BAB 3
PELAKSANAAN KERJA

3.1 Kedudukan dan Koordinasi

3.1.1 Kedudukan

Pelaksanaan kegiatan magang berada di bawah Divisi *Digital Marketing* PT Hexaon Business Mitrasindo. Dalam kegiatan tersebut, posisi yang dijalani adalah sebagai *website developer* dengan fokus utama pada pengembangan sisi *backend* sistem *website* Hexa School. Seluruh aktivitas dan tanggung jawab yang diberikan berada di bawah pengawasan dan arahan Bapak Praditya Afisda selaku *Head of Digital Marketing*.

3.1.2 Koordinasi

Koordinasi selama pelaksanaan magang dilakukan melalui beberapa mekanisme. Pertemuan langsung (*offline*) dilaksanakan di kantor untuk membahas perkembangan proyek, diskusi teknis, serta pembagian tugas dalam tim. Selain itu, komunikasi harian dan koordinasi lanjutan dilakukan melalui aplikasi WhatsApp guna memastikan kelancaran proses pengembangan.

Dalam aspek teknis, kolaborasi pengembangan sistem difasilitasi melalui *platform* GitHub sebagai media pengelolaan kode sumber. Penggunaan GitHub memungkinkan integrasi kontribusi antaranggota tim, pengendalian versi, serta pemantauan perkembangan proyek secara terstruktur dan terdokumentasi.

3.2 Tugas yang Dilakukan

Selama pelaksanaan kegiatan magang, proses pembuatan sisi *backend* dari proyek *website* dilakukan menggunakan *framework* Express.js. Selama pelaksanaan kegiatan magang, berbagai aktivitas dilakukan dalam rangka mendukung pengembangan proyek yang telah ditugaskan. Aktivitas-aktivitas tersebut merepresentasikan kontribusi yang dikerjakan sepanjang periode magang berlangsung. Berikut adalah rincian tugas dan *progress* yang dilakukan di PT. Hexaon Business Mitrasindo diuraikan pada Tabel 3.1.

Tabel 3.1. Tabel Pekerjaan tiap minggu selama magang

Minggu Ke -	**Pekerjaan yang dilakukan**
1	Pengenalan proyek oleh senior tim tentang apa saja fitur yang dibutuhkan, pembuatan skema database, dan pengenalan terhadap alat *palm scanner*.
2	Memulai kolaborasi dengan membuat *repository* github dan setup proyek *backend* dengan express.js.
3	membuat *migration* dan *models* sesuai, dengan skema database yang sudah dibuat serta, membuat *controller*, dan *endpoint* API untuk masing-masing untuk masing-masing entitas.
4	memulai percobaan menggunakan alat *palm scanner* dengan melakukan *scanning* telapak tangan dalam mode *record* dengan menggunakan API yang disediakan di file dokumentasi alatnya dengan menggunakan *tool* postman.
5	masih mempelajari kembali dan mencoba kembali dalam mengoperasikan mode *record* untuk memulai *scanning* telapak tangan dengan API yang disediakan.
6	melakukan integrasi dengan database untuk menyimpan setiap hasil *record* telapak tangan dan diskusi kembali dengan tim tentang penambahan fitur yang dibuat serta, ada penambahan dan perubahan pada skema database awal.
7	menambahkan *migration, models, controller*, dan API untuk setiap entitas tambahan serta menyesuaikan kembali setiap *primary key, foreign key*, dan relasi antar entitas pada *migration* dan *models*.
8	memulai proses menjalankan mode *background* pada alat *palm scanner* untuk melakukan *scanning* telapak tangan, guna mengenali data telapak tangan yang sudah tersimpan di dalam *database*.
	Lanjut pada halaman berikutnya

Tabel 3.1: Tabel Pekerjaan tiap minggu selama magang (lanjutan)

Minggu Ke -	Pekerjaan yang dilakukan
9	masih mempelajari dan mencoba dalam mode *background* yang masih belum bisa membaca dari *database*.
10	melanjutkan mode *background* untuk disinkronkan dengan *database*
11	memperbaiki *controller* dan *models* yang kurang sesuai
12	memperbaiki masalah pada mesin *palm scanner* yang tidak mau respons pada *background mode*
13	menambahkan beberapa *endpoint* tambahan pada *controller* dan *routes* untuk kebutuhan sisi frontend

