BAB 4
HASIL PENELITIAN

4.1 RINGKASAN HASIL PENELITIAN

Penelitian ini menghasilkan sebuah aplikasi *e-voting* berbasis web yang dibangun menggunakan *framework* Django dan basis data MySQL. Sistem ini dikembangkan untuk mendukung proses Pemilihan Umum Raya (Pemira) di Universitas Jenderal Achmad Yani Yogyakarta, dengan tujuan meningkatkan keamanan, efisiensi, serta transparansi pemilihan. Proses pengembangan aplikasi mengikuti metode Waterfall, yang terdiri dari tahapan analisis kebutuhan, desain sistem, implementasi, pengujian, dan evaluasi.

Aplikasi yang dihasilkan memiliki fitur utama seperti autentikasi berbasis *token one-time password* (OTP), manajemen data pemilih dan kandidat, pengiriman token melalui email, rekapitulasi hasil suara secara *real-time*, serta fitur deteksi kecurangan yang mampu memonitor aktivitas anomali, seperti percobaan *voting* ganda atau akses tidak wajar. Sistem ini juga menerapkan audit log untuk setiap aktivitas penting, serta enkripsi data penting guna menjaga integritas dan kerahasiaan informasi.

Hasil pengujian sistem menunjukkan bahwa seluruh fitur utama berjalan sesuai fungsinya, baik pada pengujian unit, integrasi, maupun pengujian penerimaan pengguna (UAT). Pengujian dilakukan dengan melibatkan admin dan sejumlah mahasiswa sebagai pemilih simulasi, dan hasilnya menunjukkan bahwa sistem dapat mengelola data secara efektif, mencegah tindakan kecurangan, serta memudahkan proses penghitungan suara. Dengan demikian, sistem yang dibangun dapat dikatakan layak untuk diimplementasikan pada pelaksanaan Pemira di Unjaya secara digital dan aman.

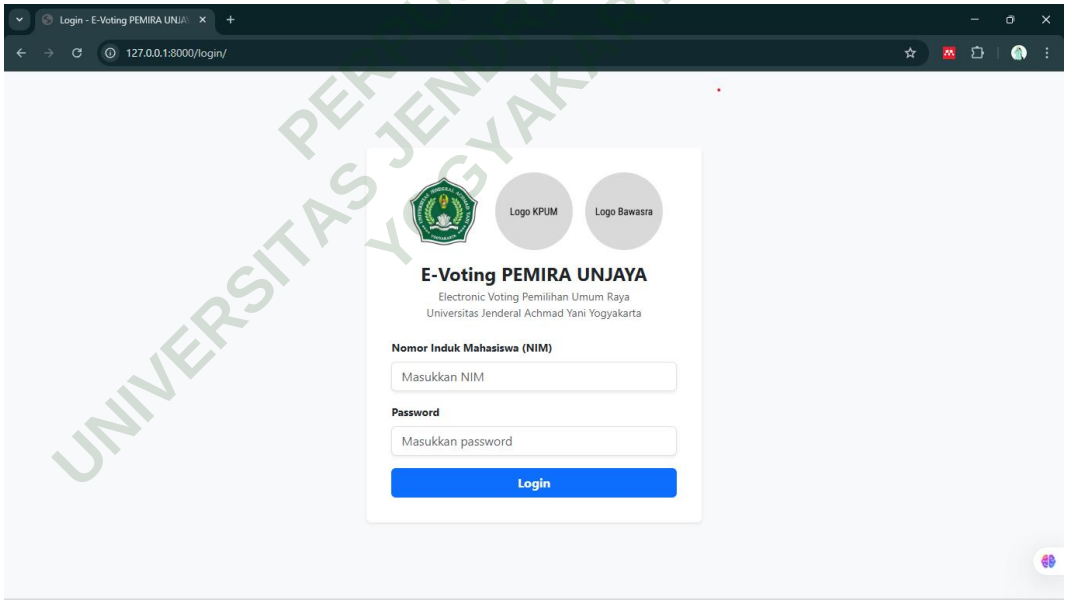
4.2 IMPLEMENTASI DESAIN ANTARMUKA

Pada tahap implementasi desain antarmuka, sistem *e-voting* Pemira Unjaya dikembangkan dengan mengacu pada prinsip kemudahan penggunaan (*usability*) dan aksesibilitas bagi seluruh pengguna, baik admin maupun pemilih dan dengan

keamanan yang terjamin. Setiap halaman dan fitur dirancang agar intuitif, responsif, serta mudah dipahami, sehingga proses pemilihan dapat berjalan lancar tanpa hambatan teknis. Berikut merupakan implementasi dari halaman penting yang terdapat pada aplikasi ini.

4.2.1 Halaman *Login User* (Admin dan *Voter*)

Saat pertama kali sistem di akses halaman login adalah halaman yang pertama kali muncul. Pada halaman ini, pengguna baik admin maupun pemilih harus memasukkan Nomor Induk Mahasiswa (NIM) serta *password* untuk dapat mengakses sistem. Selain kolom isian login, pada bagian atas juga ditampilkan logo institusi dan panitia sebagai bentuk identitas resmi aplikasi. Desain antarmuka dibuat sederhana dan mudah dipahami agar seluruh pengguna dapat mengakses sistem dengan nyaman dan tanpa kendala. Gambar 4.1 menunjukan implementasi antarmuka login user (admin/*voter*).



Gambar 4.1 Halaman Login User

Potongan kode program untuk proses login user yang diimplementasikan dalam sistem ini dapat dilihat sebagai berikut.

```
class RedirectToLoginView(View):
  def get(self, request):
    return redirect('main:login')
```

```
class LoginView(View):
  def get(self, request):
    if request.user.is_authenticated:
      if request.user.role == 'admin':
        return redirect('admin:dashboard_admin')
      elif request.user.role == 'voter':
        return redirect('voters:home')
    return render(request, 'main/login.html')

  def post(self, request):
    nim = request.POST.get('nim')
    password = request.POST.get('password')
    user = authenticate(request, nim=nim, password=password)

    if user is not None:
      login(request, user)
      request.session.set_expiry(18000)  # 5 jam

      request.session['nim'] = user.nim
      request.session['role'] = user.role
      request.session['full_name'] = user.full_name

      if user.role == 'admin':
        return redirect('admin:dashboard_admin')
      elif user.role == 'voter':
        return redirect('voters:home')
    else:
      messages.error(request, 'NIM atau Password salah')

    return render(request, 'main/login.html')
```

Kode LoginView di atas menangani proses login user pada aplikasi. Saat user mengakses halaman login, sistem akan mengecek apakah user sudah login atau belum; jika sudah, user langsung diarahkan ke dashboard sesuai perannya (admin atau *voter*). Saat data login dikirimkan, sistem akan melakukan autentikasi berdasarkan NIM dan *password*. Jika valid, user akan masuk ke sistem dan diarahkan sesuai peran, jika tidak, akan muncul pesan error. Proses ini memastikan hanya user terdaftar yang dapat mengakses fitur utama aplikasi. Dibawah ini juga terdapat kode program models dari *Account* untuk penyimpanan data user di*database*.

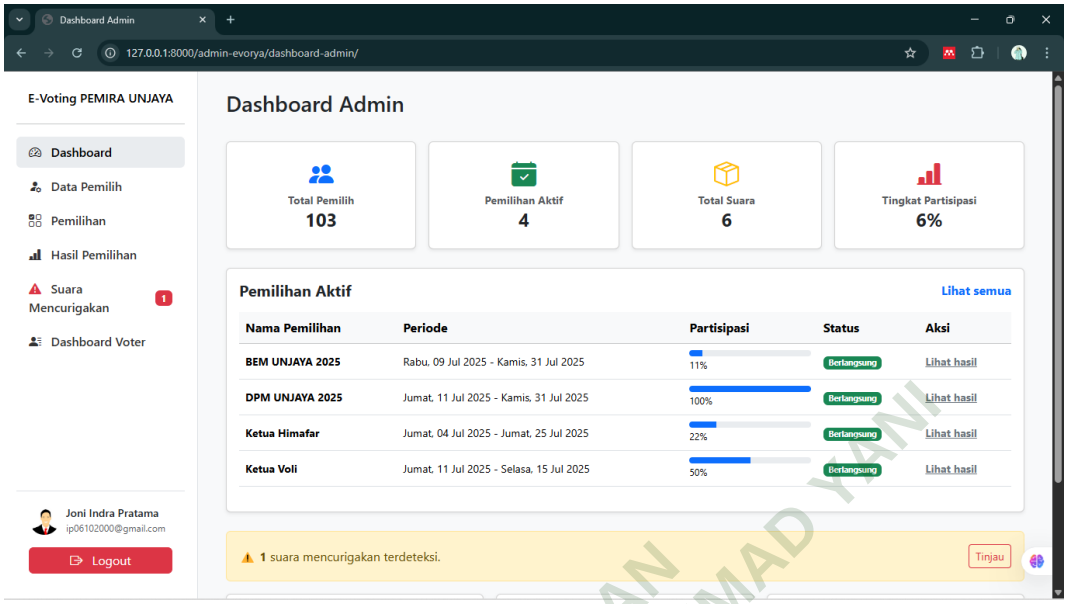
```
class Account(AbstractUser):
  nim = models.CharField(max_length=20, primary_key=True,
        unique=True)
  prodi = models.ForeignKey(Prodi, on_delete=models.SET_NULL,
          null=True, blank=True)
  email = models.EmailField(unique=True)
```

```
  photo_profile = models.ImageField(
    upload_to='profile/images/photo_profile/',
    null=True,
    blank=True,
    default='profile/images/photo_profile/default.png'
  )

  ROLE_CHOICES = (
    ('admin', 'Admin'),
    ('voter', 'Voter'),
  )
  role = models.CharField(max_length=10, choices=ROLE_CHOICES,
         default='voter')

  username = None
  USERNAME_FIELD = 'nim'
  REQUIRED_FIELDS = ['first_name', 'last_name', 'email', 'role']

  objects = AccountManager()

  def __str__(self):
    return f"{self.nim} - {self.full_name}"

  @property
  def full_name(self):
    return f"{self.first_name} {self.last_name}"
```

Model *Account* pada kode di atas berfungsi sebagai representasi data pengguna dalam sistem *e-voting*, baik sebagai admin maupun *voter*. Model ini menyimpan atribut seperti NIM (sebagai username dan primary key), nama lengkap, email, foto profil, serta relasi ke program studi. Selain itu, model ini juga membedakan peran user melalui atribut role. Dengan struktur ini, sistem dapat mengelola autentikasi dan identitas pengguna secara terpusat dan efisien.

4.2.2 Halaman Dashboard Admin

Halaman dashboard admin merupakan tampilan utama yang muncul setelah admin berhasil login ke dalam sistem. Pada halaman ini, admin dapat melihat ringkasan data seperti total pemilih, jumlah pemilihan aktif, total suara masuk, dan persentase tingkat partisipasi. Selain itu, terdapat daftar pemilihan aktif lengkap dengan informasi periode, status, partisipasi, serta fitur akses cepat untuk melihat hasil. Notifikasi suara mencurigakan juga ditampilkan agar admin dapat segera menindaklanjuti. Gambar 4.2 menunjukan implementasi antarmuka dashboard admin.



Gambar 4.2 Halaman Dashboard Admin

Cuplikan kode program yang mengatur logika tampilan dan data pada dashboard admin dapat dilihat sebagai berikut.

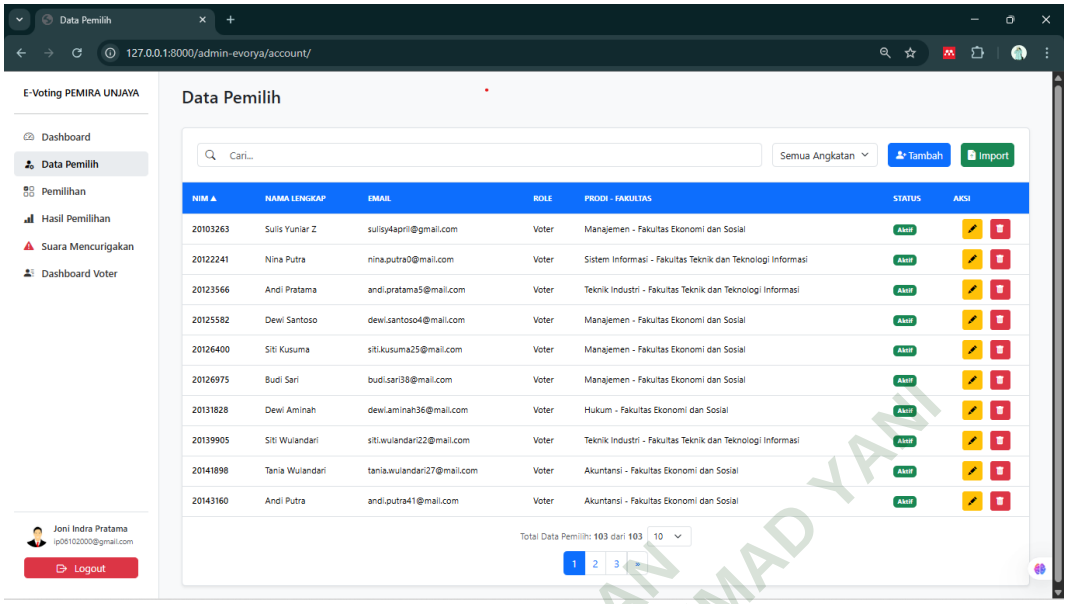
```
class DashboardAdminView(View):
  def get(self, request):
    user = request.user
    total_suara_mencurigakan = Suara.objects.filter
                               (is_suspicious=True).count()
    rekomendasi_ganti_password = False
    if not request.session.get('password_checked_admin'):
      if user.check_password(user.nim):
        rekomendasi_ganti_password = True
      request.session['password_checked_admin'] = True

    now = timezone.localtime(timezone.now())
    semua_pemilihan = Pemilihan.objects.all()
    total_pemilih = Account.objects.filter(is_superuser=False)
                    .count()
    total_suara = Suara.objects.count()
    partisipasi_total = (total_suara / total_pemilih * 100) if
                        total_pemilih > 0 else 0

    pemilihan_aktif_qs = semua_pemilihan.filter(
      waktu_mulai__lte=now,
      waktu_selesai__gte=now
    )
    pemilihan_aktif = []
    for p in pemilihan_aktif_qs:
      jumlah_pemilih = p.pemilih.count()
      jumlah_suara = p.suara_list.count()
      partisipasi = (jumlah_suara / jumlah_pemilih * 100) if
                    jumlah_pemilih > 0 else 0
```

```
        pemilihan_aktif.append({
          'id': p.id,
          'nama': p.nama,
          'waktu_mulai': p.waktu_mulai,
          'waktu_selesai': p.waktu_selesai,
          'partisipasi': round(partisipasi),
        })

      context = {
        'total_pemilih': total_pemilih,
        'jumlah_pemilihan_aktif': pemilihan_aktif_qs.count(),
        'total_suara': total_suara,
        'partisipasi_total': round(partisipasi_total),
        'pemilihan_aktif': pemilihan_aktif,
        'rekomendasi_ganti_password': rekomendasi_ganti_password,
        'total_kecurangan': total_suara_mencurigakan,
      }
      return render(request, 'admin/dashboard_admin.html',
                    context)
```

