1. Dokumen yang dibutuhkan berdasarkan FR

Berdasarkan dokumen dalam Functional Requirement, chatbot harus mampu menjawab pertanyaan terkait informasi kampus dari **knowledge base dokumen resmi**. Berikut adalah **daftar dokumen** yang diperlukan:

No	Kategori Dokumen	Jenis Data	Contoh Konten
1	**Program Studi**	Daftar program studi, akreditasi, kurikulum	Nama prodi (Sistem Informasi, Teknik Informatika), deskripsi, mata kuliah wajib
2	**Panduan Akademik**	Buku panduan mahasiswa, kalender akademik	Peraturan KRS, cuti akademik, yudisium, jadwal UTS/UAS
3	**Prosedur Pendaftaran**	Alur pendaftaran mahasiswa baru (PMB)	Syarat pendaftaran, biaya pendaftaran, jadwal seleksi
4	**Biaya Pendidikan**	Rincian biaya kuliah per semester/tahun	SPP, uang pangkal, biaya praktikum, beasiswa
5	**Fasilitas Kampus**	Informasi laboratorium, perpustakaan, asrama	Jam operasional, layanan yang tersedia
6	**SOP Layanan Akademik**	Prosedur administrasi (legalisir, surat keterangan, dll)	Cara mengajukan cuti, pindah program studi, dll
7	**FAQ (Frequently Asked Questions)**	Pertanyaan umum mahasiswa dan jawabannya	"Bagaimana cara daftar ulang?", "Kapan jadwal wisuda?"
8	**Kontak Unit Layanan**	Informasi kontak BAA, BAAK, Keuangan, dll	Email, nomor telepon, jam pelayanan

Catatan: Data di atas diambil dari **smile.tazkia.ac.id** melalui API, kemudian diproses menjadi **embedding vector** dan disimpan di **Qdrant Vector Database** untuk keperluan **semantic search** saat RAG process.

---

2. Hybrid Approach Database

Berdasarkan arsitektur RAG yang dijelaskan dalam **Arsitektur Chatbot & Mekanisme.pdf**, sistem Sapa Tazkia menggunakan **hybrid approach**:

A. Data Dokumen → Realtime Fetch via API, lalu Cache di Vector Database

Aspek	Penjelasan
Proses Awal (Indexing)	Dokumen dari `smile.tazkia.ac.id` diambil via API → di-convert menjadi **chunks** (potongan teks) → di-generate menjadi **embedding vector** → disimpan di **Qdrant**
Proses Query (Runtime)	Ketika user bertanya, sistem tidak fetch ulang dari API, melainkan **query langsung ke Qdrant** untuk mencari dokumen yang paling relevan (cosine similarity > 0.7)
Update Data	Jika dokumen di `smile.tazkia.ac.id` berubah (misal: kalender akademik update), sistem perlu **re-index** dokumen tersebut ke Qdrant

B. Data Akademik Mahasiswa → Realtime Fetch via API (Tidak Disimpan)

Data	Metode	Alasan
Nilai Mahasiswa	**Realtime API Call** ke `smile.tazkia.ac.id`	Data sensitif, harus selalu up-to-date, tidak perlu duplikasi di database lokal
IPK/IPS	**Realtime API Call**	Data berubah setiap semester, lebih efisien fetch on-demand
Jadwal Kuliah	**Realtime API Call**	Jadwal bisa berubah sewaktu-waktu (jadwal pengganti, libur, dll)
Status Pembayaran	**Realtime API Call**	Data keuangan harus realtime untuk akurasi

Kesimpulan:

- **Dokumen statis (panduan, SOP, FAQ)** → **Disimpan di Qdrant** setelah indexing dari API (untuk performa).

- **Data dinamis mahasiswa (nilai, jadwal)** → **Realtime fetch** dari `smile.tazkia.ac.id` (untuk akurasi dan keamanan).

---

3. List API yang Dibutuhkan

Berikut adalah **daftar API endpoint** yang harus disediakan oleh `smile.tazkia.ac.id` untuk integrasi dengan Sapa Tazkia:

A. API untuk Knowledge Base (Dokumen Kampus)

No	Endpoint API	Method	Deskripsi	Response Example
1	`/api/documents/programs`	GET	Daftar program studi	`[{id, name, code, accreditation, description}]`
2	`/api/documents/academic-guide`	GET	Buku panduan akademik (full text atau per section)	`{title, content, last_updated}`
3	`/api/documents/calendar`	GET	Kalender akademik	`[{event, date, semester, year}]`
4	`/api/documents/tuition-fees`	GET	Informasi biaya kuliah	`{program_id, semester_fee, registration_fee}`
5	`/api/documents/facilities`	GET	Daftar fasilitas kampus	`[{name, description, location, hours}]`
6	`/api/documents/sop`	GET	SOP layanan akademik	`[{title, category, steps, requirements}]`
7	`/api/documents/faq`	GET	FAQ mahasiswa	`[{question, answer, category}]`

8	`/api/documents/contacts`	GET	Kontak unit layanan	`[{unit_name, phone, email, location}]`

Catatan:

- API ini dipanggil **secara periodik** (misal: setiap 24 jam atau saat admin trigger update) untuk **re-indexing** data ke Qdrant.
- Format response sebaiknya **JSON** dengan struktur yang konsisten.