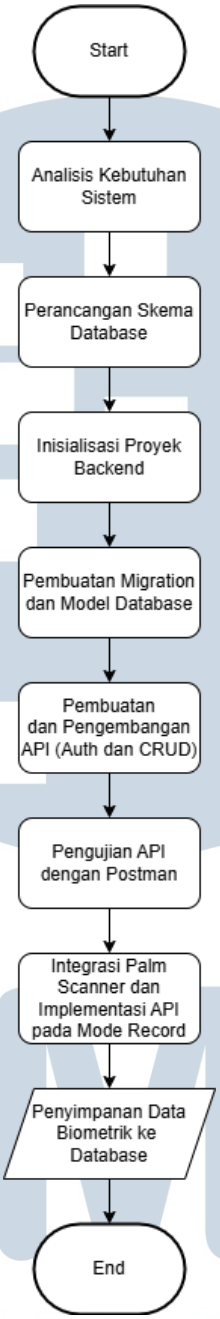
3.3 Uraian Pelaksanaan Magang

Selama pelaksanaan kegiatan magang setiap proses kerja berlangsung secara bertahap, dimulai dari perancangan awal dan inisialisasi proyek, pengembangan struktur database dan API dasar, hingga integrasi perangkat palm scanner. Tahap-tahap selanjutnya mencakup penyesuaian struktur sistem, implementasi mode background, optimalisasi logika backend, serta pengembangan endpoint tambahan untuk mendukung kebutuhan frontend. Seluruh tahapan tersebut dipaparkan secara sistematis pada subbagian berikut.

3.3.1 *Flowchart* Alur Kerja Pengembangan Sistem Backend

A *Flowchart* Pengembangan Sistem Backend dan Perekaman Data Biometrik

Flowchart pada Gambar 3.1 menggambarkan alur kerja pengembangan sistem *backend* Hexa School selama pelaksanaan kegiatan magang. Alur ini menunjukkan tahapan pengembangan sistem secara bertahap, dimulai dari analisis kebutuhan hingga sistem hingga integrasi dengan mode *Record* alat *palm scanner*.



Gambar 3.1. Flowchart Pengembangan Sistem Backend Hexa School dan Perekaman Data Biometrik

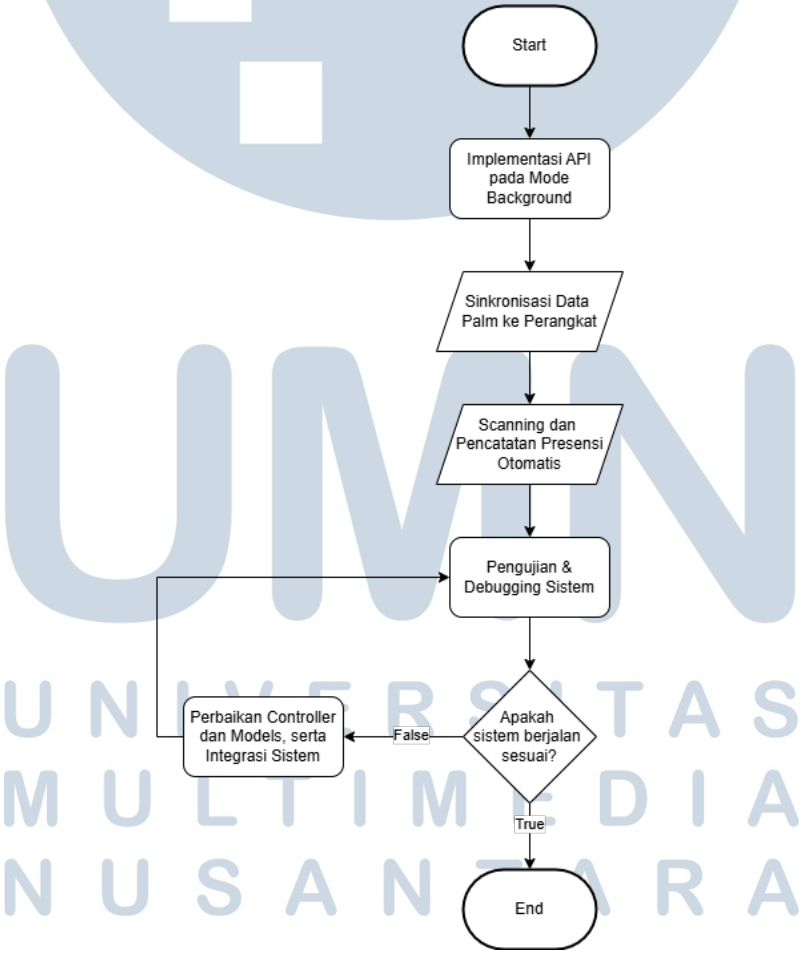
Berdasarkan flowchart tersebut, proses pengembangan diawali dengan tahap analisis kebutuhan sistem dan perancangan skema basis data. Selanjutnya dilakukan inisialisasi proyek *backend* menggunakan Express.js, diikuti dengan pembuatan *migration* dan *model* basis data. Tahap berikutnya mencakup pengembangan API dasar yang meliputi autentikasi dan operasi *CRUD*, serta pengujian API