Potongan kode di atas merupakan implementasi dari tampilan dashboard admin pada aplikasi *e-voting*. View ini bertugas untuk menampilkan data rekapitulasi, seperti total pemilih, total suara, jumlah pemilihan aktif, tingkat partisipasi, dan jumlah suara mencurigakan. Selain itu, view ini juga menyediakan data pemilihan yang sedang aktif beserta tingkat partisipasinya. Kode ini juga memberikan rekomendasi ganti *password* jika admin masih menggunakan *password* default. Semua data tersebut kemudian dikirim ke template untuk ditampilkan pada halaman dashboard admin.

4.2.3 Halaman Data Pemilih

Pada halaman ini, admin dapat melihat daftar seluruh pemilih yang terdaftar beserta informasi detail seperti NIM, nama lengkap, email, role, program studi, fakultas, serta status keaktifan. Terdapat fitur pencarian, filter berdasarkan angkatan, serta tombol untuk menambah, mengedit, dan menghapus data pemilih. Selain itu, admin juga dapat melakukan impor data pemilih secara massal. Halaman ini memudahkan pengelolaan data pemilih secara efisien dan terstruktur. Gambar 4.3 menunjukan implementasi antarmuka data pemilih pada sistem *e-voting*.



Gambar 4.3 Halaman Data Pemilih

Cuplikan baris kode program yang digunakan untuk proses menampilkan data pemilih pada sistem ini dapat dilihat pada kode berikut.

```
class UserListView(View):
  def get(self, request):
    search_query = request.GET.get('q', '')
    sort_by = request.GET.get('sort', 'nim')
    direction = request.GET.get('dir', 'asc')
    selected_angkatan = request.GET.get('angkatan', '')
    order = sort_by if direction == 'asc' else f'-{sort_by}'

    users = Account.objects.exclude(is_superuser=1)
            .select_related('prodi__fakultas')
    if search_query:
      users = users.filter(
        Q(nim__icontains=search_query) |
        Q(first_name__icontains=search_query) |
        Q(last_name__icontains=search_query) |
        Q(email__icontains=search_query) |
        Q(role__icontains=search_query) |
        Q(prodi__nama__icontains=search_query) |
        Q(prodi__fakultas__nama__icontains=search_query)
      )
    if selected_angkatan:
      users = users.filter(nim__startswith=selected_angkatan)
    users = users.order_by(order)
    per_page = request.GET.get('per_page', '10')
    paginator = Paginator(users, int(per_page))
    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)
    angkatan_list = (
      Account.objects.exclude(is_superuser=1)
```

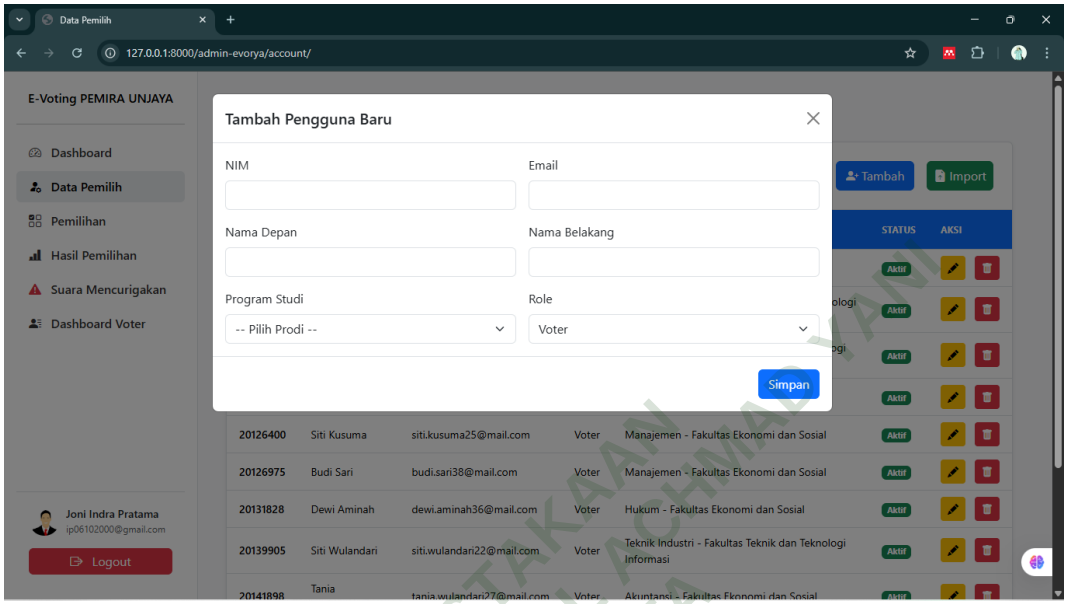
```
    .values_list('nim', flat=True)
  )
  angkatan_list = sorted(set(nim[:2] for nim in angkatan_list
                  if len(nim) >= 2))
  context = {
    'page_obj': page_obj,
    'search_query': search_query,
    'sort_by': sort_by,
    'direction': direction,
    'field_labels': [
      ('nim', 'NIM'),
      ('first_name', 'Nama Lengkap'),
      ('email', 'Email'),
      ('role', 'Role'),
      ('prodi__nama', 'Prodi - Fakultas'),
      ('is_active', 'Status')
    ],
    "prodis": Prodi.objects.select_related("fakultas").all(),
    "angkatan_list": angkatan_list,
    "selected_angkatan": selected_angkatan,
    "per_page": per_page,
    "total_users": users.count(),
    "total_all_users": Account.objects.exclude(is_superuser=1)
                       .count(),
  }

  if request.headers.get('x-requested-with') ==
  'XMLHttpRequest':
    return render(request, 'admin/partials/user_table.html',
                  context)
  return render(request, 'admin/user_list.html', context)
```

Potongan kode di atas merupakan implementasi **UserListView** untuk menampilkan daftar data pemilih pada halaman admin. Fungsi ini menangani pencarian, filter berdasarkan angkatan, pengurutan data, dan paginasi daftar user yang bukan superuser. Admin dapat mencari user berdasarkan NIM, nama, email, role, prodi, atau fakultas, serta mengurutkan data sesuai kebutuhan. Data juga bisa difilter berdasarkan angkatan tertentu. Hasil akhirnya akan ditampilkan dalam bentuk tabel yang bisa diperbarui secara dinamis menggunakan AJAX.

4.2.4 Tambah Data Pemilih

Tampilan tambah pengguna baru memungkinkan admin untuk memasukkan data user baru ke dalam sistem, seperti NIM, email, nama depan, nama belakang, program studi, dan role. Seluruh *field* harus diisi sebelum menekan tombol “Simpan” agar data pengguna baru dapat tersimpan. Fitur ini mempercepat proses

input dan registrasi pemilih di aplikasi. Gambar 4.4 menunjukan implementasi antarmuka tambah pengguna baru.



Gambar 4.4 Tambah Data Pemilih

Cuplikan baris kode program yang digunakan untuk proses menambahkan data pemilih pada sistem ini dapat dilihat pada kode berikut.

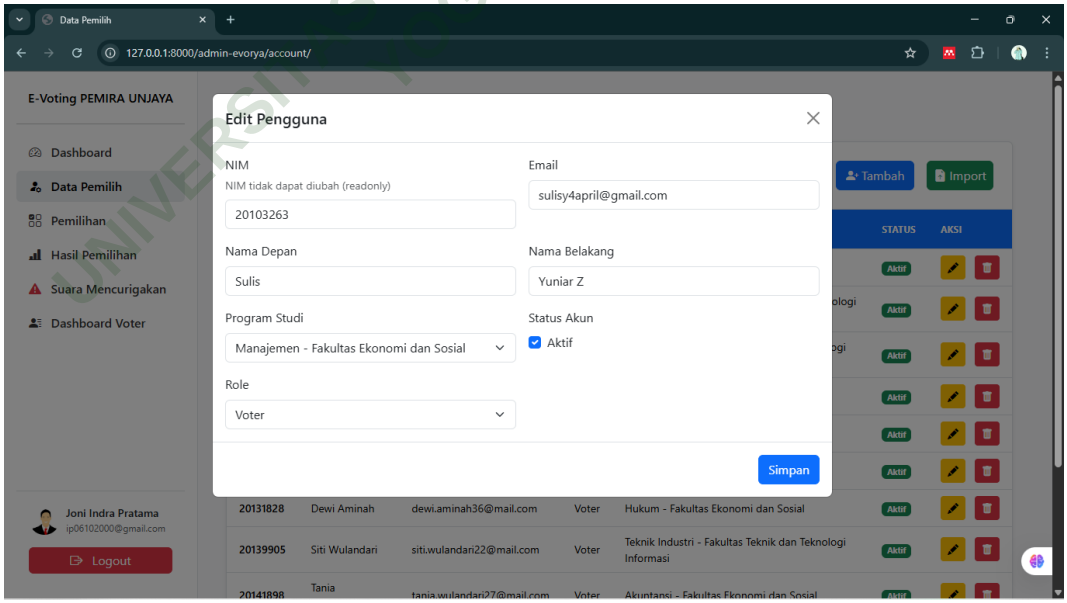
```
class AddUserView(View):
  def post(self, request):
    nim = request.POST.get('nim')
    email = request.POST.get('email')
    password = request.POST.get('password')
    first_name = request.POST.get('first_name')
    last_name = request.POST.get('last_name')
    role = request.POST.get('role')
    prodi_id = request.POST.get('prodi')

    if Account.objects.filter(nim=nim).exists():
      messages.error(request, f"NIM '{nim}' sudah terdaftar.
                Tidak dapat menambahkan pengguna yang sama.")
      return redirect('admin:user_list')

    try:
      Account.objects.create_user(
        nim=nim,
        email=email,
        password=password,
        first_name=first_name,
        last_name=last_name,
        role=role,
        prodi_id=prodi_id
```

```
        )
        messages.success(request, "User berhasil ditambahkan.")
    except Exception as e:
        messages.error(request, f"Gagal menambahkan user: {e}")
    return redirect('admin:user_list')
```

Potongan kode di atas berfungsi untuk menangani proses penambahan data pengguna baru ke dalam sistem. Pada bagian ini, data yang diinputkan oleh admin melalui form seperti NIM, email, nama, *password*, peran (role), dan program studi akan diambil dari request dan selanjutnya dilakukan pengecekan apakah NIM yang dimasukkan sudah terdaftar sebelumnya. Jika belum ada, maka data user baru akan disimpan ke *database*. Jika terjadi kesalahan, sistem akan menampilkan pesan error yang sesuai.

4.2.5 Edit User

Tampilan edit pengguna memungkinkan admin memperbarui data user seperti nama depan, nama belakang, email, program studi, status akun, dan role user. Field NIM bersifat readonly. Perubahan disimpan melalui tombol “Simpan”. Fitur ini membantu admin memastikan keakuratan data pemilih. Gambar 4.5 menunjukkan implementasi antarmuka edit data pengguna.



