BAB 4
HASIL PENELITIAN

4.1 RINGKASAN HASIL PENELITIAN

Sistem rekomendasi yang dibuat mempunyai dua peran utama, yaitu admin dan *user*. Admin bertugas mengelola data buku secara penuh, mulai dari menambahkan, mengedit, menghapus, hingga mencari dan melihat data buku, dengan akses yang dibatasi melalui proses *login*. Sistem ini dibangun menggunakan *framework* Django berbasis Python, dan menggunakan PostgreSQL sebagai basis data. Sistem menghasilkan rekomendasi dengan menerapkan metode *Content-based filtering*, dimulai dari proses *preprocessing* yang mencakup *case folding*, *tokenizing*, *filtering*, dan *stemming*. Setelah itu, sistem menghitung bobot kata serta kesamaan antar data memakai algoritma TF-IDF dan *Cosine similarity* guna menentukan rekomendasi yang paling relevan.

Dataset yang digunakan berjumlah 500 buku, dikumpulkan dari situs Gramedia dengan bantuan ekstensi Chrome Easy Scraper untuk mengambil data seperti judul, penulis, dan gambar sampul. Sementara itu, informasi seperti genre dan deskripsi buku dikumpulkan secara manual melalui pencarian di Google dan situs resmi Gramedia. Sistem ini berhasil dibangun sebagai *platform* rekomendasi buku berbasis web yang mampu memberikan saran buku otomatis berdasarkan kemiripan konten antar buku.

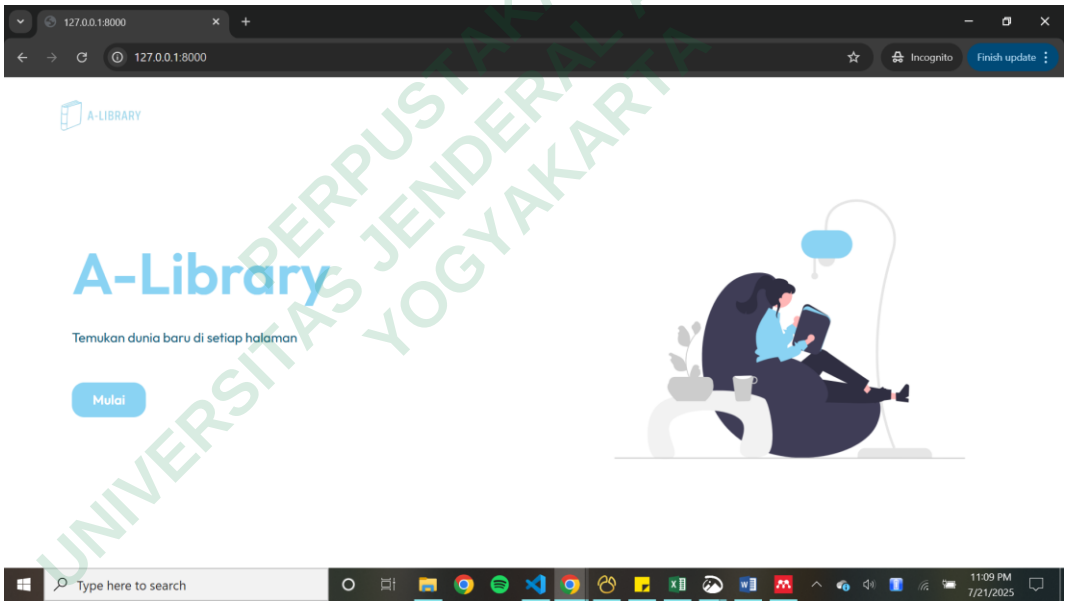
4.2 IMPLEMENTASI DESAIN ANTARMUKA

Tampilan pada aplikasi dirancang agar pengguna dapat berinteraksi dengan sistem secara mudah dan intuitif. Karena desain antarmuka ini menjadi bagian yang langsung dilihat dan digunakan oleh pengguna, maka perancangannya perlu dilakukan dengan cermat dan fungsional. Desain tampilan dari *platform* literasi berbasis *website* yang menerapkan algoritma *content-based filtering* ditujukan untuk memudahkan masyarakat umum dalam mengakses buku dan menerima rekomendasi buku.

Tampilan dalam *platform* ini terdiri dari berbagai halaman, antara lain: halaman beranda, *login*, *dashboard* admin, pencarian buku, *form* tambah dan edit buku, halaman utama, *list* buku, *filter* berdasarkan kategori, serta detail buku. Implementasi dari masing-masing halaman tersebut akan dijelaskan sebagai berikut.

4.2.1 Halaman Beranda

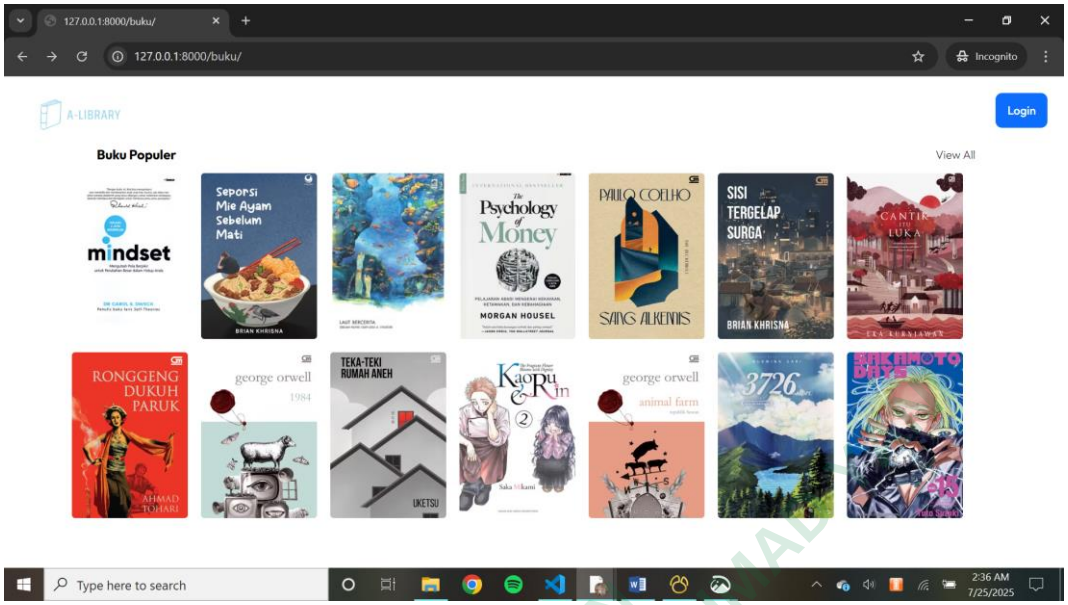
Halaman beranda ialah tampilan pertama yang tampak ketika program dijalankan dan berfungsi sebagai akses awal ke dalam sistem. Di halaman ini ada tombol "Mulai" yang jika ditekan akan memandu pengguna ke halaman *login* untuk mengakses fitur-fitur *platform*. Desain halaman ini dibuat sederhana dan informatif agar mudah dipahami dan memberikan kesan awal yang baik bagi pengguna. Implementasi halaman ini dapat dilihat pada Gambar 4.1.



Gambar 4.1 Halaman Beranda

4.2.2 Halaman Utama

Halaman utama ialah halaman di mana *user* dan admin setelah dari halaman beranda dapat melihat data buku populer. Implementasi halaman ini dapat dilihat pada Gambar 4.2.



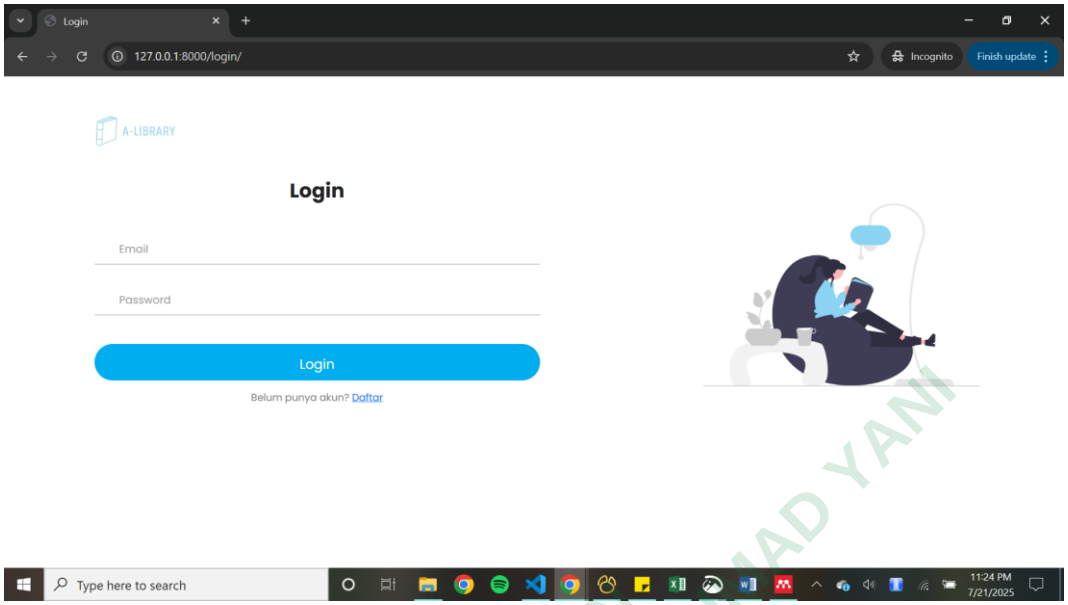
Gambar 4.2 Halaman Utama

Potongan kode di bawah ini ialah kode yang berfungsi untuk menampilkan data buku di halaman utama.

```
class BukuView(View):
    model = Buku
    template_name = 'buku/index.html'
    def get(self, request):
        buku_list = Buku.objects.all()
        return render(request, self.template_name, {'ebook':
            buku_list})
```

4.2.3 Halaman *Login*

Halaman *login* akan muncul setelah admin menekan tombol “Login” di halaman utama. Pada halaman ini, admin diminta untuk memasukkan *email* dan *password* agar dapat masuk ke dalam *dashboard* admin. Implementasi halaman ini dapat dilihat pada Gambar 4.3.