menggunakan Postman.

Setelah API dasar berjalan dengan baik, dilakukan integrasi perangkat *palm scanner* melalui implementasi API pada mode *record*. Data biometrik hasil pemindaian kemudian disimpan ke dalam basis data sebagai sumber data utama untuk proses identifikasi pada tahap selanjutnya.

B Flowchart Integrasi Perangkat Mode *Background* dan Presensi Otomatis

Setelah sistem *backend* siap digunakan, tahap selanjutnya adalah integrasi perangkat *palm scanner* dan implementasi mekanisme pencatatan presensi otomatis. Flowchart pada Gambar 3.2 menunjukkan alur operasional sistem pada mode *background*, termasuk proses sinkronisasi data, pencatatan presensi, serta pengujian sistem secara iteratif.



Gambar 3.2. Flowchart Integrasi Perangkat dengan Mode *Background* dan Presensi Otomatis

Berdasarkan flowchart tersebut, proses dimulai dengan implementasi API pada mode *background* yang memungkinkan perangkat *palm scanner* melakukan sinkronisasi data biometrik dari sistem *backend*. Data yang telah tersinkronisasi kemudian digunakan oleh perangkat untuk melakukan pemindaian telapak tangan dan pencatatan presensi secara otomatis.

Pada tahap akhir, dilakukan proses pengujian dan debugging sistem untuk memastikan seluruh fungsi berjalan sesuai dengan kebutuhan. Apabila ditemukan kesalahan atau ketidaksesuaian, maka dilakukan perbaikan pada *controller*, *model*, serta integrasi sistem, kemudian dilakukan pengujian ulang. Proses ini dilakukan secara berulang hingga sistem dinyatakan berjalan dengan baik dan siap digunakan.

3.3.2 Tahap Perancangan Sistem dan Inisialisasi Pengembangan

Tahap awal pelaksanaan difokuskan pada proses pemahaman proyek secara menyeluruh, meliputi penjelasan alur sistem, fitur-fitur yang akan dibangun, serta mekanisme integrasi perangkat *palm scanner* sebagai komponen utama sistem. Pada tahap ini juga dilakukan penyusunan rancangan awal skema *database* sebagai dasar struktur penyimpanan data.

Setelah memahami kebutuhan sistem, dilakukan proses inisialisasi pengembangan melalui pembuatan *repository* GitHub sebagai media kolaborasi tim. Struktur dasar proyek *backend* menggunakan Express.js kemudian diinisialisasi, mencakup konfigurasi lingkungan pengembangan serta pengujian awal koneksi sistem untuk memastikan seluruh komponen siap dikembangkan lebih lanjut.