Gambar 4.5 Edit Data Pemilih

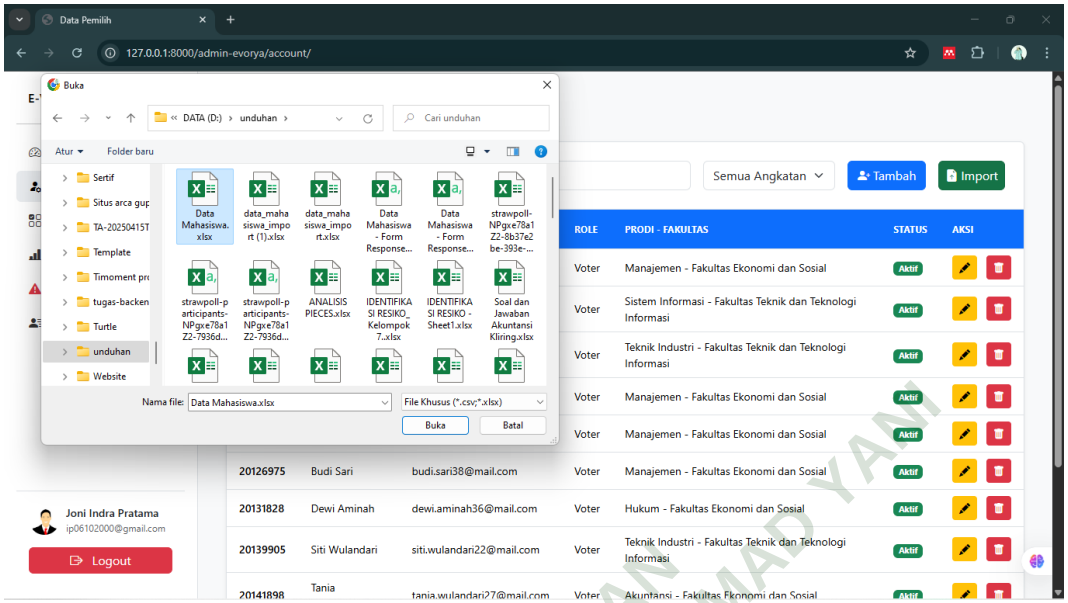
Cuplikan baris kode program yang digunakan untuk proses mengubah data pemilih pada sistem ini dapat dilihat pada kode berikut.

```
class EditUserView(View):
  def post(self, request):
    nim = request.POST.get('nim')
    email = request.POST.get('email')
    first_name = request.POST.get('first_name')
    last_name = request.POST.get('last_name')
    role = request.POST.get('role')
    prodi_id = request.POST.get('prodi')
    is_active = request.POST.get('is_active') == 'on'
    try:
      user = Account.objects.get(nim=nim)
      user.email = email
      user.first_name = first_name
      user.last_name = last_name
      user.role = role
      user.prodi_id = prodi_id
      user.is_active = is_active
      user.save()
      messages.success(request, "User berhasil diperbarui.")
    except Account.DoesNotExist:
      messages.error(request, "User tidak ditemukan.")
    except Exception as e:
      messages.error(request, f"Gagal mengedit user: {e}")
    return redirect('admin:user_list')
```

Kode program di atas digunakan untuk memperbarui data pengguna yang sudah ada di dalam sistem. Admin dapat mengedit atribut pengguna seperti email, nama, role, prodi, dan status aktif. Jika data berhasil diperbarui, sistem akan menampilkan pesan sukses, sedangkan jika terjadi kesalahan atau pengguna tidak ditemukan, sistem akan memberikan pesan error yang sesuai.

4.2.6 Import *Excel*

Pada Gambar 4.6 memperlihatkan proses impor data pemilih secara massal ke dalam sistem, di mana admin dapat memilih file data mahasiswa berformat Excel (.xlsx) untuk diunggah. Fitur ini memudahkan penambahan banyak data pemilih sekaligus secara efisien.



Gambar 4.6 Import Excel Data Pemilih

Potongan kode program yang digunakan untuk mengelola proses impor data dapat dilihat pada kode berikut.

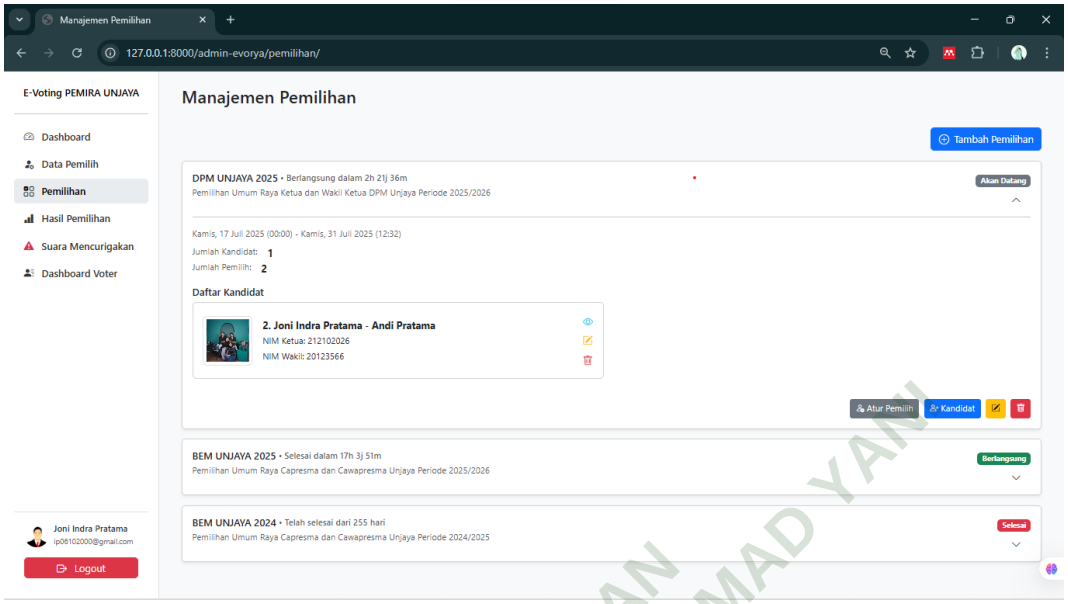
```
class ImportUserView(View):
  def post(self, request):
    file = request.FILES.get('file')
    if not file:
      messages.error(request, "File tidak ditemukan.")
      return redirect('admin:user_list')
    try:
      df = pd.read_excel(file) if file.name.endswith(('.xls',
          '.xlsx')) else pd.read_csv(file)
      sukses = 0
      dilewati = 0
      for _, row in df.iterrows():
        nim = str(row['NIM']).strip()
        email = row['Email'].strip()
        nama_lengkap = row['Nama Lengkap'].strip()
        nama_parts = nama_lengkap.split()
        first_name = nama_parts[0]
        last_name = " ".join(nama_parts[1:]) if len(nama_parts)
                    > 1 else ""
        prodi_nama = row['Prodi'].strip()
        try:
          prodi = Prodi.objects.get(nama__iexact=prodi_nama)
        except Prodi.DoesNotExist:
          messages.warning(request, f"Prodi '{prodi_nama}' tidak
                           ditemukan.")
          continue
        if Account.objects.filter(nim=nim).exists():
          dilewati += 1
          continue
```

```
        Account.objects.create_user(
          nim=nim,
          email=email,
          password=nim,
          first_name=first_name,
          last_name=last_name,
          role='voter',
          prodi=prodi
        )
        sukses += 1
      messages.success(request, f"Berhasil mengimpor {sukses}
                       data pengguna. {dilewati} data dilewati
                       karena NIM sudah ada.")
    except Exception as e:
      messages.error(request, f"Terjadi kesalahan saat impor:
                     {e}")
    return redirect('admin:user_list')
```

Fungsi dari potongan kode di atas adalah untuk menangani proses impor data pemilih dari file Excel atau CSV ke dalam sistem. Saat admin mengunggah file, sistem akan membaca dan memproses setiap baris data, melakukan validasi prodi, serta menambahkan data pemilih baru jika NIM belum terdaftar. Jika terjadi duplikasi atau prodi tidak ditemukan, data tersebut akan dilewati, dan hasil impor akan diinformasikan ke admin melalui pesan sukses atau peringatan.

4.2.7 Halaman Pemilihan

Halaman Manajemen Pemilihan menampilkan daftar seluruh pemilihan yang telah dibuat, termasuk status, deskripsi, periode pelaksanaan, jumlah kandidat, dan jumlah pemilih pada setiap pemilihan. Admin dapat menambah, mengedit, menghapus, serta mengelola kandidat dan pemilih untuk setiap pemilihan melalui tombol aksi yang tersedia. Navigasi halaman juga memungkinkan admin melihat detail masing-masing kandidat secara langsung. Gambar 4.7 menunjukan implementasi antarmuka halaman pemilihan.



Gambar 4.7 Halaman Pemilihan

Cuplikan baris kode program yang digunakan untuk menampilkan data pemilihan pada sistem ini dapat dilihat pada kode berikut.

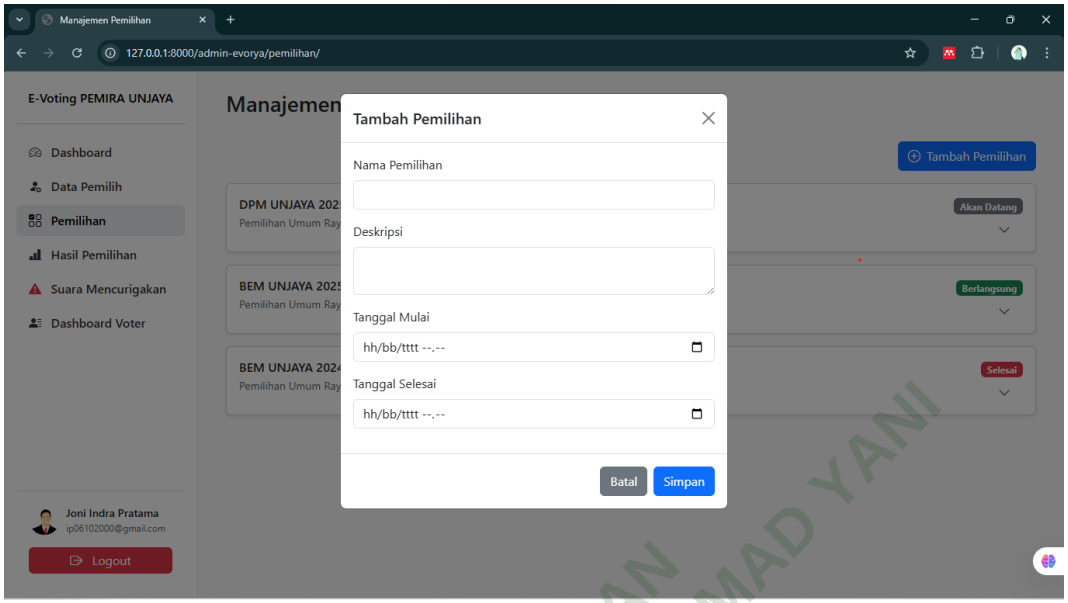
```
class PemilihanListView(View):
  def get(self, request):
    search_query = request.GET.get('q', '')
    sort_by = request.GET.get('sort', 'waktu_mulai')
    direction = request.GET.get('dir', 'desc')
    order = sort_by if direction == 'asc' else f'-{sort_by}'

    pemilihan_qs = Pemilihan.objects.annotate
                  (jumlah_pemilih=Count('pemilih'))
    if search_query:
      pemilihan_qs = pemilihan_qs.filter(
        Q(nama__icontains=search_query) |
        Q(deskripsi__icontains=search_query)
      )
    pemilihan_qs = pemilihan_qs.order_by(order)

    per_page = request.GET.get('per_page', '10')
    paginator = Paginator(pemilihan_qs, int(per_page))
    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)

    semua_mahasiswa = Account.objects.exclude(is_superuser=True)
    prodi_list = sorted(
      set(m.prodi for m in semua_mahasiswa if m.prodi),
      key=lambda p: (p.fakultas.nama, p.nama)
    )
    angkatan_list = sorted(
      set(mahasiswa.nim[:2] for mahasiswa in semua_mahasiswa if
          mahasiswa.nim and len(mahasiswa.nim) >= 2),
```

```
      key=lambda x: int(x)
    )

    context = {
      'daftar_pemilihan': page_obj,
      'page_obj': page_obj,
      'search_query': search_query,
      'sort_by': sort_by,
      'direction': direction,
      'per_page': per_page,
      'field_labels': [
        ('nama', 'Nama'),
        ('waktu_mulai', 'Mulai'),
        ('waktu_selesai', 'Selesai'),
        ('status', 'Status'),
      ],
      'semua_mahasiswa': semua_mahasiswa,
      'angkatan_list': angkatan_list,
      'prodi_list': prodi_list,
    }
    return render(request, 'admin/pemilihan_list.html', context)
```

Potongan kode di atas berfungsi untuk menampilkan daftar data pemilihan dalam sistem, beserta fitur pencarian, pengurutan, dan paginasi. Data yang ditampilkan bisa difilter berdasarkan kata kunci dan diurutkan sesuai kebutuhan admin. Selain itu, juga diambil data prodi dan angkatan untuk keperluan filter dan tampilan pada halaman daftar pemilihan.

4.2.8 Tambah Data Pemilihan

Gambar 4.8 merupakan tampilan form tambah pemilihan pada sistem, di mana admin dapat menginput nama pemilihan, deskripsi, tanggal mulai, dan tanggal selesai. Fitur ini memudahkan admin untuk menambah jadwal pemilihan baru dengan antarmuka yang sederhana dan intuitif.