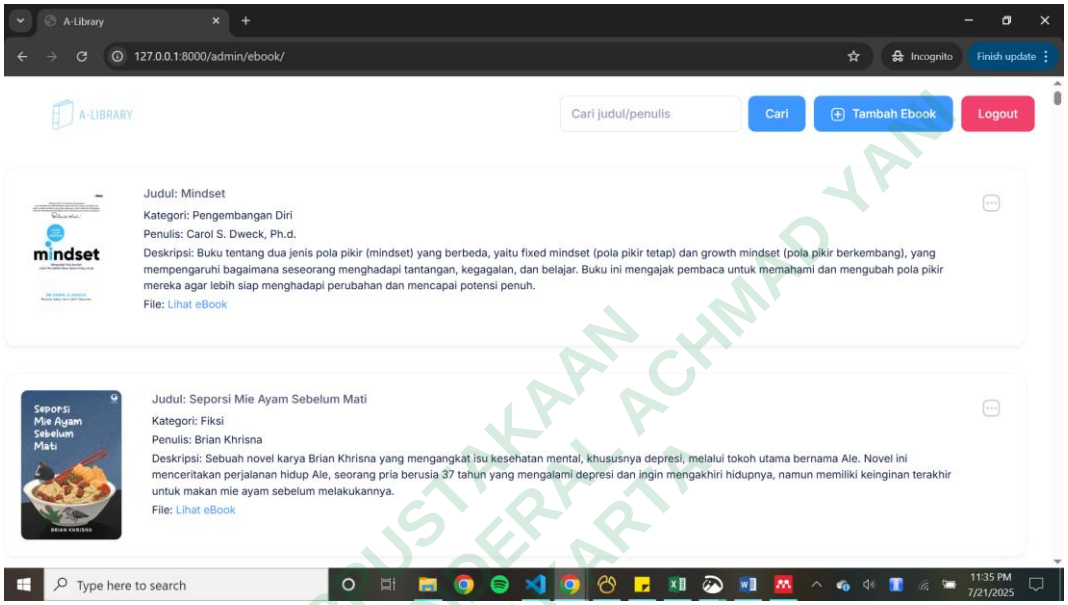
Gambar 4.3 Halaman *Login*

Potongan kode di bawah ini ialah kode yang berfungsi buat menampilkan *login*.

```
class LoginView(View):
    def get(self, request):
        form = LoginForm()
          return render(request, "signup/login.html", {"form":
form})
    def post(self, request):
        form = LoginForm(request.POST)
        if form.is_valid():
            email = form.cleaned_data.get('email')
             password = form.cleaned_data.get('pass1')
           user = authenticate(request, username=email,
           password=password)
            if user is not None:
                login(request, user)
              return redirect('view_ebook' if user.roles ==
           'admin' else 'buku')
           messages.error(request, "Email atau password salah.")
          return render(request, "signup/login.html", {"form":
form})
```

4.2.4 Halaman *Dashboard* Admin

Halaman *dashboard* admin memperlihatkan data buku yang ada di *database*. Di halaman ini, admin dapat menambah, mencari, mengubah, dan melakukan penghapusan data buku. Implementasi halaman ini dapat dilihat pada Gambar 4.4.



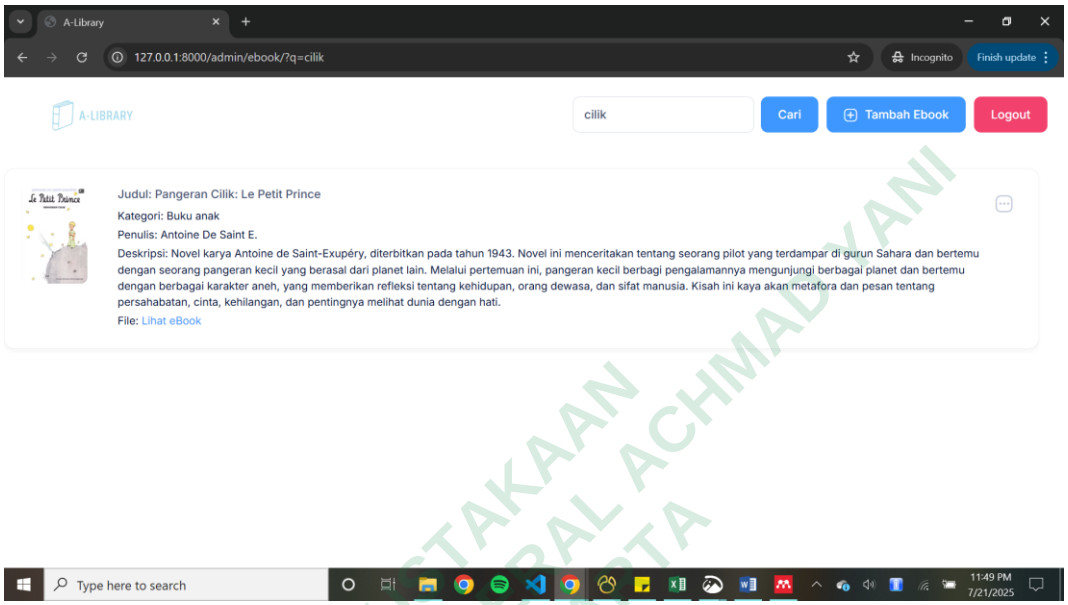
Gambar 4.4 Halaman *Dashboard* Admin

Potongan kode di bawah ini merupakan kode yang berfungsi untuk menampilkan halaman *dashboard* admin.

```
@method_decorator(login_required, name='dispatch')
class EbookView(View):
    def get(self, request):
        ebooks = Buku.objects.all()
        query = request.GET.get("q", "").strip()
        if query:
            ebooks =
           ebooks.filter(Q(judul__icontains=query) |
           Q(penulis__icontains=query))
        return render(request, 'admin/index_ebook.html', {
            'ebook': ebooks,
            'query': query,
            'show_search': True,
        })
```

4.2.5 Halaman Cari Buku

Halaman cari buku dapat membuat admin mencari data buku yang ada di halaman *dashboard* admin. Implementasi halaman ini ditampilkan pada Gambar 4.5.



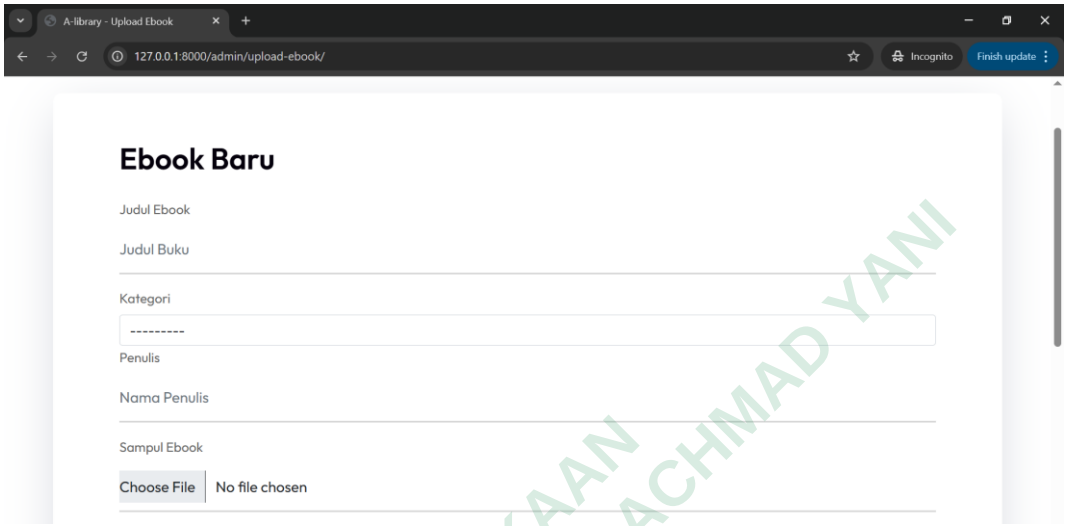
Gambar 4.5 Halaman Cari Buku

Potongan kode berikut berfungsi untuk mencari data buku menggunakan fitur *search bar*.

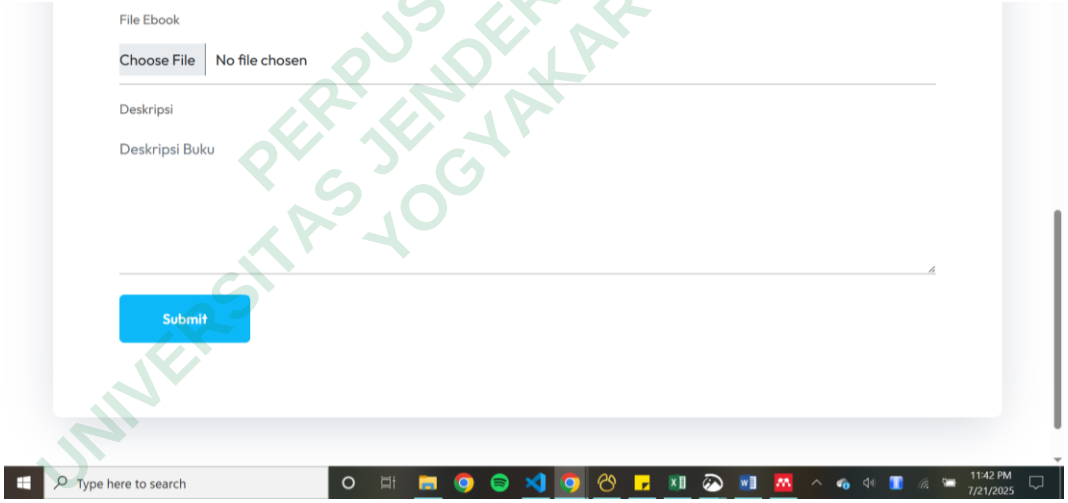
```
@method_decorator(login_required, name='dispatch')
class EbookView(View):
    def get(self, request):
        ebooks = Buku.objects.all()
        query = request.GET.get("q", "").strip()
        if query:
            ebooks =
           ebooks.filter(Q(judul__icontains=query) |
           Q(penulis__icontains=query))
        return render(request, 'admin/index_ebook.html', {
            'ebook': ebooks,
            'query': query,
            'show_search': True,
        })
```

4.2.6 Halaman *Form Upload* Buku

Halaman *form upload* buku dapat membuat admin menambah data buku baru ke *database*. Implementasi halaman ini ditampilkan pada Gambar 4.6 dan 4.7.