3.3.3 Pembuatan Struktur Database

Tahap ini berfokus pada pembentukan fondasi utama sistem melalui pembuatan *migration* dan *models* berdasarkan skema *database* yang telah dirancang. Struktur data ini menjadi dasar bagi sistem dalam melakukan proses penyimpanan dan pengelolaan informasi.

Basis data yang digunakan pada sistem Hexa School memanfaatkan layanan Supabase sebagai *managed database* berbasis PostgreSQL. Pemilihan Supabase didasarkan pada kemudahan integrasi dengan aplikasi berbasis Node.js, dukungan terhadap skalabilitas, serta kemampuannya dalam menyediakan layanan basis data yang stabil dan terkelola. PostgreSQL digunakan sebagai *Database Management System* (DBMS) utama karena mendukung pengelolaan data relasional yang

kompleks. Untuk memberikan gambaran mengenai struktur basis data yang digunakan, ditampilkan cuplikan skema *database* pada Gambar 3.3.

Gambar 3.3. Skema Database Sistem Hexa School

3.3.4 Pengembangan API Dasar

Setelah struktur basis data selesai dibangun, tahap berikutnya adalah pengembangan *endpoint* API untuk memastikan sistem *backend* dapat berinteraksi dengan basis data secara efektif. Salah satu komponen utama pada fase ini adalah pembuatan API autentikasi yang meliputi proses registrasi dan *login*.

Berikut adalah contoh struktur rute sederhana yang digunakan untuk menghubungkan permintaan klien ke *controller* autentikasi:

```
router.post("/register", AuthController.register);
router.post("/login", AuthController.login);
```

Kode 3.1: Cuplikan Routes Autentikasi

Pada proses registrasi, dilakukan beberapa langkah penting seperti validasi input, pengecekan keberadaan email, proses hashing kata sandi, serta pembuatan data pengguna. Cuplikan berikut menunjukkan bagian inti logika registrasi:

```
const existingUser = await users.findOne({ where: { email } });
if (existingUser) {
  return res.status(400).json({ message: "Email already registered" })
    ;
```

```
}

const hashedPassword = await bcrypt.hash(password, 10);

const newUser = await users.create({
  username,
  email,
  password: hashedPassword,
  role_id,
});
```

Kode 3.2: Proses Validasi dan Hashing Password

Flow autentikasi selanjutnya berlanjut pada proses *login*. Sistem melakukan pencarian pengguna berdasarkan identitas yang diberikan (*email*, *nip*, atau *nisn*), memverifikasi kecocokan kata sandi, serta menghasilkan *token* JWT seperti terlihat pada cuplikan berikut:

```
const isMatch = await bcrypt.compare(password, user.password);
if (!isMatch) {
  return res.status(401).json({ message: "Invalid identifier or
    password" });
}

const token = jwt.sign(
  { id: user.id, email: user.email, role_id: user.role_id },
  SECRET_KEY,
  { expiresIn: "1h" }
);

return res.json({
  message: "Login success",
  token,
  user,
});
```

Kode 3.3: Cuplikan Proses Login dan JWT

Dengan diselesaikannya tahap ini, sistem telah memiliki API dasar yang dapat melayani autentikasi dan pengelolaan data awal sebagai bagian dari fondasi pengembangan fitur-fitur lanjutan.

3.3.5 Pengujian Perangkat dan Pendalaman Mode Record

Tahap selanjutnya merupakan fase integrasi awal perangkat *palm scanner*. Pengujian dilakukan menggunakan mode *record*, yaitu mode yang digunakan untuk melakukan perekaman data telapak tangan. Pada tahap awal, API yang disediakan oleh perangkat diuji menggunakan Postman untuk memastikan format komunikasi, parameter permintaan, serta struktur *response* yang dihasilkan perangkat sesuai dengan dokumentasi.