Gambar 4.8 Tambah Data Pemilihan

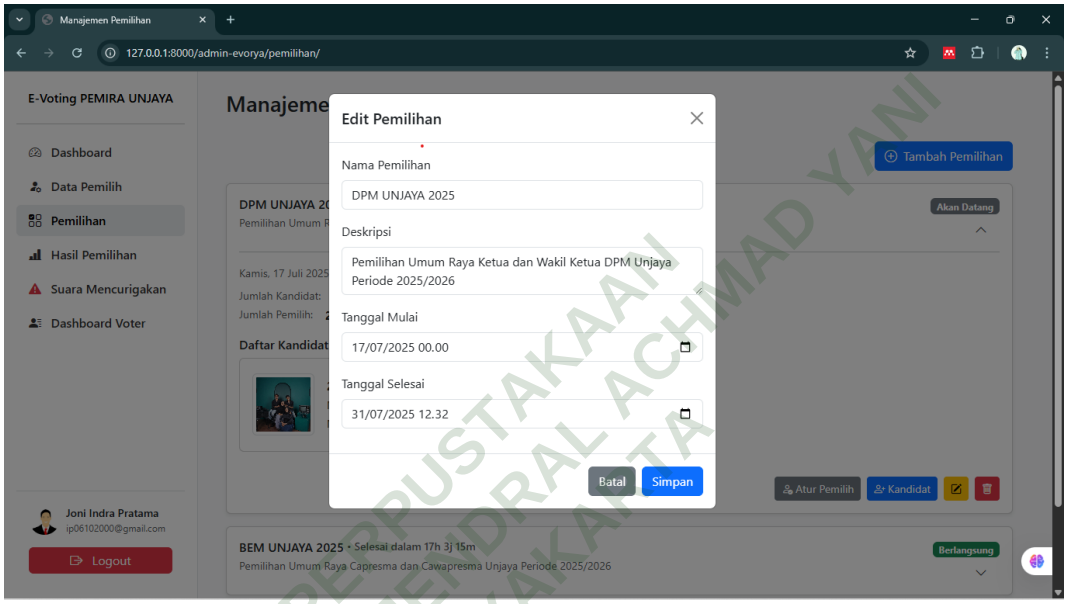
Cuplikan baris kode program yang digunakan untuk menambah data pemilihan pada sistem ini dapat dilihat pada kode berikut.

```
class AddPemilihanView(View):
  def post(self, request):
    nama = request.POST.get('nama')
    deskripsi = request.POST.get('deskripsi')
    waktu_mulai = request.POST.get('waktu_mulai')
    waktu_selesai = request.POST.get('waktu_selesai')
    try:
      Pemilihan.objects.create(
        nama=nama,
        deskripsi=deskripsi,
        waktu_mulai=waktu_mulai,
        waktu_selesai=waktu_selesai
      )
      messages.success(request, "Pemilihan berhasil
                       ditambahkan.")
    except Exception as e:
      messages.error(request, f"Gagal menambahkan pemilihan:
                     {e}")
    return redirect('admin:pemilihan_list')
```

Potongan kode di atas berfungsi untuk menambahkan data pemilihan baru ke dalam sistem. Ketika admin mengisi dan mengirimkan form tambah pemilihan, data yang diinput akan diproses dan disimpan ke *database* jika valid, kemudian akan muncul notifikasi sukses atau gagal sesuai hasil prosesnya.

4.2.9 Edit Pemilihan

Gambar 4.9 menampilkan form edit pemilihan yang memungkinkan admin untuk memperbarui informasi nama, deskripsi, tanggal mulai, dan tanggal selesai pada data pemilihan yang telah ada. Fitur ini mempermudah admin dalam melakukan perubahan data tanpa harus membuat ulang pemilihan baru.



Gambar 4.9 Edit Data Pemilihan

Cuplikan baris kode program yang digunakan untuk memperbarui data pemilihan pada sistem ini dapat dilihat pada kode berikut.

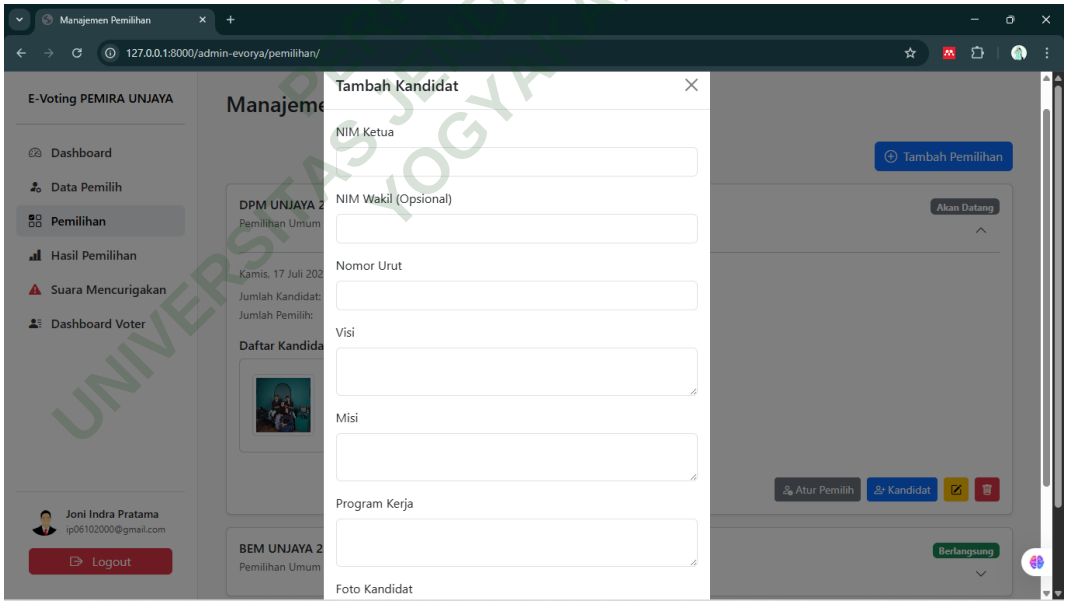
```
class EditPemilihanView(View):
  def post(self, request):
    id = request.POST.get('id')
    nama = request.POST.get('nama')
    deskripsi = request.POST.get('deskripsi')
    waktu_mulai = request.POST.get('waktu_mulai')
    waktu_selesai = request.POST.get('waktu_selesai')

    try:
      pemilihan = Pemilihan.objects.get(id=id)
      pemilihan.nama = nama
      pemilihan.deskripsi = deskripsi
      pemilihan.waktu_mulai = waktu_mulai
      pemilihan.waktu_selesai = waktu_selesai
      pemilihan.save()

      messages.success(request, "Pemilihan berhasil
                       diperbarui.")
```

```
    except Pemilihan.DoesNotExist:
      messages.error(request, "Pemilihan tidak ditemukan.")
    except Exception as e:
      messages.error(request, f"Gagal mengedit pemilihan: {e}")

    return redirect('admin:pemilihan_list')
```

Kode di atas digunakan untuk memperbarui data pemilihan berdasarkan input form edit pemilihan. Jika data berhasil diperbarui, maka akan ditampilkan pesan sukses. Namun jika terjadi kesalahan, misalnya data tidak ditemukan atau gagal saat proses simpan, maka akan muncul pesan error yang sesuai.

4.2.10 Tambah Kandidat

Tampilan form pada halaman "Tambah Kandidat" digunakan untuk menambahkan data kandidat baru pada suatu pemilihan. Admin dapat mengisi data seperti NIM Ketua, NIM Wakil (opsional), nomor urut, visi, misi, program kerja, serta mengunggah foto kandidat melalui form ini. Gambar 4.10 menunjukan impelemntasi antarmuka tambah kandidat pada sistem.



Gambar 4.10 Tambah Kandidat

Cuplikan baris kode program yang digunakan untuk menambah data kandidat pada sistem ini dapat dilihat pada kode berikut.

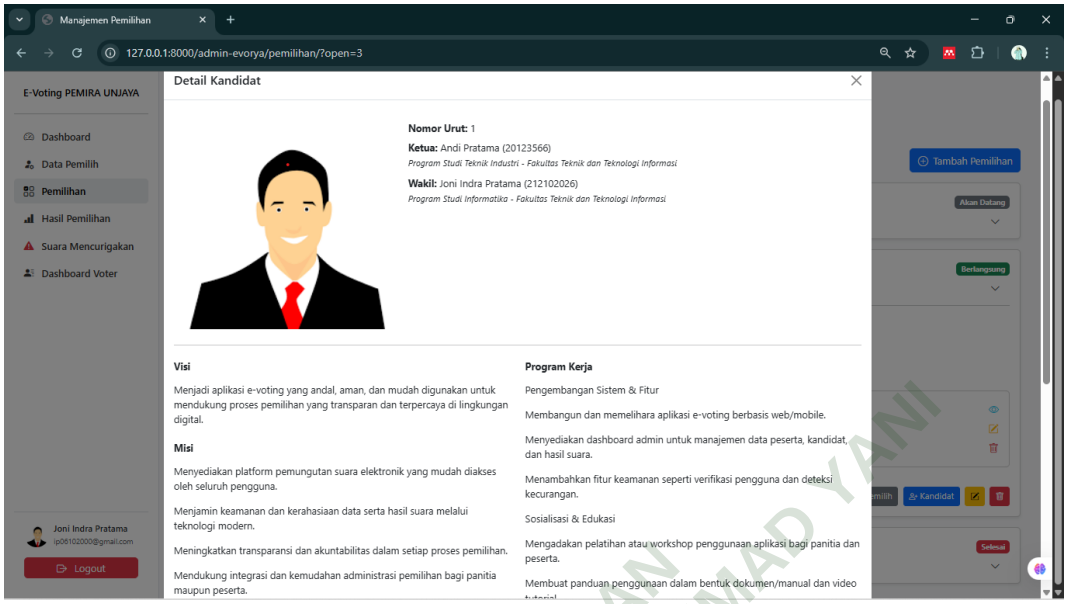
```
class TambahKandidatView(View):
  def post(self, request, pemilihan_id):
    pemilihan = get_object_or_404(Pemilihan, id=pemilihan_id)
    ketua_nim = request.POST.get('ketua')
    wakil_nim = request.POST.get('wakil')
    nomor_urut = request.POST.get('nomor_urut')
    visi = request.POST.get('visi')
    misi = request.POST.get('misi')
    proker = request.POST.get('proker')
    foto = request.FILES.get('foto')

    if not nomor_urut or not nomor_urut.isdigit() or int
           (nomor_urut) <= 0:
      messages.error(request, "Nomor urut harus berupa angka
                     positif.")
      return redirect('admin:pemilihan_list')

    nomor_urut = int(nomor_urut)

    try:
      ketua = Account.objects.get(nim=ketua_nim)
    except Account.DoesNotExist:
      messages.error(request, f"NIM Ketua '{ketua_nim}' tidak
                     ditemukan.")
      return redirect('admin:pemilihan_list')

    wakil = None
    if wakil_nim:
      if wakil_nim == ketua_nim:
        messages.error(request, "NIM Ketua dan Wakil tidak boleh
                       sama.")
        return redirect('admin:pemilihan_list')
      try:
        wakil = Account.objects.get(nim=wakil_nim)
      except Account.DoesNotExist:
        messages.error(request, f"NIM Wakil '{wakil_nim}' tidak
                       ditemukan.")
        return redirect('admin:pemilihan_list')
    if Kandidat.objects.filter(pemilihan=pemilihan,
    nomor_urut=nomor_urut).exists():
      messages.error(request, f"Nomor urut {nomor_urut} sudah
                     digunakan dalam pemilihan ini.")
      return redirect('admin:pemilihan_list')
    if Kandidat.objects.filter(pemilihan=pemilihan)
       .filter(ketua=ketua).exists() or \
       Kandidat.objects.filter(pemilihan=pemilihan)
       .filter(wakil=ketua).exists():
      messages.error(request, f"Ketua dengan NIM '{ketua_nim}'
                     sudah menjadi kandidat.")
      return redirect('admin:pemilihan_list')

    if wakil and (Kandidat.objects.filter(pemilihan=pemilihan)
```

```
                .filter(ketua=wakil).exists() or \
                Kandidat.objects.filter(pemilihan=pemilihan)
                .filter(wakil=wakil).exists()):
    messages.error(request, f"Wakil dengan NIM '{wakil_nim}'
                    sudah menjadi kandidat.")
    return redirect('admin:pemilihan_list')
  Kandidat.objects.create(
    pemilihan=pemilihan,
    ketua=ketua,
    wakil=wakil,
    nomor_urut=nomor_urut,
    visi=visi,
    misi=misi,
    proker=proker,
    foto=foto
  )
  messages.success(request, "Kandidat berhasil ditambahkan.")
  return redirect(f"{reverse('admin:pemilihan_list')}?open=
                  {pemilihan.id}")
```

Kode program di atas digunakan untuk memproses penambahan data kandidat baru pada suatu pemilihan. Fungsi ini menerima data dari form tambah kandidat, memvalidasi input seperti nomor urut, NIM ketua dan wakil, serta memastikan tidak ada duplikasi kandidat pada satu pemilihan. Jika validasi berhasil, maka data kandidat akan disimpan ke *database* dan notifikasi keberhasilan ditampilkan.