Gambar 4.6 Halaman *Form* tambah buku



Gambar 4.7 Halaman *Form* Tambah Buku

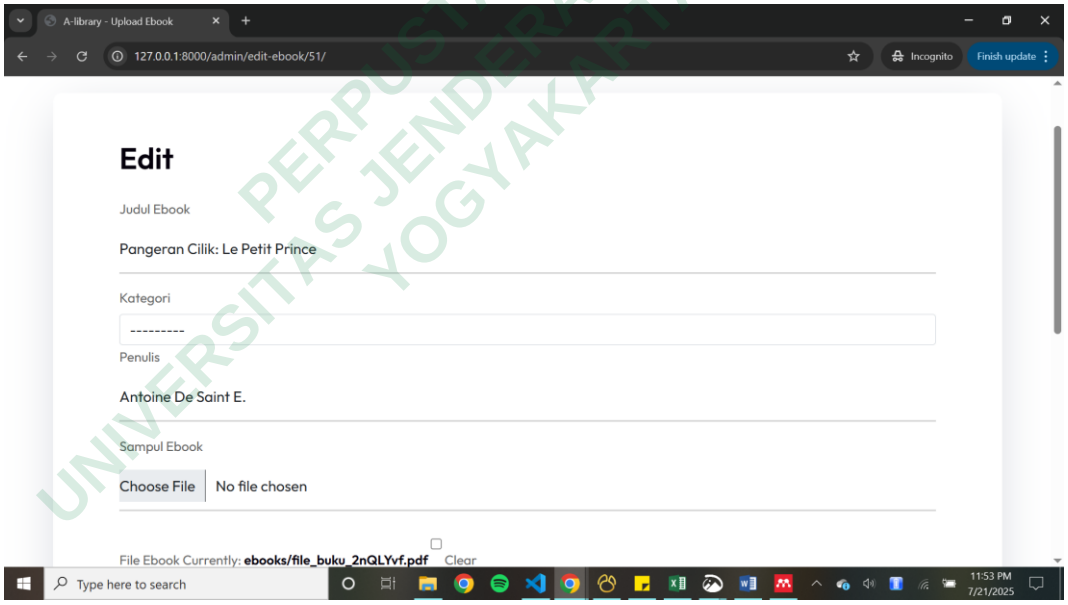
Potongan kode di bawah ini merupakan kode yang berfungsi untuk menambahkan data buku baru ke *database*.

```
@method_decorator(login_required, name='dispatch')
class EbookFormView(View):
    def get(self, request):
        form = BukuForm()
```

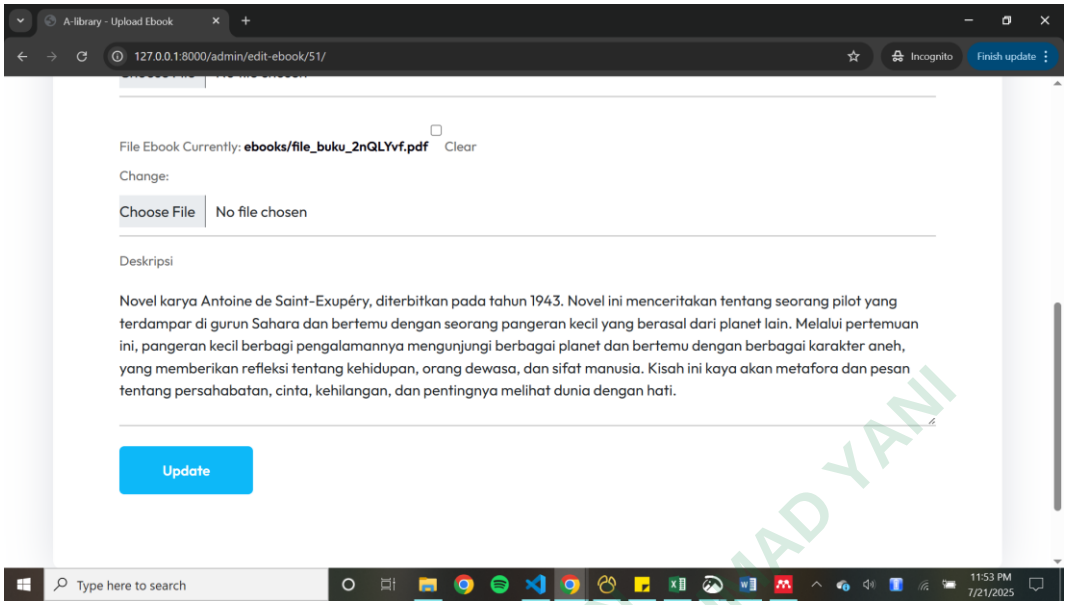
```
            return render(request, "buku/form.html", {"form":
            form})
        def post(self, request):
            form = BukuForm(request.POST, request.FILES)
            if form.is_valid():
                form.save()
                    messages.success(request, "Buku berhasil
                    disimpan.")
                return redirect('view_ebook')
            else:
                messages.error(request, "Terdapat kesalahan.
               Silakan periksa kembali.")
                  return render(request, "buku/form.html",
                  {"form": form})
```

4.2.7 Halaman *Form* Edit Buku

Halaman *form* edit buku dapat membuat admin mengedit data buku yang ada di *database* dan kemudian menyimpannya. Implementasi halaman ini ditampilkan pada Gambar 4.8 dan 4.9.



Gambar 4.8 Halaman *Form* Edit Buku



Gambar 4.9 Halaman *Form* Edit Buku

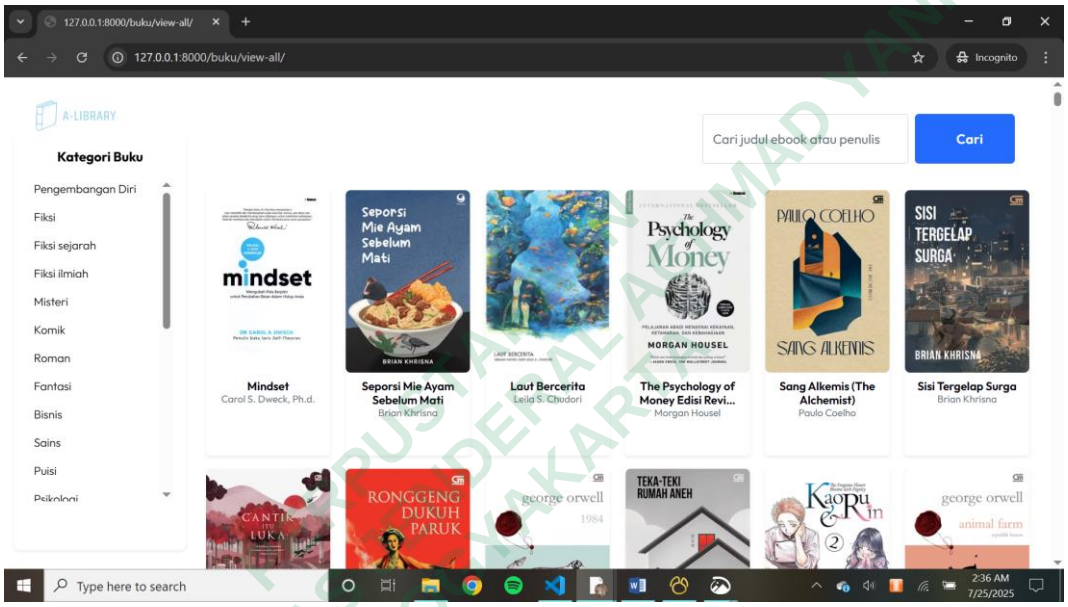
Potongan kode berikut berperan untuk mengubah data buku yang tersedia dan memperbaruinya ke *database*.

```
@method_decorator(login_required, name='dispatch')
class EbookUpdate(View):
    def get(self, request, id):
        ebook = get_object_or_404(Buku, id=id)
        form = BukuForm(instance=ebook)
        return render(request, 'buku/form.html', {'edit':
       True, 'form': form, 'ebook': ebook})
    def post(self, request, id):
        ebook = get_object_or_404(Buku, id=id)
            form = BukuForm(request.POST, request.FILES,
              instance=ebook)
        if form.is_valid():
            try:
                with transaction.atomic():
                    form.save()
                   messages.success(request, 'Buku berhasil
                      diperbarui.')
                    return redirect('view_ebook')
            except Exception as e:
                print('Error =', e)
                      messages.error(request, 'Buku gagal
                      diperbarui.')
        else:
```

```
                messages.error(request, 'Tolong periksa kembali
                data yang Anda masukkan.')
        return render(request, 'buku/form.html', {'edit': True,
    'form': form, 'ebook': ebook})
```

4.2.8 Halaman *List* Buku

Halaman *list* buku di mana *user* dapat mengaksesnya setelah klik "view all" di halaman utama. Dalam halaman ini, terdapat fitur pencarian, dan *sidebar* kategori buku. Implementasi halaman ini ditampilkan pada Gambar 4.10.