B. API untuk Data Akademik Mahasiswa (Realtime)

No	Endpoint API	Method	Deskripsi	Request Params	Response Example
1	`/api/students/auth`	POST	Autentikasi mahasiswa (login)	`{nim, password}`	`{token, user_id, name, program_studi}`
2	`/api/students/profile`	GET	Data profil mahasiswa	Header: `Authorization: Bearer <token>`	`{nim, name, email, phone, program_studi, angkatan}`
3	`/api/students/grades`	GET	Nilai semester tertentu	`?semester=5`	`[{course_code, course_name, grade, sks, grade_point}]`
4	`/api/students/transcript`	GET	Transkrip lengkap (semua semester)	Header: `Authorization: Bearer <token>`	`{total_sks, ipk, semesters: [{semester, courses: [...]}]}`

5	`/api/students/schedule`	GET	Jadwal kuliah mahasiswa	`?semester=current`	`[{day, time, course_name, room, lecturer}]`
6	`/api/students/academic-status`	GET	Status akademik (aktif/cuti/dll)	Header: `Authorization: Bearer <token>`	`{status, semester_active, total_semesters}`
7	`/api/students/payment-status`	GET	Status pembayaran kuliah	Header: `Authorization: Bearer <token>`	`{semester, paid, amount, due_date}`

Catatan:

- API ini memerlukan **autentikasi JWT** (token) untuk keamanan.
- Data **tidak disimpan** di database Sapa Tazkia, hanya **di-fetch saat diperlukan**.

C. API untuk Admin/Monitoring (Opsional)

No	Endpoint API	Method	Deskripsi	Request Params
1	`/api/admin/trigger-reindex`	POST	Trigger re-indexing dokumen ke Qdrant	`{document_category}`
2	`/api/admin/document-version`	GET	Cek versi/timestamp terakhir update dokumen	`?category=academic-guide`

Alur Kerja Integrasi API dalam RAG Process

Berikut adalah **alur lengkap** bagaimana API `smile.tazkia.ac.id` digunakan dalam sistem Sapa Tazkia:

Fase 1: Indexing (Dilakukan Pertama Kali atau Saat Update)

1. Backend Sapa Tazkia → Panggil API smile.tazkia.ac.id/api/documents/*
2. Download dokumen (JSON response)
3. Split dokumen menjadi chunks (misal: per 500 kata)
4. Generate embedding vector untuk setiap chunk (menggunakan model embedding)
5. Simpan vector + metadata ke Qdrant Vector Database

Fase 2: Query Runtime (Saat User Bertanya)
```
1. User bertanya: "Apa saja program studi di Tazkia?"
2. Backend → Generate embedding untuk query user
3. Backend → Query Qdrant untuk mencari top 5 dokumen relevan (similarity > 0.7)
4. Backend → Compile context dari dokumen yang ditemukan
5. Backend → Kirim context + query ke AI API (Gemini/GPT)
6. AI → Generate jawaban berdasarkan context
7. Backend → Return jawaban ke Frontend
```

Fase 3: Data Mahasiswa (Realtime Fetch)
```
1. Mahasiswa login → Backend panggil smile.tazkia.ac.id/api/students/auth
2. Dapat JWT token → Simpan di session (Redis/memory)
3. Mahasiswa tanya: "Berapa IPK saya?"
4. Backend → Panggil smile.tazkia.ac.id/api/students/transcript dengan token
5. Backend → Parse response dan format jawaban
6. Backend → Return jawaban ke Frontend
```

Rekomendasi Tambahan

1. **Webhook untuk Update Otomatis:**
    - `smile.tazkia.ac.id` bisa mengirim **webhook** ke Sapa Tazkia saat ada dokumen yang diupdate, sehingga sistem langsung re-index tanpa menunggu cron job.
2. **Caching untuk Performa:**

    - Gunakan **Redis** untuk cache response API yang jarang berubah (misal: daftar program studi, FAQ) agar tidak perlu fetch setiap kali.
3. **Rate Limiting API:**
    - Pastikan `smile.tazkia.ac.id` memiliki **rate limiting** untuk mencegah overload jika banyak user query bersamaan.
4. **Error Handling:**
    - Jika API `smile.tazkia.ac.id` down, sistem harus memiliki **fallback mechanism** (misal: jawab "Maaf, sistem sedang maintenance").