4.2.11 Details Kandidat

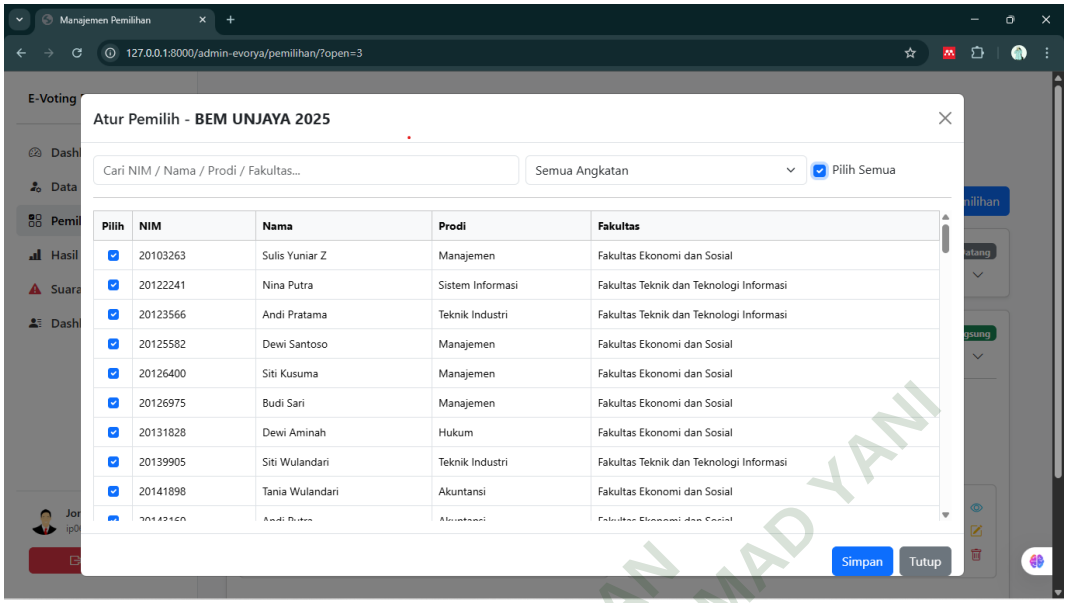
Halaman detail kandidat menampilkan informasi lengkap mengenai kandidat yang terdaftar pada suatu pemilihan. Pada tampilan ini, ditampilkan foto kandidat, nomor urut, nama ketua dan wakil beserta program studi masing-masing, serta visi, misi, dan program kerja yang diusung oleh kandidat. Informasi ini disajikan secara jelas dan terstruktur untuk memudahkan admin maupun pemilih dalam mengenal setiap kandidat. Gambar 4.11 menunjukan impelemntasi antarmuka detail kandidat pada sistem.



Gambar 4.11 Details Kandidat

4.2.12 Atur Pemilih

Halaman atur pemilih digunakan untuk menentukan siapa saja yang dapat menjadi pemilih dalam suatu pemilihan. Pada halaman ini, admin dapat melakukan pencarian, filter berdasarkan angkatan, dan memilih seluruh daftar mahasiswa yang ingin diikutsertakan sebagai pemilih. Setiap baris menampilkan NIM, nama, prodi, dan fakultas dari masing-masing mahasiswa, sehingga proses seleksi menjadi lebih mudah dan terstruktur. Gambar 4.12 menunjukan impelemntasi antarmuka atur pemilih pada sistem.



Gambar 4.12 Atur Pemilih

Cuplikan baris kode program yang digunakan untuk mengatur pemilih dalam sistem ini dapat dilihat pada kode berikut.

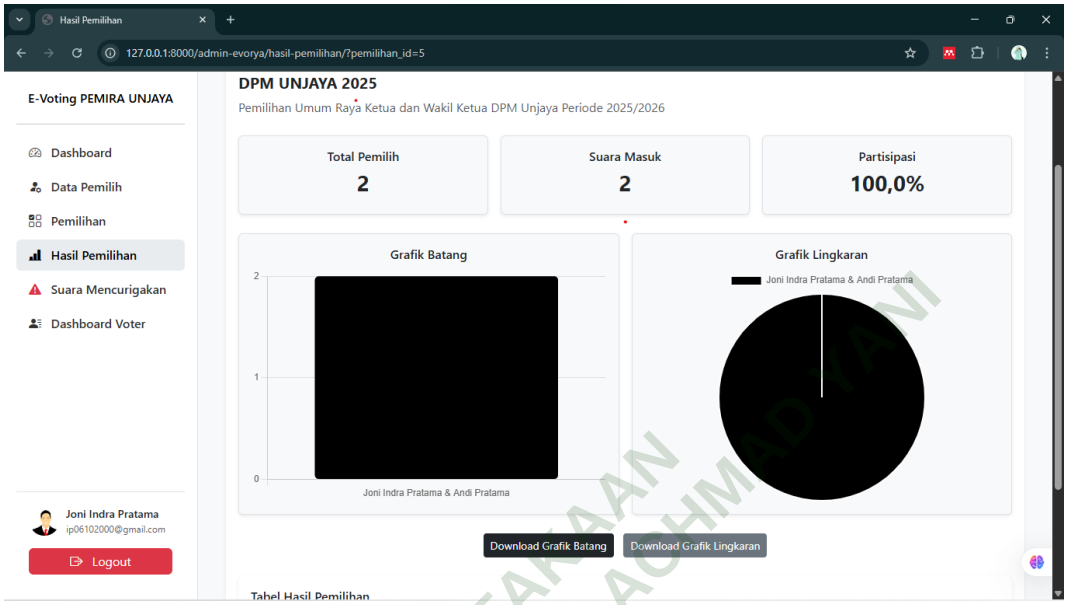
```
class AturPemilihView(View):
  def post(self, request, pemilihan_id):
    pemilihan = get_object_or_404(Pemilihan, id=pemilihan_id)
    pemilih_nim_list = request.POST.getlist('pemilih')
    pemilih_qs = Account.objects.filter(
                  nim__in=pemilih_nim_list)
    pemilihan.pemilih.set(pemilih_qs)
    messages.success(request, "Daftar pemilih berhasil
                      diperbarui.")
    return redirect(f"{reverse('admin:pemilihan_list')}?open=
                    {pemilihan_id}")
```

Fungsi dari kode di atas adalah untuk memperbarui daftar pemilih pada suatu pemilihan berdasarkan NIM yang dipilih oleh admin. Data pemilih yang dipilih akan disimpan ke dalam daftar pemilih pada objek pemilihan terkait. Jika proses berhasil, sistem akan menampilkan pesan sukses dan mengembalikan admin ke halaman daftar pemilihan.

4.2.13 Halaman Hasil Pemilihan

Pada halaman ini, admin dapat melihat rekapitulasi hasil pemilihan berupa jumlah total pemilih, suara yang masuk, serta tingkat partisipasi. Selain itu, hasil pemilihan juga divisualisasikan dalam bentuk grafik batang dan grafik lingkaran

untuk memudahkan analisis. Terdapat pula fitur untuk mengunduh grafik hasil pemilihan. Gambar 4.13 menunjukan implementasi antarmuka hasil pemilihan.



Gambar 4.13 Halaman Hasil Pemilihan

Cuplikan baris kode program yang digunakan untuk menampilkan hasil pemilihan serta visualisasi grafik dapat dilihat pada kode berikut:

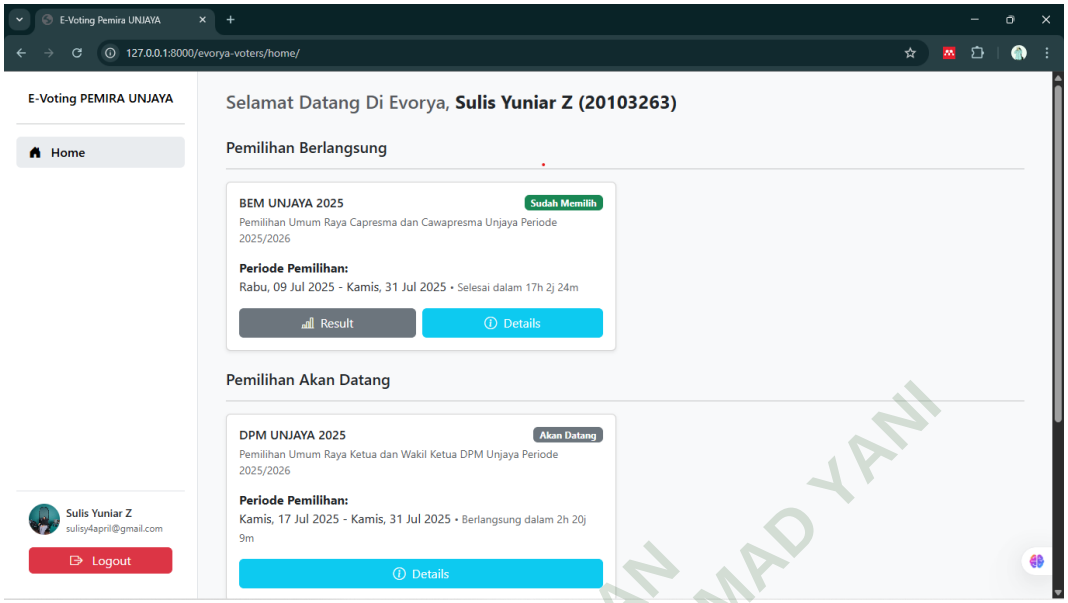
```
class HasilPemilihanView(TemplateView):
  template_name = 'admin/hasil_pemilihan.html'

  def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    semua_pemilihan = Pemilihan.objects.all().order_by
                      ('-waktu_mulai')
    pemilihan_id = self.request.GET.get('pemilihan_id')
    pemilihan = None
    suara_kandidat = []
    total_suara = 0
    jumlah_pemilih = 0
    partisipasi = 0
    names = []
    values = []
    if pemilihan_id:
      pemilihan = get_object_or_404(Pemilihan, id=pemilihan_id)
      suara_dict = dict(
        Suara.objects.filter(pemilihan=pemilihan)
        .values_list('kandidat_id')
        .annotate(jumlah=Count('id'))
      )
      total_suara = sum(suara_dict.values())
      jumlah_pemilih = pemilihan.pemilih.count()
      partisipasi = (total_suara / jumlah_pemilih * 100) if
```

```
                jumlah_pemilih > 0 else 0
    kandidat_qs = Kandidat.objects.filter(pemilihan=pemilihan)
                .select_related('ketua', 'wakil')
                .order_by('nomor_urut')
    for kandidat in kandidat_qs:
      jumlah = suara_dict.get(kandidat.id, 0)
      persentase = (jumlah / total_suara * 100) if total_suara
                 > 0 else 0
      kandidat.jumlah = jumlah
      kandidat.persentase = round(persentase, 2)
      label_nama = kandidat.ketua.get_full_name()
      if kandidat.wakil:
        label_nama += f" & {kandidat.wakil.get_full_name()}"
      names.append(label_nama)
      values.append(jumlah)
      suara_kandidat.append(kandidat)
  context.update({
    'semua_pemilihan': semua_pemilihan,
    'pemilihan': pemilihan,
    'suara_kandidat': suara_kandidat,
    'jumlah_pemilih': jumlah_pemilih,
    'total_suara': total_suara,
    'persentase_partisipasi': round(partisipasi, 2),
    'names': json.dumps(names),
    'values': json.dumps(values),
  })
  return context
```

Kode di atas merupakan bagian dari proses untuk menampilkan hasil pemilihan pada sistem *e-voting*. Fungsi ini mengambil data pemilihan, menghitung total suara yang masuk, jumlah pemilih, dan tingkat partisipasi, serta menghasilkan data yang diperlukan untuk menampilkan grafik hasil suara tiap kandidat. Data kandidat dan hasil perolehan suaranya kemudian ditampilkan dalam bentuk tabel dan grafik pada antarmuka admin.

4.2.14 Dashboard Pemilih

Halaman utama untuk pengguna (*voter*) yang sudah login menampilkan daftar pemilihan yang sedang berlangsung dan pemilihan yang akan datang. Pada bagian ini, pengguna dapat melihat detail pemilihan, periode pelaksanaan, serta status apakah sudah melakukan pemilihan atau belum. Tersedia juga tombol untuk melihat hasil dan detail dari setiap pemilihan yang tersedia. Gambar 4.14 menunjukan impelemntasi antarmuka halaman utama *voter* pada sistem.



Gambar 4.14 Dashboard Pemilih

Cuplikan baris kode program yang digunakan untuk menampilkan halaman utama *voter* dan memfilter daftar pemilihan berdasarkan statusnya dapat dilihat pada kode berikut.