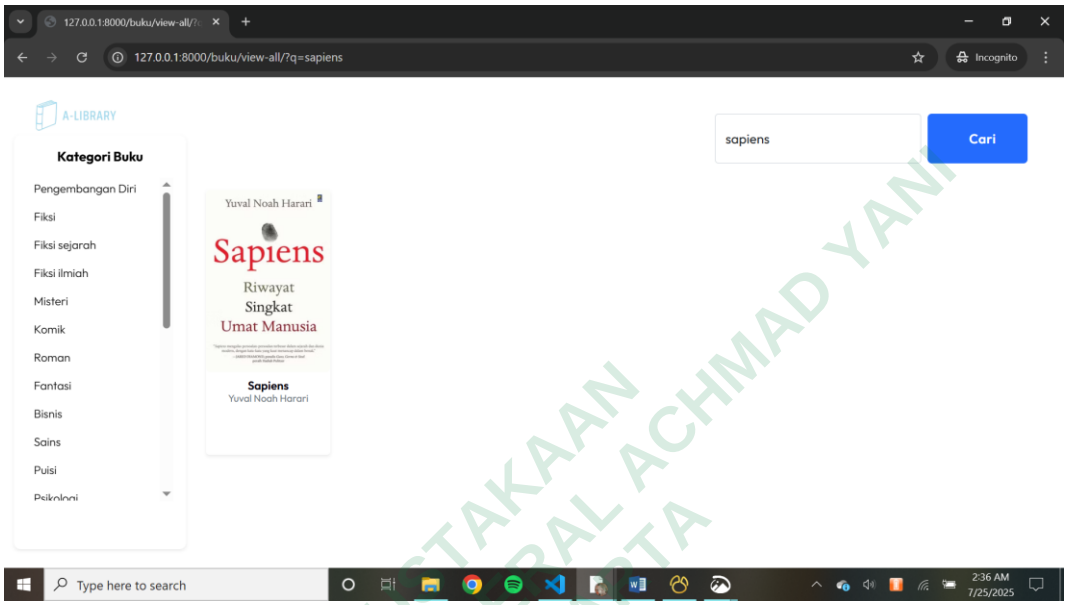
Gambar 4.10 Halaman *List* Buku

Potongan kode di bawah ini merupakan kode yang berfungsi untuk menampilkan *list* buku.

```
class BukuListView(View):
    def get(self, request):
        query = request.GET.get("q", "").strip()
        ebooks = Buku.objects.all()
        if query:
            ebooks = ebooks.filter(Q(judul__icontains=query)
              | Q(penulis__icontains=query))
        return render(request, 'buku/list.html', {
            'ebook': ebooks,
            'query': query,
            'show_search': True,
        })
```

4.2.9 Halaman Pencarian Buku

Halaman pencarian buku memungkinkan *user* menelusuri buku yang diinginkan dengan memasukkan judul atau nama penulis ke dalam fitur *search bar*. Implementasi halaman ini ditampilkan pada Gambar 4.11.



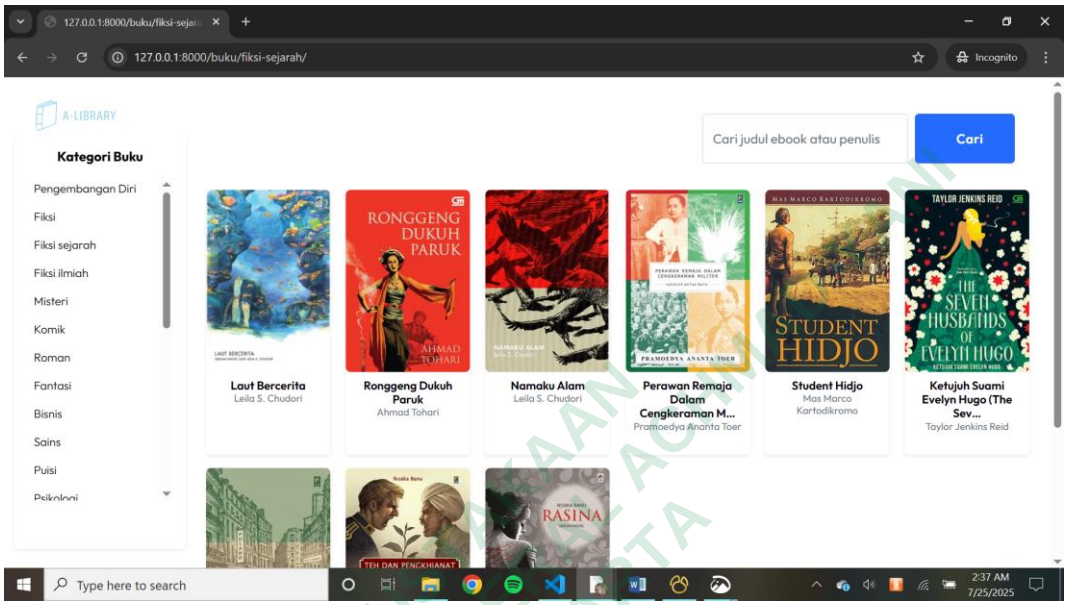
Gambar 4.11 Halaman Pencarian Buku

Potongan kode di bawah ini merupakan kode yang berfungsi untuk menampilkan buku yang ingin dicari berdasarkan kata kunci judul atau penulis buku.

```
class BukuListView(View):
    def get(self, request):
        query = request.GET.get("q", "").strip()
        ebooks = Buku.objects.all()
        if query:
            ebooks = ebooks.filter(Q(judul__icontains=query)
              | Q(penulis__icontains=query))
        return render(request, 'buku/list.html', {
            'ebook': ebooks,
            'query': query,
            'show_search': True,
        })
```

4.2.10 Halaman *Filter* buku berdasarkan kategori

Halaman *filter* buku berdasarkan kategori dapat membuat *user* melihat buku yang berada dalam satu kategori. Implementasi halaman ini ditampilkan pada Gambar 4.12.



Gambar 4.12 Halaman *Filter* Buku Berdasarkan Kategori

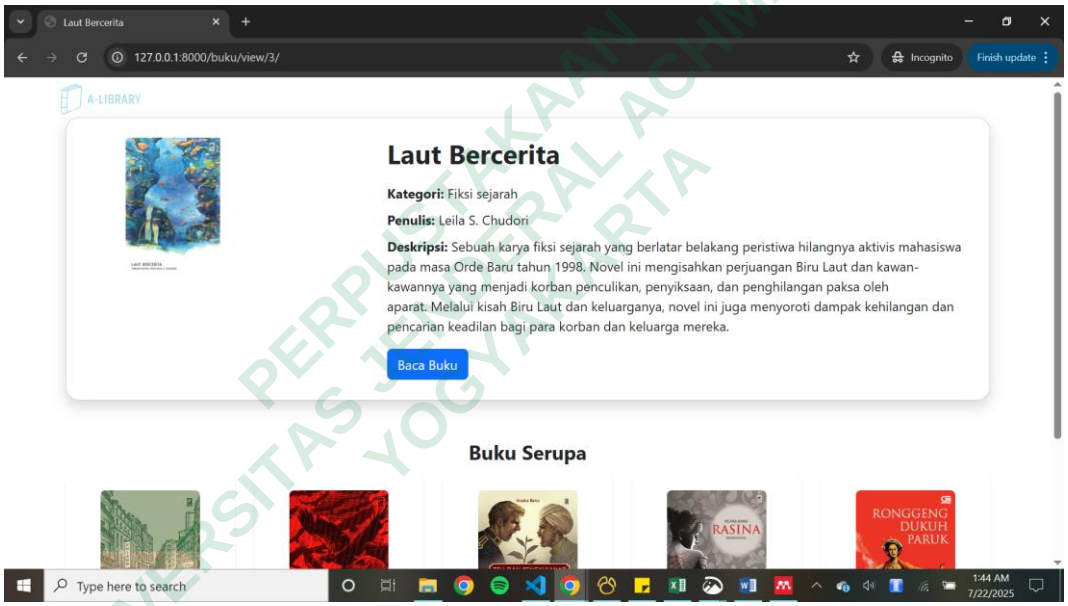
Potongan kode berikut berfungsi untuk menampilkan data buku sesuai dengan kategori yang dipilih.

```
class BukuCategoryView(View):
    def get(self, request, genre_slug):
        query = request.GET.get("q", "").strip()
        genre_name = GENRE_SLUG_MAP.get(genre_slug)
        if not genre_name:
            ebooks = []
            genre_name = 'Kategori Tidak Diketahui'
        else:
            ebooks =
           Buku.objects.filter(genre__iexact=genre_name)
            if query:
                ebooks = ebooks.filter(
                 Q(judul__icontains=query) |
                Q(penulis__icontains=query)
                )
        return render(request, 'buku/list.html', {
            'ebook': ebooks,
            'query': query,
```

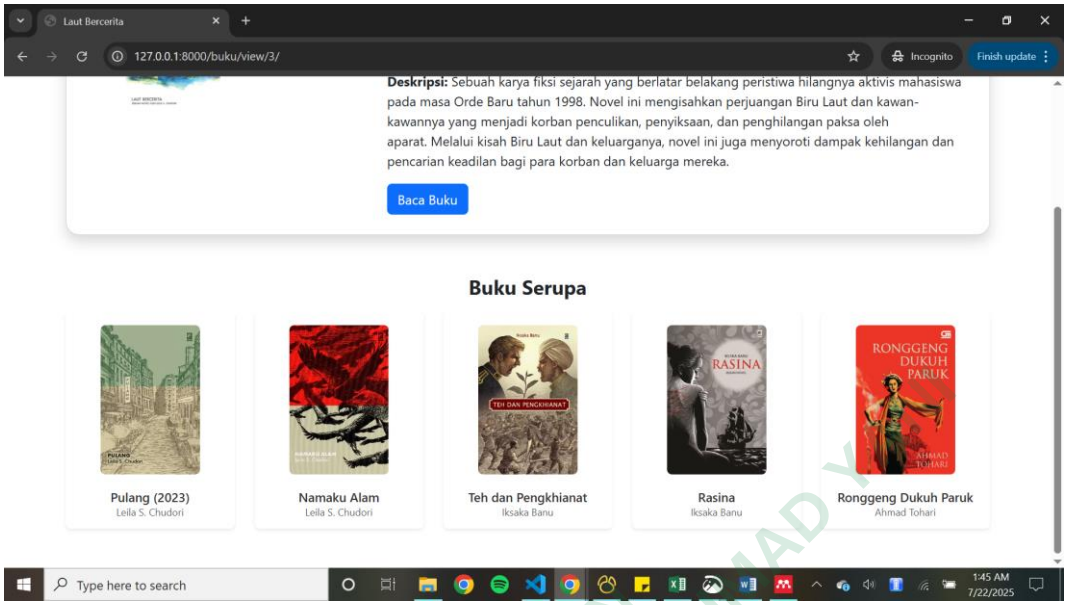
```
                'category_name': genre_name,
                'show_search': True,
                'genre_slug': genre_slug,
            })
```

4.2.11 Halaman Detail Buku

Halaman detail buku menyajikan data lengkap mengenai buku tersebut seperti judul, penulis, kategori dan deskripsi buku. Terdapat tombol "Baca Buku" yang akan mengarahkan *user* ke halaman yang menampilkan pdf buku dan *user* dapat mengunduhnya. Di halaman ini terdapat 5 rekomendasi buku serupa yang terletak di bawah detail buku. Implementasi halaman ini ditampilkan pada Gambar 4.13. dan 4.14.