Salah satu endpoint utama yang diuji adalah `/palm/start`, yaitu perintah untuk mengaktifkan mode perekaman. Endpoint ini menggunakan metode `POST` dan tidak memerlukan data tambahan. Contoh *response* dari perangkat ditunjukkan berikut:

```
\textit{Response} berhasil:
{
    "code": 200,
    "message": "OK"
}
```

Pengujian kemudian diperdalam dengan melakukan variasi pemanggilan API guna memahami lebih rinci parameter, pola *response*, serta alur pengiriman data dari perangkat. Pada tahap ini, integrasi dilakukan melalui kode pada sisi *backend* untuk memastikan perangkat dapat berkomunikasi dengan sistem secara konsisten. Contoh implementasi pemanggilan endpoint `/palm/start` ditunjukkan pada cuplikan kode berikut:

```
const startPalmPrint = async (req, res) => {
  try {
    const result = await requestDevice("/palm/start", "POST");
    res.status(200).json({ message: "Palm print recognition started",
    result });
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
};
```

Kode 3.4: Cuplikan proses palm print

Berdasarkan hasil pengujian, perangkat memberikan alur operasi yang konsisten, yaitu:

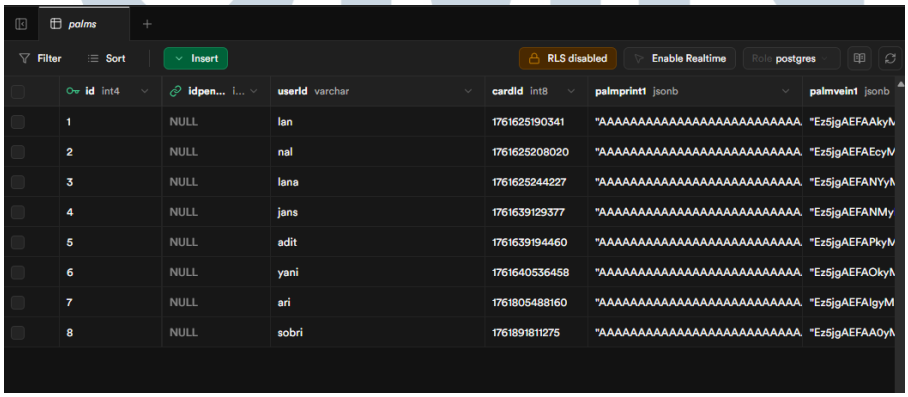
1. Menghapus data perekaman sebelumnya melalui endpoint `/palm/clear`.
2. Mengaktifkan mode perekaman menggunakan `/palm/start`.
3. Memantau status proses melalui `/palm/getState`.
4. Mengambil fitur biometrik menggunakan `/palm/getEnrollFeat`.
5. Mengambil citra telapak tangan menggunakan `/palm/getEnrollImg`.

Seluruh proses ini juga telah diintegrasikan dengan antarmuka aplikasi sehingga pemindaian dapat dilakukan langsung melalui UI tanpa harus menggunakan Postman. Tahap ini menjadi krusial karena hasil pemindaian telapak tangan akan menjadi input utama bagi sistem identifikasi yang sedang dikembangkan.

3.3.6 Integrasi Data Rekaman dan Penyesuaian Struktur Sistem

Setelah memahami alur kerja mode *record*, langkah selanjutnya adalah melakukan integrasi hasil pemindaian telapak tangan dengan basis data sistem. Setiap data hasil perekaman—baik citra telapak tangan maupun parameter fitur biometrik—disimpan ke dalam tabel `palms` sebagai sumber informasi utama untuk proses verifikasi dan identifikasi pada tahap berikutnya.

Untuk memastikan data dapat dikelola secara terpusat dan diakses oleh seluruh anggota tim pengembang, basis data ditempatkan pada layanan Supabase. Gambar 3.4 menunjukkan tampilan tabel `palms` yang berisi beberapa entri hasil perekaman dari perangkat *palm scanner*.



Gambar 3.4. Tabel `palms` pada Supabase yang menyimpan data hasil perekaman telapak tangan

Pada tahap ini dilakukan pula evaluasi ulang terhadap kebutuhan data oleh tim pengembang. Beberapa penyesuaian diperlukan, seperti penambahan atribut baru, restrukturisasi relasi antartabel, serta penetapan aturan integritas data. Perubahan tersebut diimplementasikan melalui pembaruan berkas *migration*, penyempurnaan *model* ORM (sequelize), dan revisi hubungan antartabel agar sesuai dengan kebutuhan pemrosesan biometrik.