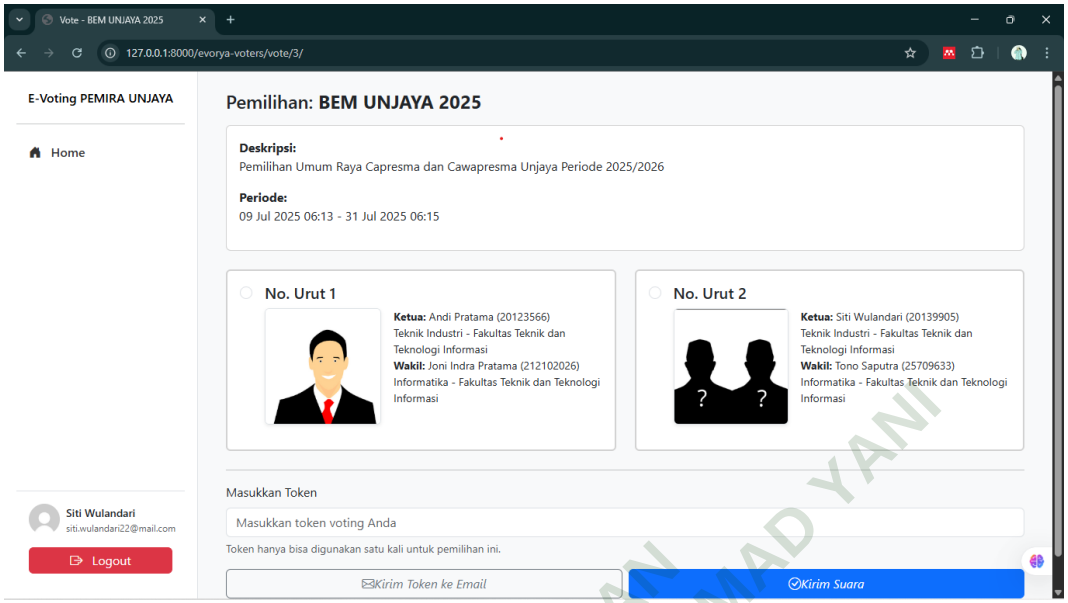
```
class HomeView(View):
  def get(self, request, *args, **kwargs):
    user = request.user
    rekomendasi_ganti_password = False
    if user.check_password(user.nim):
      rekomendasi_ganti_password = True

    semua_pemilihan = user.pemilihan_diikuti.all()
                      .order_by('-waktu_mulai')
    data = []
    for p in semua_pemilihan:
      sudah_memilih = Suara.objects.filter
                      (pemilihan=p, pemilih=user).exists()
      total_suara = Suara.objects.filter(pemilihan=p).count()
      kandidat_suara = []
      for k in p.kandidat_list.all():
        jumlah_suara = Suara.objects.filter(kandidat=k).count()
        persen = round((jumlah_suara / total_suara) * 100, 2) if
                 total_suara > 0 else 0
        kandidat_suara.append({
          'kandidat': k,
          'jumlah_suara': jumlah_suara,
          'persen': persen,
        })
      data.append({
        'pemilihan': p,
        'status': p.status,
```

```
        'sudah_memilih': sudah_memilih,
        'kandidat_suara': kandidat_suara,
        'total_suara': total_suara,
      })
    grouped = {
      'Berlangsung': [],
      'Akan Datang': [],
      'Selesai': [],
    }
    for item in data:
      grouped[item['status']].append(item)
    grouped_pemilihan = [
      ('Berlangsung', grouped['Berlangsung']),
      ('Akan Datang', grouped['Akan Datang']),
      ('Selesai', grouped['Selesai']),
    ]
    total_pemilihan = len(semua_pemilihan)
    return render(request, 'voter/home.html', {
      'grouped_pemilihan': grouped_pemilihan,
      'total_pemilihan': total_pemilihan,
      'rekomendasi_ganti_password': rekomendasi_ganti_password,
    })
```

Potongan kode di atas berfungsi untuk menampilkan halaman utama bagi pengguna *voter*. Kode ini mengambil seluruh data pemilihan yang diikuti oleh pengguna, kemudian mengelompokkan status pemilihan menjadi “Berlangsung”, “Akan Datang”, dan “Selesai”, serta menampilkan status pemilih apakah sudah memilih atau belum. Data hasil perolehan suara untuk tiap kandidat pada setiap pemilihan juga diproses agar dapat ditampilkan kepada pengguna.

4.2.15 Halaman *Vote*/Pemilihan Kandidat

Halaman ini merupakan tampilan proses pemilihan oleh *voter*. Pengguna dapat melihat informasi detail mengenai pemilihan, daftar kandidat beserta pasangannya, dan memilih salah satu kandidat dengan menginputkan token *voting* yang diterima melalui email. Setelah token dimasukkan, pengguna dapat menekan tombol "Kirim Suara" untuk mengirimkan pilihannya. Token juga bisa terisi otomatis dengan menggunakan link yang dikirimkan ke email. Gambar 4.15 menunjukan impelemntasi antarmuka proses *voting* oleh user (*voter*).



Gambar 4.15 Halaman Vote/Pemilihan Kandidat

Cuplikan baris kode program yang digunakan untuk proses *voting* oleh *voter* dapat dilihat pada kode berikut.

```
class VotePageView(View):
  def get(self, request, pemilihan_id):
    pemilihan = get_object_or_404(Pemilihan, id=pemilihan_id)
    if request.user not in pemilihan.pemilih.all():
      messages.error(request, "Anda tidak berhak memilih dalam
                     pemilihan ini.")
      return redirect('voters:home')
    if Suara.objects.filter(pemilihan=pemilihan,
       pemilih=request.user).exists():
      messages.warning(request, "Anda sudah memberikan suara.")
      return redirect('voters:hasil_pemilihan', pemilihan.id)
    kandidat_list = pemilihan.kandidat_list.all()
                    .order_by('nomor_urut')
    return render(request, 'voter/vote_page.html', {
      'pemilihan': pemilihan,
      'kandidat_list': kandidat_list,
    })

  def post(self, request, pemilihan_id):
    pemilihan = get_object_or_404(Pemilihan, id=pemilihan_id)
    kandidat_id = request.POST.get('kandidat')
    if not kandidat_id:
      messages.error(request, "Silakan pilih salah satu
                     kandidat.")
      return redirect('voters:vote_page', pemilihan.id)
    kandidat = get_object_or_404(Kandidat, id=kandidat_id,
               pemilihan=pemilihan)
    if Suara.objects.filter(pemilihan=pemilihan,
       pemilih=request.user).exists():
```

```
      messages.warning(request, "Anda sudah memberikan suara.")
      return redirect('voters:hasil_pemilihan', pemilihan.id)
    Suara.objects.create(
      pemilihan=pemilihan,
      pemilih=request.user,
      kandidat=kandidat,
    )
    messages.success(request, "Terima kasih, suara Anda telah
                     terekam.")
    return redirect('voters:hasil_pemilihan', pemilihan.id)

# Kirim Token
class SendTokenView(View):
  def get(self, request, pemilihan_id):
    user = request.user
    pemilihan = get_object_or_404(Pemilihan, id=pemilihan_id)
    if user not in pemilihan.pemilih.all():
      messages.error(request, "Anda tidak terdaftar sebagai
                     pemilih pada pemilihan ini.")
      return redirect('voters:vote_page',
                      pemilihan_id=pemilihan_id)
    if pemilihan.status != 'Berlangsung':
      messages.error(request, "Pemilihan belum dimulai atau
                     sudah selesai.")
      return redirect('voters:vote_page',
                      pemilihan_id=pemilihan_id)
    token_obj, created = TokenVote.objects.get_or_create(
      pemilihan=pemilihan,
      pemilih=user,
      defaults={'token': get_random_string(12).upper()}
    )
    if not created and token_obj.digunakan:
      messages.error(request, "Token Anda sudah digunakan.")
      return redirect('voters:vote_page',
                      pemilihan_id=pemilihan_id)
    try:
      vote_url = request.build_absolute_uri(
        reverse('voters:vote_page', args=[pemilihan.id]) +
                f"?token={token_obj.token}"
      )
      message = f"""
      Hai {user.full_name},
      Berikut adalah token untuk mengikuti pemilihan:
      {pemilihan.nama}
      🔐 Token Anda: {token_obj.token}
      Klik link berikut untuk langsung memilih:
      {vote_url}
      Token ini hanya bisa digunakan sekali dan berlaku hingga
      {pemilihan.waktu_selesai.strftime('%d %B %Y %H:%M')}.
      Jangan bagikan token ini kepada orang lain.
      Terima kasih,
      Panitia Pemira UNJAYA
      """
      send_mail(
        subject=f"Token Voting: {pemilihan.nama}",
        message=message.strip(),
```

```
        from_email='ip06102000@gmail.com',
        recipient_list=[user.email],
        fail_silently=False,
      )
      messages.success(request, "Token berhasil dikirim ke email
                       Anda.")
    except Exception as e:
      messages.error(request, f"Gagal mengirim email: {str(e)}")
    return redirect('voters:vote_page',
                    pemilihan_id=pemilihan_id)

# Submit Pemilihan
class SubmitVoteView(View):
  def post(self, request, pemilihan_id):
    pemilihan = get_object_or_404(Pemilihan, id=pemilihan_id)
    user = request.user
    token_input = request.POST.get('token')
    kandidat_id = request.POST.get('kandidat')
    if not pemilihan.pemilih.filter(pk=user.pk).exists():
      messages.error(request, "Anda tidak terdaftar sebagai
                     pemilih.")
      return redirect('voters:vote_page',
                      pemilihan_id=pemilihan.id)

    if Suara.objects.filter(pemilihan=pemilihan,
       pemilih=user).exists():
      messages.warning(request, "Anda sudah memberikan suara.")
      return redirect('voters:vote_page',
                      pemilihan_id=pemilihan.id)
    try:
      token = TokenVote.objects.get(pemilihan=pemilihan,
              pemilih=user, token=token_input)
    except TokenVote.DoesNotExist:
      messages.error(request, "Token tidak valid.")
      return redirect('voters:vote_page',
                      pemilihan_id=pemilihan.id)
    if token.digunakan:
      messages.error(request, "Token sudah digunakan.")
      return redirect('voters:vote_page',
                      pemilihan_id=pemilihan.id)
    if token.expired_at and timezone.now() > token.expired_at:
      messages.error(request, "Token sudah kedaluwarsa.")
      return redirect('voters:vote_page',
                      pemilihan_id=pemilihan.id)
    kandidat = get_object_or_404(Kandidat, id=kandidat_id,
               pemilihan=pemilihan)
    ip = request.META.get('REMOTE_ADDR')
    user_agent = request.META.get('HTTP_USER_AGENT', '')
    suara = Suara.objects.create(
      pemilihan=pemilihan,
      kandidat=kandidat,
      pemilih=user,
      ip_address=ip,
      user_agent=user_agent
    )
    token.digunakan = True
```

```
    token.waktu_digunakan = timezone.now()
    token.save()
    kecurigaan, alasan = deteksi_kecurangan(suara)
    if kecurigaan:
      suara.is_suspicious = True
      suara.alasan_kecurangan = alasan
      suara.save()
      messages.warning(request, "Voting Anda berhasil, namun
                  aktivitas Anda terdeteksi mencurigakan.")
    else:
      messages.success(request, "Suara Anda berhasil dikirim.")
    return redirect('voters:home')
```

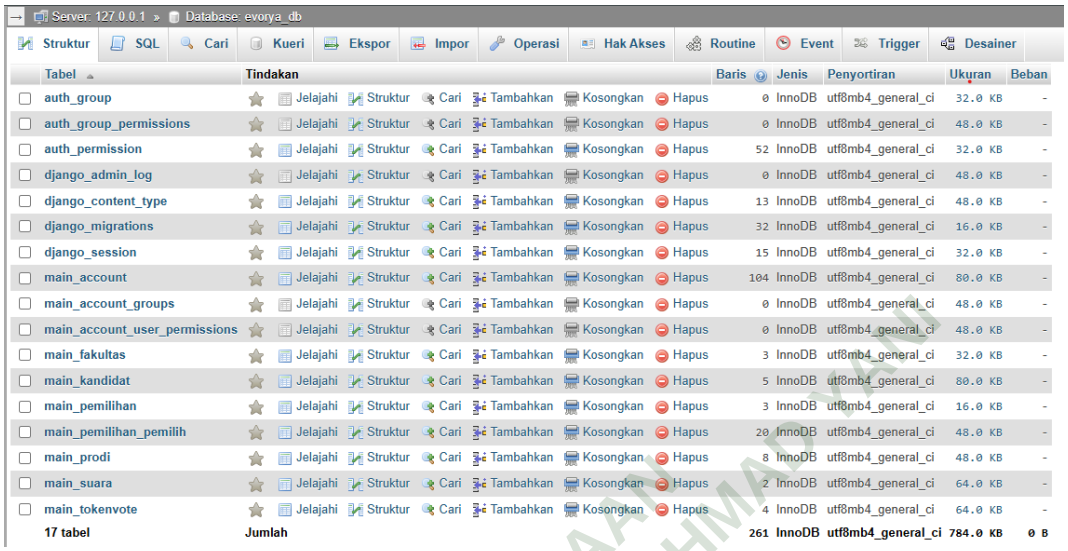
Potongan kode di atas menangani seluruh proses *voting* dari sisi *voter*. Terdapat tiga kelas utama: VotePageView untuk menampilkan halaman *voting* dan validasi hak memilih, SendTokenView untuk mengirim token *voting* ke email pemilih, dan SubmitVoteView untuk memvalidasi token serta merekam suara ke sistem. Seluruh validasi akses, status pemilihan, validasi token, hingga deteksi kecurangan telah diintegrasikan agar proses pemilihan berjalan aman dan hanya satu suara yang dapat diberikan oleh setiap pemilih.

4.3 BASIS DATA

Tampilan basis data evorya_db yang digunakan pada sistem *e-voting* PEMIRA UNJAYA. Basis data ini terdiri dari 17 tabel utama yang mencakup tabel-tabel autentikasi dari Django (auth_*, django_*) serta tabel-tabel inti yang dikembangkan untuk keperluan sistem, seperti main_*account*, main_fakultas, main_prodi, main_pemilihan, main_kandidat, main_suara, dan main_tokenvote. Tabel-tabel ini saling terhubung untuk mengelola data pengguna, kandidat, daftar pemilihan, proses *voting*, hingga token keamanan untuk memastikan integritas dan keamanan suara. Semua data yang tersimpan pada basis data ini menggunakan format penyimpanan InnoDB dan penulisan karakter utf8mb4 untuk mendukung karakter Unicode.

Seluruh struktur tabel dirancang untuk memenuhi kebutuhan sistem, mulai dari pengelolaan user, proses autentikasi, perekaman suara, hingga manajemen kandidat dan daftar pemilih pada tiap pemilihan. Selain itu, *database* ini juga mendukung fitur audit dan keamanan dengan adanya tabel token *voting* dan status kecurigaan pada suara.