Gambar 4.13 Halaman Detail Buku



Gambar 4.14 Halaman Detail Buku

Potongan kode di bawah ini ialah kode yang berfungsi menampilkan informasi buku yang bersangkutan dan memberikan 5 rekomendasi buku serupa.

```
class BukuDetail(View):
    def get(self, request, id):
        buku = get_object_or_404(Buku, id=id)
            model_path = os.path.join(settings.BASE_DIR,
            'ml_model')
        try:
            with open(os.path.join(model_path,
            'tfidf_vectorizer.pkl'), 'rb') as vec_file:
                vectorizer = pickle.load(vec_file)
            with open(os.path.join(model_path,
            'cosine_sim_matrix.pkl'), 'rb') as sim_file:
                cosine_sim = pickle.load(sim_file)
            semua_buku = list(Buku.objects.order_by('id'))
                index_buku = next(i for i, b in
            enumerate(semua_buku) if b.id == buku.id)
            skor_similaritas =
            list(enumerate(cosine_sim[index_buku]))
            skor_similaritas = sorted(skor_similaritas,
            key=lambda x: x[1], reverse=True)
            rekomendasi = []
            for i, score in skor_similaritas:
                if semua_buku[i].id != buku.id:
                    rekomendasi.append(semua_buku[i])
                if len(rekomendasi) == 5:
```

```
                    break
            except Exception as e:
                print(f"Error saat load rekomendasi real-time:
                {e}")
                rekomendasi = []
            return render(request, 'buku/details.html', {
                'buku': buku,
                'rekomendasi': rekomendasi,
            })
```

4.3 BASIS DATA

Basis data dimanfaatkan guna menyimpan seluruh informasi penting yang dibutuhkan dalam pembuatan dan pengembangan *platform* literasi berbasis web. Seluruh data tersebut disimpan menggunakan PostgreSQL. Di dalamnya terdapat dua basis data, yaitu "alibrary_admin_buku" dan "alibrary_frontend_user" yang masing-masing berisi tabel buku dan tabel pengguna yang berperan dalam mendukung jalannya sistem.

4.3.1 Tabel Buku

Tabel buku berperan dalam menyimpan data informasi buku yang dimasukkan oleh admin. Tabel buku berisi id, judul, penulis, genre, deskripsi, cover_url, cover_file, dan file_buku. Berikut adalah struktur tabel buku yang ditampilkan di gambar 4.15.

alibrary_admin_buku

	Field	Type	Extra
P	id	int8	
	judul	varchar(255) COLLATE "default"	
	penulis	varchar(255) COLLATE "default"	
	genre	varchar(50) COLLATE "default"	
	deskripsi	text COLLATE "default"	
	cover_url	varchar(200) COLLATE "default"	Allow Null
	cover_file	varchar(100) COLLATE "default"	Allow Null
	file_buku	varchar(100) COLLATE "default"	

Gambar 4.15 *Database* Tabel Buku

4.3.2 Tabel *User*

Tabel ini dipakai untuk mencatatkan data akun pengguna *website* ke basis data, yang terbagi menjadi dua jenis peran, yaitu admin dan *user*. Kolom-kolom yang terdapat dalam tabel *user* meliputi last_login, is_superuser, id_user, name,

email, password, roles, is_active, is_staff, dan date_joined. Struktur tabel *user* secara lengkap ditampilkan pada Gambar 4.16.

alibrary_frontend_user

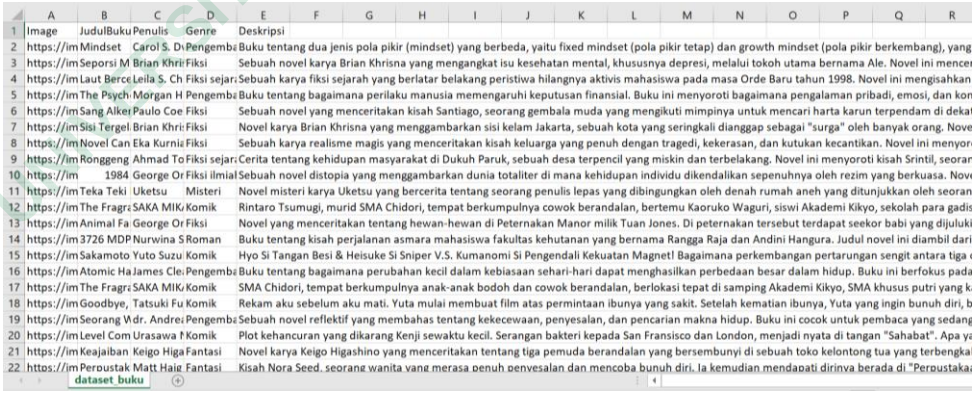
	Field	Type	Extra
	last_login	timestamptz(6)	Allow Null
	is_superuser	bool	
P	id_user	uuid	
	name	varchar(255) COLLATE "default"	
	email	varchar(255) COLLATE "default"	
	password	varchar(255) COLLATE "default"	
	roles	varchar(10) COLLATE "default"	
	is_active	bool	
	is_staff	bool	
	date_joined	timestamptz(6)	

Index	Fields	Extra
alibrary_frontend_user_email_01f7b6b1_like	email	

Gambar 4.16 *Database* Tabel *User*

4.4 IMPLEMENTASI SISTEM REKOMENDASI

Data yang dipakai di pembuatan sistem rekomendasi dengan algoritma *content-based filtering* diperoleh dari situs Gramedia dengan metode *web scraping* menggunakan ekstensi Chrome bernama Easy Scraper. Sebanyak 500 data buku berhasil dikumpulkan yang mencakup variabel penting seperti judul, penulis, dan *cover* buku sebagai dasar dalam membangun sistem rekomendasi. Sementara itu, informasi tambahan seperti genre dan deskripsi buku dikumpulkan secara manual melalui pencarian di Google dan situs resmi Gramedia. Dataset ditampilkan pada gambar 4.17.



	A	B	C	D	E
1	Image	JudulBuku	Penulis	Genre	Deskripsi
2	https://im	Mindset	Carol S. D	Pengemba	Buku tentang dua jenis pola pikir (mindset) yang berbeda, yaitu fixed mindset (pola pikir tetap) dan growth mindset (pola pikir berkembang), yang
3	https://im	Seporsi M	Brian Khri	Fiksi	Sebuah novel karya Brian Khrisna yang mengangkat isu kesehatan mental, khususnya depresi, melalui tokoh utama bernama Ale. Novel ini mencer
4	https://im	Laut Berce	Leila S. Ch	Fiksi sejar	Sebuah karya fiksi sejarah yang berlatar belakang peristiwa hilangnya aktivis mahasiswa pada masa Orde Baru tahun 1998. Novel ini mengisahkan
5	https://im	The Psych	Morgan H	Pengemba	Buku tentang bagaimana perilaku manusia memengaruhi keputusan finansial. Buku ini menyoroti bagaimana pengalaman pribadi, emosi, dan kon
6	https://im	Sang Alkei	Paulo Coe	Fiksi	Sebuah novel yang menceritakan kisah Santiago, seorang gembala muda yang mengikuti mimpinya untuk mencari harta karun terpendam di deka
7	https://im	Sisi Tergel	Brian Khri	Fiksi	Novel karya Brian Khrisna yang menggambarkan sisi kelam Jakarta, sebuah kota yang seringkali dianggap sebagai "surga" oleh banyak orang. Nove
8	https://im	Novel Can	Eka Kurnia	Fiksi	Sebuah karya realisme magis yang menceritakan kisah keluarga yang penuh dengan tragedi, kekerasan, dan kutukan kecantikan. Novel ini menyor
9	https://im	Ronggeng	Ahmad To	Fiksi sejar	Cerita tentang kehidupan masyarakat di Dukuh Paruk, sebuah desa terpencil yang miskin dan terbelakang. Novel ini menyoroti kisah Srintil, seorar
10	https://im	1984	George Or	Fiksi ilmial	Sebuah novel distopia yang menggambarkan dunia totaliter di mana kehidupan individu dikendalikan sepenuhnya oleh rezim yang berkuasa. Nov
11	https://im	Teka Teki	Uketsu	Misteri	Novel misteri karya Uketsu yang bercerita tentang seorang penulis lepas yang dibingungkan oleh denah rumah aneh yang ditunjukkan oleh seoran
12	https://im	The Fragra	SAKA MIK	Komik	Rintaro Tsumugi, murid SMA Chidori, tempat berkumpulnya cowok berandalan, bertemu Kaoruko Waguri, siswi Akademi Kikyo, sekolah para gadis
13	https://im	Animal Fa	George Or	Fiksi	Novel yang menceritakan tentang hewan-hewan di Peternakan Manor milik Tuan Jones. Di peternakan tersebut terdapat seekor babi yang dijuluki
14	https://im	3726 MDP	Nurwina S	Roman	Buku tentang kisah perjalanan asmara mahasiswa fakultas kehutanan yang bernama Rangga Raja dan Andini Hangura. Judul novel ini diambil dari
15	https://im	Sakamoto	Yuto Suzu	Komik	Hyo Si Tangan Besi & Heisuke Si Sniper V.S. Kumanomi Si Pengendali Kekuatan Magnet! Bagaimana perkembangan pertarungan sengit antara tiga
16	https://im	Atomic Ha	James Cle	Pengemba	Buku tentang bagaimana perubahan kecil dalam kebiasaan sehari-hari dapat menghasilkan perbedaan besar dalam hidup. Buku ini berfokus pada
17	https://im	The Fragra	SAKA MIK	Komik	SMA Chidori, tempat berkumpulnya anak-anak bodoh dan cowok berandalan, berlokasi tepat di samping Akademi Kikyo, SMA khusus putri yang k
18	https://im	Goodbye,	Tatsuki Fu	Komik	Rekam aku sebelum aku mati. Yuta mulai membuat film atas permintaan ibunya yang sakit. Setelah kematian ibunya, Yuta yang ingin bunuh diri, b
19	https://im	Seorang W	dr. Andrea	Pengemba	Sebuah novel reflektif yang membahas tentang kekecewaan, penyesalan, dan pencarian makna hidup. Buku ini cocok untuk pembaca yang sedang
20	https://im	Level Com	Urasawa N	Komik	Plot kehancuran yang dikarang Kenji sewaktu kecil. Serangan bakteri kepada San Fransisco dan London, menjadi nyata di tangan "Sahabat". Apa ya
21	https://im	Keajaiban	Keigo Higa	Fantasi	Novel karya Keigo Higashino yang menceritakan tentang tiga pemuda berandalan yang bersembunyi di sebuah toko kelontong tua yang terbengkal
22	https://im	Perpustak	Matt Haig	Fantasi	Kisah Nora Seed, seorang wanita yang merasa penuh penyesalan dan mencoba bunuh diri. Ia kemudian mendapati dirinya berada di "Perpustakaa

dataset_buku

Gambar 4.17 Dataset Buku

Tahap berikutnya adalah melakukan *preprocessing* pada dataset buku untuk menghapus elemen-elemen yang tidak relevan dalam deskripsi, seperti simbol,

angka, dan kata-kata yang kurang bermakna. Proses ini dilakukan secara bertahap dan berurutan guna mempersiapkan data sebelum dianalisis lebih lanjut.