Selain itu, pengembangan endpoint API tambahan juga dilakukan untuk memastikan hasil perekaman dapat diproses, disimpan, dan diakses dengan benar oleh modul lain dalam sistem. Dengan demikian, integrasi antara perangkat, layanan backend, dan basis data dapat berjalan secara konsisten dan mendukung kelanjutan proses verifikasi maupun identifikasi pengguna.

3.3.7 Implementasi Mode *Background*

Mode *background* merupakan mekanisme sinkronisasi data antara sistem *backend* dan perangkat *palm scanner*, di mana seluruh data biometrik pengguna dikelola secara terpusat pada sistem. Pada mode ini, perangkat tidak melakukan perekaman data biometrik secara mandiri, melainkan menerima data yang telah disimpan sebelumnya dari *database* melalui sistem *backend*.

Data biometrik tersebut dikirimkan melalui *endpoint getWorkflowUsers* dan disimpan secara lokal pada perangkat untuk keperluan proses identifikasi. Ketika pengguna melakukan pemindaian telapak tangan, perangkat akan melakukan pencocokan secara langsung terhadap data lokal tersebut. Hasil proses identifikasi kemudian dikirimkan kembali ke sistem *backend* melalui *endpoint uploadAccessInfo* sebagai dasar pencatatan presensi secara otomatis.

A Sinkronisasi Data Pengguna pada Mode *Background*

Pada mode *background*, perangkat *palm scanner* secara berkala mengirimkan permintaan ke sistem *backend* untuk memperoleh data pengguna yang perlu diproses. Proses ini dilakukan melalui endpoint pengambilan data *workflow* (*getWorkflowUsers*), di mana data diklasifikasikan berdasarkan status tertentu. Status `101` menunjukkan data pengguna yang perlu ditambahkan ke perangkat, sedangkan status `102` menunjukkan data pengguna yang perlu dihapus dari perangkat.

Permintaan data *workflow* dikirimkan oleh perangkat dalam format

JSON dengan parameter sebagaimana ditunjukkan pada Tabel 3.2. Sistem *backend* kemudian mengambil data yang sesuai dari basis data Supabase dan mengirimkannya kembali ke perangkat dengan struktur *paging* sesuai spesifikasi protokol MP30.

Tabel 3.2. Parameter Permintaan Data *Workflow* MP30

Parameter	**Tipe**	**Wajib**	**Keterangan**
deviceSn	string	Ya	Nomor seri perangkat
status	string	Ya	101: tambah data, 102: hapus data
pageIndex	int	Ya	Nomor halaman data
pageSize	int	Ya	Jumlah data per halaman

Apabila status bernilai `101`, sistem *backend* akan mengirimkan data biometrik telapak tangan yang mencakup fitur *palmprint* dan *palm vein*. Contoh *response* sukses untuk status `101` ditunjukkan sebagai berikut:

```
{
  "code": 0,
  "dataToal": 100,
  "data": [
    {
      "userId": "Tluson",
      "cardId": 192030349,
      "updateTime": "2023-09-12 15:20:57",
      "palmprint1": "XXXXXXXX",
      "palmvein1": "XXXXXXXX",
      "palmprint2": "XXXXXXXX",
      "palmvein2": "XXXXXXXX"
    }
  ],
  "pageTotal": 10,
  "pageIndex": 1,
  "pageSize": 10,
  "message": "OK"
}
```

Sebaliknya, apabila status bernilai `102`, sistem hanya mengirimkan daftar

userId yang perlu dihapus dari perangkat. Setelah perangkat berhasil memproses data tersebut, perangkat akan mengirimkan laporan balik melalui endpoint konfirmasi *workflow* untuk memperbarui status data di dalam basis data Supabase agar tidak dikirim ulang pada proses sinkronisasi berikutnya.

B Pencatatan Akses dan Presensi Otomatis

Selain sinkronisasi data pengguna, mode *background* juga mendukung pencatatan akses secara otomatis. Setiap kali terjadi proses autentikasi telapak tangan, perangkat akan mengirimkan informasi akses ke sistem *backend* melalui endpoint *uploadAccessInfo*. Informasi ini digunakan sebagai dasar pencatatan presensi secara otomatis.

