan data, sistem membantu calon mahasiswa membuat keputusan yang lebih tepat dan relevan dengan karakteristik mereka.

4.7.2 Integrasi Hasil Tes RIASEC dan Algoritma KNN

Integrasi antara hasil tes RIASEC dan algoritma K-Nearest Neighbor (KNN) merupakan inti dari mekanisme sistem. Prosesnya sebagai berikut:

1. Tes RIASEC menghasilkan skor numerik untuk masing-masing aspek kepribadian RIASEC: *realistic*, *investigative*, *artistic*, *social*, *enterprising*, dan *conventional*. Skor ini dihitung dari total jawaban "Ya" pada soal terkait aspek kepribadian tertentu.
2. Skor tersebut digunakan sebagai vektor fitur (6 dimensi) yang dibandingkan dengan *dataset* mahasiswa Unjaya yang telah memiliki label jurusan.
3. Algoritma KNN menghitung kedekatan (*similarity*) menggunakan metode Euclidean Distance untuk menemukan k tetangga terdekat.
4. Rekomendasi jurusan ditentukan berdasarkan label jurusan dari tetangga terdekat.

Proses ini menunjukkan bahwa sistem tidak hanya memberikan hasil secara acak, namun berdasarkan kedekatan data numerik yang terstruktur dengan metode klasifikasi yang teruji.

4.7.3 Akurasi Rekomendasi Algoritma KNN

Berdasarkan pengujian akurasi yang dilakukan menggunakan data uji dari dataset yang sama, sistem berhasil memberikan rekomendasi yang sesuai, terutama pada jurusan pertama yang selalu identik dengan data asal (karena jarak nol terhadap dirinya sendiri). Untuk jurusan kedua dan ketiga, sistem menunjukkan akurasi yang baik, karena hasil rekomendasi sesuai dengan data jurusan yang

memiliki jarak kedekatan paling kecil berdasarkan hasil perhitungan manual menggunakan Euclidean Distance. Ini membuktikan bahwa implementasi algoritma KNN dalam sistem sudah berjalan sesuai dengan konsep dasar dan logika klasifikasi berbasis kedekatan. Tingkat akurasi yang dicapai menunjukkan bahwa sistem dapat diandalkan untuk memberikan saran jurusan yang valid dan relevan.