4.4.1 *Case folding*

Case folding adalah proses yang dilakukan untuk menyeragamkan teks dengan memodifikasi seluruh huruf menjadi huruf kecil menggunakan metode lower() pada *string* Python. Tujuan dari proses ini adalah memastikan seluruh data teks berada dalam format yang konsisten (Albab et al., 2023). Implementasi kode untuk tahap ini dapat dilihat pada kode di bawah ini.

```
df["JudulBuku_clean"] = df["JudulBuku"].apply(lambda x: x.lower()
                        if isinstance(x, str) else x)
df["Penulis_clean"] = df["Penulis"].apply(lambda x: x.lower() if
                        isinstance(x, str) else x)
df["Genre_clean"]  =  df["Genre"].apply(lambda  x:  x.lower()  if
                        isinstance(x, str) else x)
df["Deskripsi_clean"] = df["Deskripsi"].apply(lambda x: x.lower()
                        if isinstance(x, str) else x)
```

Hasil dari kode *Case Folding* dapat dilihat pada gambar 4.18.

```
print(df[["JudulBuku_clean", "Penulis_clean", "Genre_clean", "Deskripsi_clean"]])
```

```
                          JudulBuku_clean                Penulis_clean  \
0                                 mindset         carol s. dweck, ph.d.
1           seporsi mie ayam sebelum mati                  brian khrisna
2                          laut bercerita              leila s. chudori
3     the psychology of money edisi revisi                 morgan housel
4             sang alkemis (the alchemist)                  paulo coelho
..                                     ...                           ...
495                 my hero academia 22               kohei horikoshi
496                      one piece 100                  eiichiro oda
497     demon slayer: kimetsu no yaiba 11              koyoharu gotouge
498      final fantasy : lost stranger 07  hazuki minase, itsuki kameya
499                awaken the giant within               anthony robbins

           Genre_clean                                  Deskripsi_clean
0    pengembangan diri  buku tentang dua jenis pola pikir (mindset) ya...
1                fiksi  sebuah novel karya brian khrisna yang mengangk...
2        fiksi sejarah  sebuah karya fiksi sejarah yang berlatar belak...
3    pengembangan diri  buku tentang bagaimana perilaku manusia memeng...
4                fiksi  sebuah novel yang menceritakan kisah santiago,...
..                 ...                                              ...
495             komik  latihan tarung gabungan kelas a versus kelas b...
496             komik  haoshoku. para pemeran utama telah berkumpul d...
497             komik  pertempuran di kota malam melawan iblis jyog...
498             komik  shogo, rei, dan alus terkena jebakan byblos da...
499  pengembangan diri  buku pengembangan diri yang mengajarkan bagaim...

[500 rows x 4 columns]
```

Gambar 4.18 Hasil *Case Folding*

Tahap berikutnya adalah penggabungan data. Setelah melalui proses *case folding*, data kemudian digabungkan ke dalam satu kolom baru yang diberi nama "combined_features". Kolom ini memuat gabungan dari beberapa elemen utama, yaitu judul buku, nama penulis, genre, dan deskripsi, yang nantinya akan digunakan sebagai dasar dalam pemberian bobot nilai yang dilakukan dengan TF-IDF, sedangkan perhitungan tingkat kemiripan antar buku menggunakan *Cosine similarity*. Selain itu, genre diulang 3 kali supaya bobotnya kuat untuk meningkatkan akurasi hasil rekomendasi. Kode program untuk penggabungan data tersebut dapat dilihat di bawah ini.

```
df['penulis_clean'] =
                    df['Penulis_clean'].str.lower().str.replace(r
                    '\s+', '_', regex=True)
df["combined_features"] = (
   df["JudulBuku_clean"].fillna('') + ' ' +
    df["penulis_clean"].fillna('') + ' ' +
```

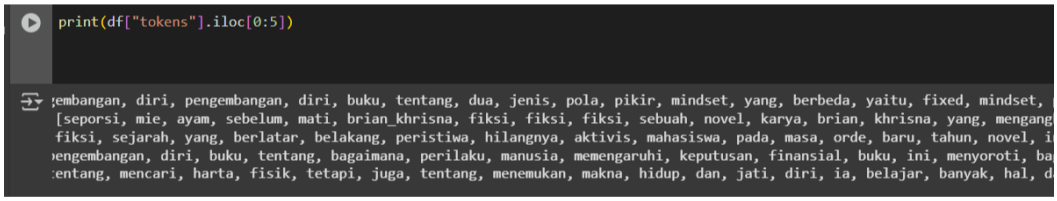
```
    df["Genre_clean"].fillna('') + ' ' +
    df["Genre_clean"].fillna('') + ' ' +
    df["Genre_clean"].fillna('') + ' ' +
    df["Deskripsi_clean"].fillna('')
)
```

4.4.2 *Tokenizing*

Tokenisasi merupakan proses memisahkan rangkaian karakter dalam teks menjadi unit-unit kata, dengan tujuan mengidentifikasi batas antar kata. Karakter seperti spasi, tab, dan enter umumnya dianggap sebagai pemisah, namun tanda baca seperti titik, koma, titik dua, dan tanda kutip dapat berperan ganda tergantung konteksnya. Penanganan karakter dalam proses ini sangat bergantung pada kebutuhan aplikasi yang dikembangkan. Tantangan dalam tokenisasi akan meningkat apabila struktur gramatikal bahasa juga perlu diperhatikan (Albab et al., 2023). Implementasi kode untuk tahap ini ditampilkan pada kode di bawah ini.

```
import re
def tokenize_and_clean(text):
    if isinstance(text, str):
        text = text.lower()
        text = re.sub(r"http\S+|www\S+|https\S+", "", text)
        text = text.encode('ascii', 'ignore').decode()
        text = re.sub(r"[\\\/@#^&*(){}[\]:;<>,.?~+=!0-9\"‟’']",
               " ", text)
        text = re.sub(r"\s+", " ", text).strip()
        tokens = text.split()
        tokens = [token for token in tokens if len(token) > 1]
        return tokens
    return []
df["tokens"] = df["combined_features"].apply(tokenize_and_clean)
```

Hasil dari kode *tokenizing* dapat dilihat pada gambar 4.19.



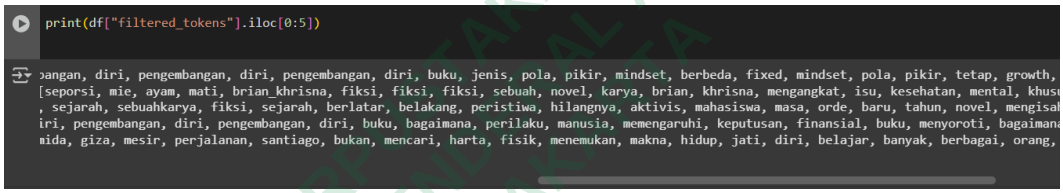
Gambar 4.19 Hasil *Tokenizing*

4.4.3 *Filtering*

Selanjutnya dilakukan proses filtering atau stopword removal, yaitu tahapan penyaringan untuk mengeliminasi kata-kata umum yang tidak memiliki nilai signifikan dalam representasi dokumen. Proses ini bertujuan untuk menyaring dan mempertahankan kata-kata penting dari hasil tokenisasi yang benar-benar merepresentasikan isi dokumen (Albab et al., 2023). Implementasi kode untuk tahap ini dapat dilihat pada kode di bawah ini.

```
from Sastrawi.StopWordRemover.StopWordRemoverFactory import StopWordRemoverFactory
stopwords = set(StopWordRemoverFactory().get_stop_words())
def filter_stopwords(token_list):
    return [token for token in token_list if token not in stopwords]
df["filtered_tokens"] = df["tokens"].apply(filter_stopwords)
```

Hasil dari kode filtering dapat dilihat pada gambar 4.20.



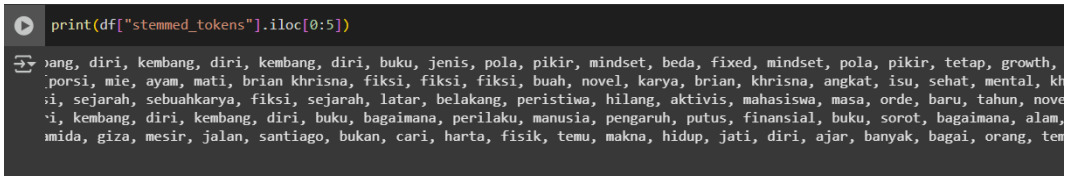
Gambar 4.20 Hasil *Filtering*

4.4.4 *Stemming*

Stemming merupakan proses untuk mengidentifikasi dan mengembalikan kata ke bentuk dasarnya dengan menghapus berbagai imbuhan yang melekat. Tujuan dari proses ini adalah menyederhanakan variasi kata menjadi satu bentuk dasar yang mewakili makna inti dari kata tersebut (Albab et al., 2023). Implementasi kode untuk tahap ini dapat dilihat pada kode di bawah ini.

```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
stemmer = StemmerFactory().create_stemmer()
def stem_tokens(tokens):
    return [stemmer.stem(token) for token in tokens]
df["stemmed_tokens"] = df["filtered_tokens"].apply(stem_tokens)
```

Hasil kode Stemming dapat dilihat pada gambar 4.21.