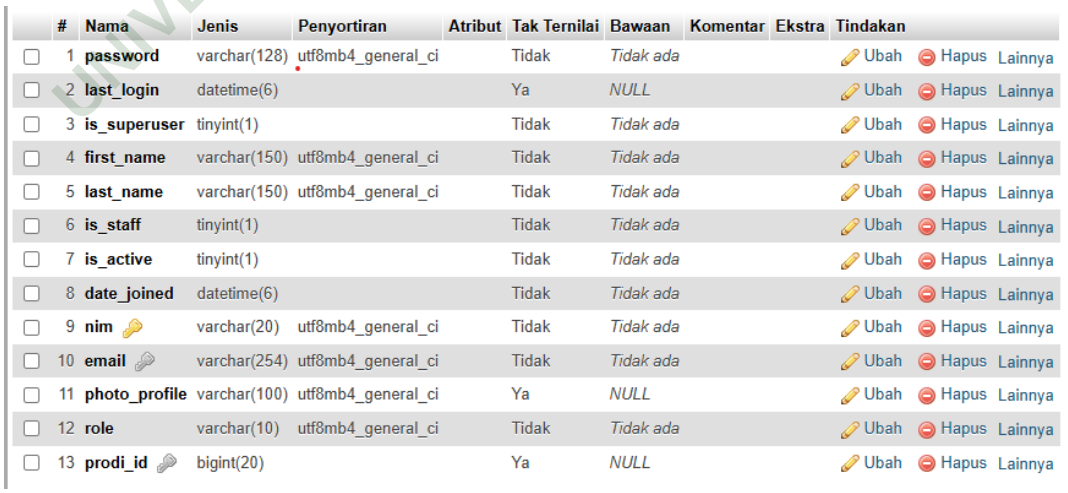
Gambar 4.16 menampilkan basis data yang digunakan pada sistem.



Gambar 4.16 Implementasi Struktur *Database*

4.3.1 Tabel *Account*

Tabel main_*account* digunakan untuk menyimpan data seluruh pengguna sistem *e-voting*, baik admin maupun pemilih (*voter*). Setiap akun diidentifikasi dengan NIM sebagai primary key, dan memiliki atribut-atribut seperti nama, email, *password*, role, status aktif, serta referensi ke program studi. Tabel ini berperan sebagai pusat autentikasi dan identitas pengguna dalam aplikasi. Gambar 4.17 menunjukan struktur tabel main_*account* pada sistem.



Gambar 4.17 Implementasi Tabel *Account*

4.3.2 Tabel Fakultas

Tabel ini berfungsi untuk menyimpan data seluruh fakultas yang ada di universitas, dengan setiap entri memiliki atribut nama fakultas dan id unik sebagai primary key. Data dari tabel ini digunakan sebagai referensi untuk relasi pada tabel lain seperti prodi dan pengguna. Gambar 4.18 menunjukan struktur tabel main_fakultas.

#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan	Komentar	Ekstra	Tindakan
1	id	bigint(20)			Tidak	*Tidak ada*		AUTO_INCREMENT	Ubah Hapus Lainnya
2	nama	varchar(100)	utf8mb4_general_ci		Tidak	*Tidak ada*			Ubah Hapus Lainnya

Gambar 4.18 Implementasi Tabel Fakultas

4.3.3 Tabel Prodi

Tabel ini digunakan untuk menyimpan data program studi yang terdapat di universitas. Setiap prodi memiliki id unik, nama prodi, dan relasi ke tabel fakultas melalui kolom fakultas_id. Dengan demikian, setiap program studi dapat diketahui berasal dari fakultas mana. Gambar 4.19 menunjukan struktur tabel main_prodi.

Jelajahi | Struktur | SQL | Cari | Tambahkan | Ekspor | Impor | Hak Akses | Operasi | Trigger

Struktur tabel | Tampilan hubungan

#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan	Komentar	Ekstra	Tindakan
1	id	bigint(20)			Tidak	*Tidak ada*		AUTO_INCREMENT	Ubah Hapus Lainnya
2	nama	varchar(100)	utf8mb4_general_ci		Tidak	*Tidak ada*			Ubah Hapus Lainnya
3	fakultas_id	bigint(20)			Tidak	*Tidak ada*			Ubah Hapus Lainnya

Gambar 4.19 Implementasi Tabel Prodi

4.3.4 Tabel Pemilihan

Tabel ini berfungsi untuk menyimpan data setiap pemilihan yang diadakan pada sistem, seperti nama pemilihan, deskripsi, serta waktu mulai dan selesai pemilihan. Setiap pemilihan akan memiliki id unik sebagai identitasnya. Gambar 4.20 menunjukan struktur tabel main_pemilihan.

#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan	Komentar	Ekstra	Tindakan
1	id	bigint(20)			Tidak	*Tidak ada*		AUTO_INCREMENT	Ubah Hapus Lainnya
2	nama	varchar(255)	utf8mb4_general_ci		Tidak	*Tidak ada*			Ubah Hapus Lainnya
3	deskripsi	longtext	utf8mb4_general_ci		Tidak	*Tidak ada*			Ubah Hapus Lainnya
4	waktu_mulai	datetime(6)			Tidak	*Tidak ada*			Ubah Hapus Lainnya
5	waktu_selesai	datetime(6)			Tidak	*Tidak ada*			Ubah Hapus Lainnya

Gambar 4.20 Implementasi Tabel Pemilihan

4.3.5 Tabel Pemilihan Pemilih

Tabel ini digunakan untuk menyimpan relasi antara pemilihan dan daftar akun yang berhak menjadi pemilih pada pemilihan tersebut. Dengan demikian, setiap pemilihan akan memiliki daftar pemilih yang dapat berpartisipasi. Gambar 4.21 menunjukan struktur tabel main_pemilihan_pemilih.

#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan	Komentar	Ekstra	Tindakan
1	id	bigint(20)			Tidak	*Tidak ada*		AUTO_INCREMENT	Ubah Hapus Lainnya
2	pemilihan_id	bigint(20)			Tidak	*Tidak ada*			Ubah Hapus Lainnya
3	account_id	varchar(20)	utf8mb4_general_ci		Tidak	*Tidak ada*			Ubah Hapus Lainnya

Gambar 4.21 Implementasi Tabel Pemilihan Pemilih

4.3.6 Tabel Kandidat

Tabel main_kandidat digunakan untuk menyimpan data para kandidat yang mengikuti setiap pemilihan. Informasi yang disimpan pada tabel ini meliputi visi, misi, program kerja, foto kandidat, data ketua dan wakil (jika ada), relasi ke pemilihan, serta nomor urut kandidat. Setiap entri pada tabel ini merepresentasikan satu kandidat (atau pasangan calon) pada suatu pemilihan. Gambar 4.22 menunjukan struktur tabel main_kandidat.

#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan	Komentar	Ekstra	Tindakan
1	id	bigint(20)			Tidak	*Tidak ada*		AUTO_INCREMENT	Ubah Hapus Lainnya
2	visi	longtext	utf8mb4_general_ci		Tidak	*Tidak ada*			Ubah Hapus Lainnya
3	misi	longtext	utf8mb4_general_ci		Tidak	*Tidak ada*			Ubah Hapus Lainnya
4	proker	longtext	utf8mb4_general_ci		Tidak	*Tidak ada*			Ubah Hapus Lainnya
5	foto	varchar(100)	utf8mb4_general_ci		Ya	*NULL*			Ubah Hapus Lainnya
6	ketua_id	varchar(20)	utf8mb4_general_ci		Tidak	*Tidak ada*			Ubah Hapus Lainnya
7	pemilihan_id	bigint(20)			Tidak	*Tidak ada*			Ubah Hapus Lainnya
8	wakil_id	varchar(20)	utf8mb4_general_ci		Ya	*NULL*			Ubah Hapus Lainnya
9	nomor_urut	int(10)		UNSIGNED	Tidak	*Tidak ada*			Ubah Hapus Lainnya

Gambar 4. 22 Implementasi Tabel Kandidat

4.3.7 Tabel Suara

Tabel main_suara digunakan untuk menyimpan setiap suara yang masuk dalam proses pemilihan. Tabel ini mencatat waktu pemungutan suara, kandidat yang dipilih, pemilih, serta pemilihan yang diikuti. Selain itu, tabel ini juga menyimpan data tambahan seperti alamat IP, user agent, serta status kecurigaan beserta alasannya, yang berguna untuk mendeteksi indikasi kecurangan dalam proses *voting*. Gambar 4.23 menunjukan struktur tabel main_suara.

#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan	Komentar	Ekstra	Tindakan
1	id	bigint(20)			Tidak	*Tidak ada*		AUTO_INCREMENT	Ubah Hapus Lainnya
2	waktu	datetime(6)			Tidak	*Tidak ada*			Ubah Hapus Lainnya
3	kandidat_id	bigint(20)			Tidak	*Tidak ada*			Ubah Hapus Lainnya
4	pemilih_id	varchar(20)	utf8mb4_general_ci		Tidak	*Tidak ada*			Ubah Hapus Lainnya
5	pemilihan_id	bigint(20)			Tidak	*Tidak ada*			Ubah Hapus Lainnya
6	ip_address	char(39)	utf8mb4_general_ci		Ya	*NULL*			Ubah Hapus Lainnya
7	user_agent	longtext	utf8mb4_general_ci		Ya	*NULL*			Ubah Hapus Lainnya
8	is_suspicious	tinyint(1)			Tidak	*Tidak ada*			Ubah Hapus Lainnya
9	alasan_kecurangan	longtext	utf8mb4_general_ci		Ya	*NULL*			Ubah Hapus Lainnya

Gambar 4.23 Implementasi Tabel Suara

4.3.8 Tabel TokenVote

Tabel main_tokenvote berfungsi untuk menyimpan data token *voting* yang digunakan oleh pemilih dalam setiap pemilihan. Tabel ini mencatat token yang di-generate, status penggunaan token, waktu pembuatan, waktu penggunaan, masa kedaluwarsa, serta informasi pemilih dan pemilihan terkait. Dengan adanya tabel ini, sistem dapat memastikan bahwa setiap token hanya dapat digunakan satu kali dan mengontrol validitas token saat proses *voting*. Gambar 4.24 menunjukan struktur tabel main_tokenvote.

#	Nama	Jenis	Penyortiran	Atribut	Tak Ternilai	Bawaan	Komentar	Ekstra	Tindakan
1	id	bigint(20)			Tidak	*Tidak ada*		AUTO_INCREMENT	Ubah Hapus Lainnya
2	token	varchar(12)	utf8mb4_general_ci		Tidak	*Tidak ada*			Ubah Hapus Lainnya
3	digunakan	tinyint(1)			Tidak	*Tidak ada*			Ubah Hapus Lainnya
4	waktu_dibuat	datetime(6)			Tidak	*Tidak ada*			Ubah Hapus Lainnya
5	waktu_digunakan	datetime(6)			Ya	*NULL*			Ubah Hapus Lainnya
6	expired_at	datetime(6)			Tidak	*Tidak ada*			Ubah Hapus Lainnya
7	pemilih_id	varchar(20)	utf8mb4_general_ci		Tidak	*Tidak ada*			Ubah Hapus Lainnya
8	pemilihan_id	bigint(20)			Tidak	*Tidak ada*			Ubah Hapus Lainnya

Gambar 4.24 Implementasi Tabel TokenVote

4.4 FITUR-FITUR SISTEM

Sistem *e-voting* PEMIRA UNJAYA dikembangkan dengan berbagai fitur utama untuk mendukung kelancaran pemilihan umum mahasiswa secara online. Beberapa fitur penting yang tersedia meliputi manajemen data pemilih, pengelolaan kandidat, pembuatan dan pengaturan jadwal pemilihan, proses *voting* yang dilengkapi sistem token untuk keamanan, serta deteksi kecurangan pada setiap suara yang masuk. Selain itu, sistem menyediakan dashboard khusus untuk admin dan *voter*, fitur impor data pemilih, pengelolaan hasil suara dalam bentuk grafik dan tabel, serta notifikasi otomatis melalui email. Setiap fitur ini dirancang agar mudah digunakan, memperkuat transparansi, dan menjaga keamanan proses pemilihan.

4.4.1 Fitur Manajemen Data Pemilih

Fitur manajemen data pemilih pada sistem *e-voting* ini memungkinkan admin untuk mengelola informasi seluruh peserta pemilihan secara mudah dan efisien. Melalui fitur ini, admin dapat menambah, mengedit, menghapus, maupun mengimpor data pemilih secara massal dari file Excel. Selain itu, tersedia fungsi pencarian, filter berdasarkan angkatan, serta pengaturan status aktif atau tidaknya akun pemilih. Dengan adanya fitur ini, proses pengelolaan data pemilih menjadi lebih terstruktur dan meminimalisir kesalahan pendataan.

4.4.2 Pengelolaan Kandidat

Fitur pengelolaan kandidat memungkinkan admin untuk menambah, mengedit, dan menghapus data kandidat dalam setiap pemilihan. Admin dapat menginput informasi detail kandidat seperti NIM ketua dan wakil, visi, misi, program kerja, serta mengunggah foto kandidat. Proses validasi juga dilakukan untuk memastikan data yang dimasukkan sudah sesuai dan tidak terjadi duplikasi kandidat. Dengan fitur ini, data kandidat setiap pemilihan dapat diatur dan ditampilkan secara rapi, sehingga memudahkan proses verifikasi dan penyajian informasi kepada pemilih.

4.4.3 Pengelolaan Pemilihan

Fitur pengelolaan pemilihan memberikan kemudahan bagi admin untuk membuat, mengatur, dan memperbarui data setiap proses pemilihan yang berlangsung. Admin dapat menentukan nama, deskripsi, waktu mulai, dan waktu selesai pemilihan, serta melakukan pengeditan atau penghapusan jika diperlukan. Selain itu, fitur ini juga menyediakan tampilan daftar seluruh pemilihan beserta statusnya, sehingga admin dapat dengan mudah memantau dan mengelola jalannya berbagai pemilihan secara terstruktur dan terorganisir.