Gambar 4.21 Hasil *Stemming*

4.5 PERHITUNGAN KEMIRIPAN

Tahap akhir dalam proses ini adalah menghitung tingkat kemiripan antar buku. Berdasarkan hasil perhitungan tersebut, sistem dapat menghasilkan rekomendasi dengan mempertimbangkan sejauh mana kemiripan suatu buku dengan buku yang lain.

4.5.1 Pembobotan Nilai TF-IDF

Dalam proses pembobotan nilai TF-IDF, sistem membuat objek dari kelas TfidfVectorizer dan menyimpannya di variabel bernama tfidf. Selanjutnya, *method* fit_transform pada objek tersebut dipanggil guna menghasilkan matriks TF-IDF berdasarkan data di kolom “filtered_tokens”. Implementasi kode untuk tahap ini dapat dilihat pada kode di bawah ini.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from Sastrawi.StopWordRemover.StopWordRemoverFactory import StopWordRemoverFactory
# Inisialisasi stopwords Sastrawi
stop_factory = StopWordRemoverFactory()
list_stopwords = stop_factory.get_stop_words()
# Gabungkan token jadi string untuk TF-IDF input
df['books'] = df['filtered_tokens'].apply(lambda x: ' '.join(x))
# Inisialisasi TF-IDF dengan stopwords dari Sastrawi
tfidf = TfidfVectorizer(
    stop_words=list_stopwords,
    max_df=0.8
    )
# Fit dan transform kolom 'books'
tfidf_matrix = tfidf.fit_transform(df['books'])
# Cek shape matrix TF-IDF
print(tfidf_matrix.shape)
```

Hasil kode pembobotan nilai menggunakan tf-idf dapat dilihat pada gambar 4.22.

```
print(tfidf_matrix)

<Compressed Sparse Row sparse matrix of dtype 'float64'
        with 19808 stored elements and shape (500, 6330)>
  Coords        Values
  (0, 3849)     0.5042462977946794
  (0, 879)      0.13441620367506393
  (0, 11)       0.13441620367506393
  (0, 44)       0.13441620367506393
  (0, 4411)     0.18088020337843097
  (0, 1319)     0.15172109979826284
  (0, 835)      0.07240699756150175
  (0, 2219)     0.09584762438069369
  (0, 4709)     0.42340615071965104
  (0, 4664)     0.4805354827099534
  (0, 556)      0.08697132042988073
  (0, 1617)     0.13441620367506393
  (0, 5882)     0.09432062583585438
  (0, 1782)     0.13441620367506393
  (0, 618)      0.10585153767991276
  (0, 3369)     0.12013387067748835
  (0, 401)      0.05740133061430906
  (0, 5320)     0.09584762438069369
  (0, 3587)     0.13378093871388672
  (0, 5659)     0.08156529138311809
  (0, 2496)     0.11553598642503192
  (0, 502)      0.08078765000073043
  (0, 3510)     0.07861669120348663
  (0, 4292)     0.06728295838554249
```

Gambar 4.22 Hasil Tf-Idf

4.5.2 *Cosine Similarity*

Cosine similarity digunakan sebagai metode guna menakar tingkat kesamaan antara dua atau lebih dokumen teks. Dalam implementasinya, fungsi linear_kernel() dimanfaatkan untuk menghitung nilai kemiripan berbasis vektor dari matriks TF-IDF (tfidf_matrix). ”. Implementasi kode untuk tahap ini dapat dilihat pada kode di bawah ini.

```
from sklearn.metrics.pairwise import linear_kernel
cosine_sim=linear_kernel(tfidf_matrix, tfidf_matrix)
books=df.reset_index()
titles=books['JudulBuku']
description=books['Deskripsi']
```

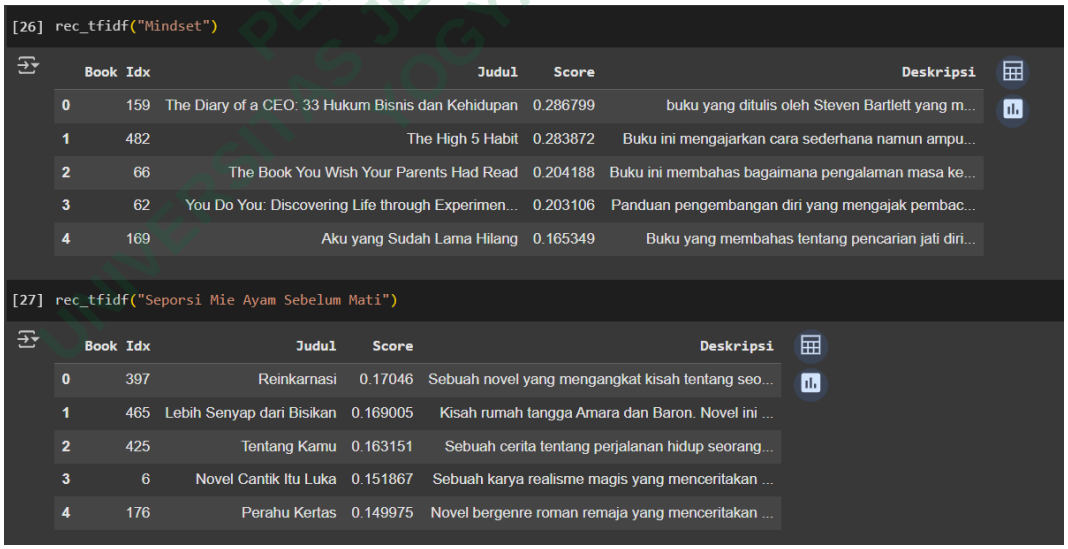
4.6 EVALUASI SISTEM REKOMENDASI

Selanjutnya, sistem akan memberikan tampilan rekomendasi buku yang mempunyai tingkat kesamaan tinggi satu sama lain. Implementasi kode ini sebagai berikut.

```
def rec_tfidf(JudulBuku, cosine_sim=cosine_sim):
      recommendation = pd.DataFrame(columns=['Book Idx',
'Judul', 'Score', 'Deskripsi'])
```

```
        count = 0
        idx = indice[JudulBuku]
        sim_scores = list(enumerate(cosine_sim[idx]))
         sim_scores = sorted(sim_scores, key=lambda x: x[1],
    reverse=True)
        sim_scores = sim_scores[1:6]
        book_indices = [i[0] for i in sim_scores]
        for i in book_indices:
            recommendation.at[count, 'Book Idx'] = i
            recommendation.at[count, 'Judul'] = titles.iloc[i]
                        recommendation.at[count,    'Score']    =
    dict(sim_scores)[i]
                                        recommendation.at[count,
    'Deskripsi']=description.iloc[i]
            count += 1
        return recommendation
```

Sistem mampu menghasilkan rekomendasi buku dengan menerima *input* berupa judul buku. Apabila judul tersebut terdapat di basis data, maka sistem akan memperlihatkan lima buku yang paling mirip berdasarkan perhitungan kemiripan, seperti yang ditampilkan pada Gambar 4.23. dan Gambar 4.24. Setelah rekomendasi ditampilkan, kinerja sistem dapat dievaluasi menggunakan metode *precision.*



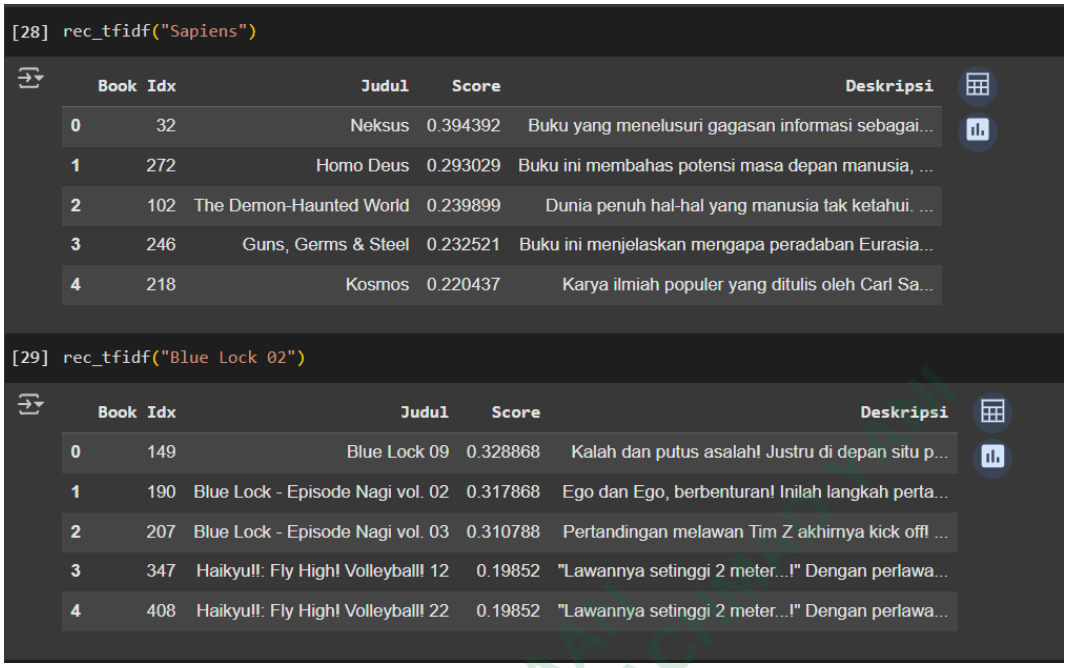
```
[26] rec_tfidf("Mindset")
```

	Book Idx	Judul	Score	Deskripsi
0	159	The Diary of a CEO: 33 Hukum Bisnis dan Kehidupan	0.286799	buku yang ditulis oleh Steven Bartlett yang m...
1	482	The High 5 Habit	0.283872	Buku ini mengajarkan cara sederhana namun ampu...
2	66	The Book You Wish Your Parents Had Read	0.204188	Buku ini membahas bagaimana pengalaman masa ke...
3	62	You Do You: Discovering Life through Experimen...	0.203106	Panduan pengembangan diri yang mengajak pembac...
4	169	Aku yang Sudah Lama Hilang	0.165349	Buku yang membahas tentang pencarian jati diri...