4.4.4 *Voting* Token Via Email

Fitur *voting* token via email berfungsi untuk meningkatkan keamanan dan validasi dalam proses pemilihan suara. Setiap pemilih yang telah ditentukan akan menerima token unik yang dikirimkan langsung ke email mereka. Token ini hanya dapat digunakan satu kali untuk memberikan suara pada periode pemilihan yang aktif. Dengan adanya mekanisme ini, hanya pemilih yang sah yang dapat mengikuti *voting*, sehingga mencegah terjadinya penyalahgunaan dan manipulasi suara.

4.5 HASIL PENGUJIAN SISTEM

Hasil pengujian menunjukkan bahwa seluruh fitur inti berjalan dengan baik dan sistem dapat digunakan dengan lancar sesuai skenario yang diharapkan.

4.5.1 Hasil Pengujian *Unit testing*

Hasil implementasi dari unit testing dapat dilihat pada tabel 4.1.

Tabel 4.1 Tabel Uji *Unit testing*

No	Skenario Pengujian	Input / Langkah	Expected Output	Status
1	Login berhasil	NIM dan *password* benar	Redirect ke beranda sesuai role	Lulus
2	Login gagal (*password* salah)	NIM benar, *password* salah	Pesan error "NIM atau *Password* salah"	Lulus
3	Login gagal (akun tidak aktif)	NIM & *password* benar, is_active=False	Pesan error "Akun Anda belum aktif"	Lulus

4	Logout berhasil	Klik logout dari halaman admin/*voter*	Redirect ke halaman login	Lulus
5	Cek proteksi akses tanpa login	Akses halaman dashboard tanpa login	Redirect ke halaman login	Lulus
6	Tambah data user baru oleh admin	Isi form tambah user	User baru berhasil ditambahkan	Lulus
7	Edit data user	Edit nama/email/role user	Data user berhasil diperbarui	Lulus
8	Hapus user oleh admin	Pilih user dan klik hapus	Data user berhasil dihapus	Lulus
9	Impor data user dari file	Upload file Excel dengan NIM/email	Data user berhasil diimpor	Lulus
10	Ubah *password* pemilih	Isi form ubah *password* dengan benar	*Password* berhasil diperbarui	Lulus
11	Ubah *password* gagal (*password* lama salah)	*Password* lama salah	Pesan error "*Password* saat ini salah"	Lulus
12	Update profil pemilih	Update data profil	Data berhasil diperbarui	Lulus
13	Cek upload foto profil	Update dengan foto baru	Foto profil berhasil diganti	Lulus
14	Validasi duplikasi NIM/email	Input NIM/email yang sudah ada	Pesan error "NIM/email sudah terdaftar"	Lulus
15	Deteksi kecurangan *voting* pada unit test	Simulasi IP yang sama digunakan berulang	Data suara dicatat sebagai "mencurigakan"	Lulus

Kesimpulan *Unit testing*:

Seluruh skenario pengujian unit berjalan dengan baik, menandakan fitur dasar dan validasi keamanan sistem sudah sesuai spesifikasi.

4.5.2 Hasil Pengujian *Integration testing*

Hasil implementasi dari integration testing dapat dilihat pada tabel 4.2.

Tabel 4.2 Tabel Uji *Integration testing*

No	Skenario Pengujian	Input / Langkah Utama	Expected Output	Status
1	Admin login & akses dashboard	Login admin	Dashboard tampil dan memuat statistik	Lulus
2	Admin membuat pemilihan dan kandidat	Form tambah pemilihan dan kandidat	Data pemilihan dan kandidat tersimpan	Lulus
3	Admin mengatur daftar pemilih	Pilih NIM untuk pemilihan	Daftar pemilih diperbarui	Lulus
4	*Voter* login dan minta token *voting*	Login *voter*, klik kirim token	Token terkirim ke email	Lulus
5	*Voter voting* dengan token valid	Submit suara + token valid	Suara terekam, token ditandai digunakan	Lulus
6	*Voting* dua kali dengan token yang sama	Submit suara dua kali	*Voting* kedua ditolak, pesan error muncul	Lulus
7	*Voting* dengan token kadaluwarsa	Token expired, submit *voting*	*Voting* ditolak, pesan error "Token kadaluwarsa"	Lulus
8	*Voting* dengan token tidak valid	Token palsu, submit *voting*	*Voting* ditolak, pesan error "Token tidak valid"	Lulus
9	Deteksi kecurangan: *voting* berulang IP	Simulasi *voter* ganti akun dengan IP sama	Suara ditandai "mencurigakan" di log kecurangan	Lulus
10	*Voting* di luar jadwal	Akses *voting* sebelum/selesai waktu	Tombol *voting* tidak aktif, *voting* ditolak	Lulus
11	*Voter* bukan daftar pemilih coba *voting*	Login *voter* yang tidak diatur sebagai pemilih	Akses ditolak, pesan error	Lulus
12	Admin tambah kandidat duplikat (nomor urut sama)	Nomor urut sudah ada	Kandidat tidak ditambah, pesan error	Lulus
13	Admin tambah kandidat ketua/wakil duplikat	Ketua/wakil sudah terdaftar di kandidat lain	Kandidat tidak ditambah, pesan error	Lulus

14	Notifikasi rekomendasi ganti *password* default (NIM)	Admin login dengan *password* = NIM	Rekomendasi muncul di dashboard admin	Lulus
15	Deteksi *voting* mencurigakan multi-device/location	*Voting* dari lokasi/device berbeda dalam waktu singkat	Suara terdaftar sebagai mencurigakan, log kecurangan terisi	Lulus

Kesimpulan *Integration testing*:

Pengujian integrasi membuktikan seluruh fitur utama, keamanan, serta deteksi kecurangan sistem *e-voting* bekerja secara terintegrasi dan dapat menangani skenario nyata serta kasus-kasus penyalahgunaan dengan efektif.

4.5.3 Hasil Pengujian *User Acceptance Testing (UAT)*

Pemilih yang mengisi kuesioner *user acceptance testing* berjumlah 10 orang. Tabel 4.3 menampilkan hasil *user acceptance testing* yang diperoleh dari responden terhadap setiap pertanyaan dalam kuesioner, di mana nilai jawaban dikalikan dengan bobot penilaian yang telah ditetapkan.

Tabel 4.3 Perhitungan Bobot Untuk Pemilih

Pernyataan	**Nilai Responden**					**Bobot**
	Sangat Tidak Sesuai x 1	**Tidak Sesuai x 2**	**Kurang sesuai x 3**	**Sesuai x 4**	**Sangat Sesuai x 5**	
1			1x3 = 3	8 x 4 = 32	26 x 5 = 130	165
2			1x3 = 3	11 x 4 = 44	23 x 5 = 115	162
3			1x3 = 3	10 x 4 = 40	24 x 5 = 120	163
4			1x3 = 3	9 x 4 = 36	25 x 5 = 125	164
5			1x3 = 3	5 x 4 = 20	29 x 5 = 145	168

6			3 x 3 = 9	11 x 4 = 44	21 x 5 = 105	158
7			1 x 3 = 3	5 x 4 = 20	29 x 5 = 145	168
8			1 x 3 = 3	11 x 4 = 44	23 x 5 = 115	162
9			4 x 3 = 12	8 x 4 = 32	23 x 5 = 115	159
10			3 x 3 = 9	11 x 4 = 44	21 x 5 = 105	158
11			1 x 3 = 3	6 x 4 = 24	28 x 5 = 140	167

Tabel 4.4 Persentase Pernyataan Kuesoner Pemilih

Pernyataan ke	Nilai Mean	Persentase	Kategori
1	165/35 = 4.7	4.7/5x100% = 94%	*Usability*
2	162/35 = 4.6	4.6/5x100% = 92%	
3	163/35 = 4.7	4.7/5x100% = 94%	Kemudahan
4	164/35 = 4.7	4.7/5x100% = 94%	
5	168/35 = 4.8	4.8/5x100% = 96%	Fungsionalitas
6	158/35 = 4.5	4.5/5x100% = 90%	
7	168/35 = 4.8	4.8/5x100% = 96%	
8	162/35 = 4.6	4.6/5x100% = 92%	Keamanan
9	159/35 = 4.5	4.5/5x100% = 90%	
10	158/35 = 4.5	4.5/5x100% = 90%	Reliabilitas
11	167/35 = 4.8	4.8/5x100% = 96%	

Pada tabel 4.4 dapat disimpulkan hasil perhitungan mean atau rata-rata persentase kuesoner pada Usability sebesar 93%, Kemudahan sebesar 94%, Fungsionalitas sebesar 94%, keamanan sebesar 91%, Reabilitas sebesar 93%, dan persentase keseluruhan adalah 93%, maka untuk halaman pengguna termasuk dalam kategori sangat sesuai.

Admin yang mengisi kuesioner *user acceptance testing* berjumlah 2 orang. Tabel 4.5 menampilkan hasil user acceptance testing yang diperoleh dari responden terhadap setiap pernyataan dalam kuesioner, di mana nilai jawaban dikalikan dengan bobot penilaian yang telah ditetapkan.

Tabel 4.5 Perhitungan Bobot Untuk Admin

Pernyataan	**Nilai Responden**					**Bobot**
	Sangat Tidak Sesuai x 1	**Tidak Sesuai x 2**	**Kurang sesuai x 3**	**Sesuai x 4**	**Sangat Sesuai x 5**	
1				1x4 = 4	4 x 5 = 20	24
2					5 x 5 = 25	25
3				2x4 = 8	3 x 5 = 15	23
4					5 x 5 = 25	25
5				1x4 = 4	4 x 5 = 20	24
6					5 x 5 = 25	25
7					5 x 5 = 25	25
8				1x4 = 4	4 x 5 = 20	24
9				1x4 = 4	4 x 5 = 20	24
10				2x4 = 8	3 x 5 = 15	23
11				2x4 = 8	3 x 5 = 15	23
12					5 x 5 = 25	25

Tabel 4.6 Persentase Pernyataan Kuesoner Admin

Pertanyaan ke	Nilai Mean	Persentase	Kategori
1	24/5 = 4.8	4.8/5x100% = 96%	*Usability*
2	25/5 = 5	5/5x100% = 100%	
3	23/5 = 4.6	4.6/5x100% = 92%	Kemudahan
4	25/5 = 5	5/5x100% = 100%	
5	24/5 = 4.8	4.8/5x100% = 96%	Fungsionalitas
6	25/5 = 5	5/5x100% = 100%	

7	25/5 = 5	5/5x100% = 100%	
8	24/5 = 4.8	4.8/5x100% = 96%	Keamanan
9	24/5 = 4.8	4.8/5x100% = 96%	
10	23/5 = 4.6	4.6/5x100% = 92%	Reliabilitas
11	23/5 = 4.6	4.6/5x100% = 92%	
12	25/5 = 5	5/5x100% = 100%	

Pada tabel 4.6 dapat disimpulkan hasil perhitungan mean atau rata-rata persentase kuesoner pada *Usability* sebesar 98%, Kemudahan sebesar 96%, Fungsionalitas sebesar 98.7%, keamanan sebesar 96%, Reabilitas sebesar 94.7%, dan persentase keseluruhan adalah 96.68%, maka untuk halaman Admin termasuk dalam kategori sangat sesuai.

4.6 PEMBAHASAN

Berdasarkan hasil pengujian unit, integrasi, serta UAT (*User Acceptance Testing*) yang telah dilakukan pada aplikasi *e-voting* Pemira UNJAYA, dapat disimpulkan bahwa sistem telah memenuhi kebutuhan pengguna baik dari sisi admin maupun pemilih. Uji UAT pada admin dan *voter* menunjukkan persentase tingkat kepuasan sangat tinggi, di mana aspek usability, kemudahan, keamanan, fungsionalitas, dan reliabilitas sistem mendapat nilai rata-rata di atas 90%.

Pengguna dari sisi admin menyatakan aplikasi mudah digunakan untuk mengelola data pemilih, kandidat, hingga proses monitoring hasil *voting*. Fitur-fitur tambahan seperti import data Excel, filter pencarian, hingga monitoring log kecurangan berjalan baik dan responsif. Untuk pemilih, mereka merasa tampilan sistem mudah dipahami, proses *voting* mudah, token *voting* dapat diterima dengan lancar di email, serta sistem mampu menjaga keamanan suara dan privasi data.

Seluruh skenario pengujian, termasuk simulasi kasus *voting* ganda, *voting* dengan token tidak valid, hingga pengelolaan data besar, dapat ditangani sistem secara efektif. Fitur deteksi kecurangan melalui log aktivitas dan analisis pola *voting* juga terbukti berfungsi baik, menambah nilai lebih dalam aspek keamanan sistem. Evaluasi ini menunjukkan sistem mampu dioperasikan dalam skala

organisasi mahasiswa secara efisien dan dapat memperkuat transparansi proses demokrasi di lingkungan kampus.

Namun, tetap terdapat beberapa catatan untuk pengembangan lanjutan, seperti pentingnya infrastruktur server yang stabil agar sistem selalu dapat diakses tanpa hambatan, serta kebutuhan penambahan modul dashboard statistik partisipasi agar panitia bisa memantau tren pemilih secara *real-time*. Selain itu, perlu sosialisasi lebih lanjut kepada seluruh pengguna sebelum implementasi sistem agar proses transisi ke *e-voting* berjalan optimal.