```
[27] rec_tfidf("Seporsi Mie Ayam Sebelum Mati")
```

	Book Idx	Judul	Score	Deskripsi
0	397	Reinkarnasi	0.17046	Sebuah novel yang mengangkat kisah tentang seo...
1	465	Lebih Senyap dari Bisikan	0.169005	Kisah rumah tangga Amara dan Baron. Novel ini ...
2	425	Tentang Kamu	0.163151	Sebuah cerita tentang perjalanan hidup seorang...
3	6	Novel Cantik Itu Luka	0.151867	Sebuah karya realisme magis yang menceritakan ...
4	176	Perahu Kertas	0.149975	Novel bergenre roman remaja yang menceritakan ...

Gambar 4.23 Hasil Sistem Rekomendasi Buku



[28] rec_tfidf("Sapiens")

	Book Idx	Judul	Score	Deskripsi
0	32	Neksus	0.394392	Buku yang menelusuri gagasan informasi sebagai...
1	272	Homo Deus	0.293029	Buku ini membahas potensi masa depan manusia, ...
2	102	The Demon-Haunted World	0.239899	Dunia penuh hal-hal yang manusia tak ketahui. ...
3	246	Guns, Germs & Steel	0.232521	Buku ini menjelaskan mengapa peradaban Eurasia...
4	218	Kosmos	0.220437	Karya ilmiah populer yang ditulis oleh Carl Sa...

[29] rec_tfidf("Blue Lock 02")

	Book Idx	Judul	Score	Deskripsi
0	149	Blue Lock 09	0.328868	Kalah dan putus asalah! Justru di depan situ p...
1	190	Blue Lock - Episode Nagi vol. 02	0.317868	Ego dan Ego, berbenturan! Inilah langkah perta...
2	207	Blue Lock - Episode Nagi vol. 03	0.310788	Pertandingan melawan Tim Z akhirnya kick off! ...
3	347	Haikyu!!: Fly High! Volleyball! 12	0.19852	"Lawannya setinggi 2 meter...!" Dengan perlawa...
4	408	Haikyu!!: Fly High! Volleyball! 22	0.19852	"Lawannya setinggi 2 meter...!" Dengan perlawa...

Gambar 4.24 Hasil Sistem Rekomendasi Buku

Dalam gambar 4.23 dan 4.24. menampilkan bahwa judul buku teratas memiliki *score cosine similarity* paling tinggi, yaitu yang paling mendekati 1 maka dianggap yang paling mirip. Dalam gambar 4.23. buku yang dicari rekomendasi serupa ialah Mindset, yang menampilkan The Diary of A CEO: 33 Hukum Bisnis dan Kehidupan sebagai buku yang paling mirip dengan buku Mindset, buku ini memiliki *cosine similarity score* sebesar 0.28679. Dalam gambar 4.23. buku yang dicari rekomendasi serupa ialah Seporsi Mie Ayam Sebelum Mati, yang menampilkan Reinkarnasi sebagai buku yang paling mirip dengan buku Seporsi Mie Ayam Sebelum Mati, buku ini memiliki *cosine similarity score* sebesar 0.17046. Dalam gambar 4.24. buku yang dicari rekomendasi serupa ialah Sapiens, yang menampilkan Neksus sebagai buku yang paling mirip dengan buku Sapiens, buku ini memiliki *cosine similarity score* sebesar 0.394392. Dalam gambar 4.24. buku yang dicari rekomendasi serupa ialah Blue Lock 02, yang menampilkan Blue Lock 09 sebagai buku yang paling mirip dengan buku Blue Lock 02, buku ini memiliki *cosine similarity score* sebesar 0.328868.

4.7 EVALUASI KINERJA SISTEM REKOMENDASI

Evaluasi sistem rekomendasi dilakukan dengan menggunakan metode *precision* dengan mengambil 20 sampel judul buku untuk mengevaluasi tingkat akurasi rekomendasi yang dihasilkan. Penilaian dilakukan secara manual terhadap lima rekomendasi teratas yang ditampilkan oleh sistem. Semakin tinggi tingkat relevansi antara buku yang direkomendasikan dengan buku acuan, maka semakin tinggi pula nilai *precision* yang diperoleh.

Tabel 4.1 Hasil *Precision* Sistem Rekomendasi

No	Data uji (Judul Buku)	Jumlah data relevan	*Precision*
1	Laut Bercerita	5/5	100%
2	The Psychology of Money	4/5	80%
3	Cantik itu Luka	5/5	100%
4	Atomic Habits	4/5	80%
5	Sakamoto days 15	4/5	80%
6	Petualangan Tintin: Si Kuping Belah	5/5	100%
7	The Fragrant Flower Blooms with Dignity-Kaoru & Rin 2	5/5	100%
8	Keajaiban Toko Kelontong Namiya	4/5	80%
9	Salvation of a Saint	5/5	100%
10	Novel Mawar Tak Berduri (Sad Cypress)	5/5	100%
11	One Piece 108	5/5	100%
12	Funiculi Funicula	4/5	80%
13	Neksus	5/5	100%
14	Moriarty the Patriot 18	4/5	80%
15	Seorang Pria yang Melalui Duka dengan Mencuci Piring	5/5	100%
16	Is It Bad or Good Habits	5/5	100%

No	Data uji (Judul Buku)	Jumlah data relevan	*Precision*
17	The Apothecary Diaries 13	5/5	100%
18	Harry Potter Dan Batu Bertuah	5/5	100%
19	English Classics: Anne of Green Gables	4/5	100%
20	Lima Sekawan: Beraksi Kembali	5/5	100%
Rata-rata *precision*			94%

Pengujian sistem rekomendasi *content-based filtering* dengan algoritma TF-IDF dan *Cosine similarity* yang ada pada tabel 4.1 menunjukkan bahwa sistem mampu mencapai rata-rata nilai *precision* sebesar 94%. Hasil ini mengindikasikan bahwa sistem memiliki performa yang baik dalam memberikan rekomendasi buku berdasarkan tingkat kemiripan informasi yang dimiliki masing-masing buku.

4.8 PENGUJIAN SISTEM

Pengujian memakai metode *black box* yang menitikberatkan pada aspek fungsional aplikasi. Metode ini bertujuan memastikan semua fitur aplikasi berfungsi sesuai harapan. Dalam pengujian ini, proses dilakukan oleh pengembang sistem. Hasil dari pengujian aplikasi yang telah melalui tahap analisis dan implementasi ditampilkan pada Tabel 4.2 dan Tabel 4.3.

Tabel 4.2 Hasil Pengujian *Website* Menggunakan Akun Admin

No	Pengguna	*Activity diagram*	Proses Pengujian	Hasil
1	Admin	*Login*	*Login* apabila *email* salah tidak dapat masuk	Berhasil
			Login apabila *password* salah tidak dapat masuk	
			Login apabila *email* dan *password* salah tidak dapat masuk	
			Login apabila *email* dan *password* benar dapat masuk	

2	Admin	Mengelola data buku	Menambah data buku baru	Berhasil
			Menyimpan data buku baru	
			Mencari data buku	
			Mengubah data buku	
			Menyimpan pembaruan data buku	
			Menghapus data buku	
			Membaca buku	
			Mengunduh buku	

Tabel 4.3 Hasil Pengujian *Website* Menggunakan Akun *User*

No	Pengguna	*Activity diagram*	Proses Pengujian	Hasil
1	*User*	*Signup*	Mengisi data untuk daftar akun	Berhasil
			Akun berhasil dibuat	
2	*User*	Melihat data buku	Melihat buku di halaman utama	Berhasil
			Melihat buku di halaman *list* buku	
			Melihat buku berdasarkan kategori	
			Mencari buku	
			Melihat detail buku	
			Menerima rekomendasi 5 buku serupa	
			Membaca file buku	
			Mengunduh file buku	

Berdasarkan hasil pengujian, dapat diambil kesimpulan bahwa seluruh fitur di *website* berjalan dengan lancar dan sesuai harapan. Pengujian yang dilakukan oleh pengembang dilakukan secara teliti dan menyeluruh terhadap fungsi-fungsi utama. Secara keseluruhan, *website* berjalan sesuai dengan desain dan implementasi yang telah direncanakan.

4.9 PEMBAHASAN

Berdasarkan hasil analisis, perancangan, implementasi, serta pengujian *website* dalam penelitian ini, diperoleh informasi bahwa *platform* literasi berbasis *website* berhasil dikembangkan sebagai solusi yang bermanfaat untuk membantu mengatasi permasalahan rendahnya tingkat literasi di Indonesia dengan menyediakan akses buku gratis. Penelitian ini dimulai dengan analisis kebutuhan pengguna, dilanjutkan dengan perancangan antarmuka, kemudian diimplementasikan menjadi sebuah *website* yang diuji secara fungsional untuk memastikan kesesuaian dengan tujuan awal. Selain itu, sistem rekomendasi yang memakai metode *Content-based filtering* menunjukkan performa luar biasa, dengan rata-rata *precision* sebesar 94%, menandakan sistem mampu memberikan rekomendasi buku yang akurat serta sesuai berdasarkan kemiripan konten.

Keterbatasan dari sistem ini terletak pada kemampuannya yang belum dapat menghasilkan rekomendasi untuk buku yang baru diunggah, karena data tersebut belum melalui proses pengolahan dalam dataset sehingga belum memiliki nilai TF-IDF dan *cosine similarity* yang dibutuhkan untuk merekomendasikan buku serupa. Selain itu, metode *content-based filtering* cenderung menghasilkan rekomendasi yang bersifat repetitif atau terlalu mirip, sehingga kurang mampu menyajikan variasi atau diversifikasi dalam daftar buku yang direkomendasikan.