Parameter yang dikirimkan oleh perangkat pada proses pencatatan akses ditunjukkan pada Tabel 3.3.

Tabel 3.3. Parameter Unggah Informasi Akses MP30

Parameter	Tipe	Wajib	Keterangan
deviceSn	string	Ya	Nomor seri perangkat
authenType	string	Ya	Metode autentikasi (telapak tangan, kartu, QR)
userId	string	Tidak	ID pengguna
cardId	string	Tidak	Nomor kartu
qrdata	string	Tidak	Data QR Code
passState	int	Ya	Status akses (berhasil/gagal)
authenTime	string	Ya	Waktu autentikasi

Contoh *response* sukses dari sistem *backend* terhadap permintaan pencatatan akses adalah sebagai berikut:

```
{
  "code": 0,
  "message": "OK"
}
```

Sistem *backend* dirancang untuk memberikan respons secepat mungkin terhadap permintaan ini sesuai dengan ketentuan protokol perangkat. Pendekatan ini bertujuan untuk mencegah perangkat mengalami kondisi *stuck* atau melakukan pengiriman data ulang secara berulang. Proses pencatatan presensi dilakukan

menggunakan prinsip *non-blocking processing*, sehingga tidak menghambat kinerja utama perangkat biometrik.

3.3.8 Optimalisasi Logika Sistem dan Penyempurnaan API

Setelah mekanisme kerja perangkat dan sistem *backend* berjalan dengan stabil, tahap selanjutnya difokuskan pada penyempurnaan logika sistem dan antarmuka pemrograman aplikasi (API). Beberapa bagian pada *controller* dan *models* dilakukan penyesuaian untuk memastikan alur pertukaran data berjalan sesuai dengan kebutuhan sistem, khususnya dalam mendukung respons yang cepat dan konsisten pada sisi *frontend*.

Selain optimalisasi logika pemrosesan data, tahap ini juga mencakup penyelesaian permasalahan teknis yang berkaitan dengan implementasi mode *background* pada perangkat *palm scanner*. Kendala yang ditemukan tidak berasal dari kesalahan logika aplikasi, melainkan dari ketidaksesuaian konfigurasi port layanan yang digunakan oleh sistem *backend* dengan port yang diakses oleh perangkat. Kondisi tersebut menyebabkan perangkat tidak dapat mengirimkan permintaan API secara optimal sehingga tidak memperoleh respons yang diharapkan.

Untuk mengatasi permasalahan tersebut, dilakukan penyesuaian konfigurasi port dan pengujian ulang terhadap endpoint yang digunakan pada mode *background*, seperti *getWorkflowUsers* dan *uploadAccessInfo*. Setelah konfigurasi disesuaikan dan layanan *backend* melakukan *listening* pada port yang tepat, komunikasi antara perangkat dan sistem dapat berjalan dengan normal. Dengan demikian, proses sinkronisasi data pengguna serta pencatatan akses dan presensi otomatis dapat kembali berfungsi secara stabil.

3.3.9 Pengembangan Endpoint Tambahan untuk Kebutuhan Frontend

Tahap ini berfokus pada penyusunan dan penyempurnaan endpoint API yang dibutuhkan oleh sisi frontend dalam proses pengembangan aplikasi. Beberapa endpoint baru ditambahkan untuk mendukung kebutuhan antarmuka pengguna, termasuk akses data akademik, pengelolaan entitas, serta pemrosesan informasi hasil pemindaian telapak tangan dari perangkat *palm scanner*.

Seluruh endpoint yang dikembangkan telah disesuaikan dengan struktur basis data final dan dirancang agar dapat diakses secara konsisten oleh modul frontend

berbasis React. Dengan terselesaikannya tahap ini, sistem backend telah memiliki fungsionalitas yang lengkap dan siap digunakan pada tahap integrasi lanjutan.

Daftar lengkap endpoint API yang tersedia ditampilkan pada Tabel 3.4. Tabel ini mencakup seluruh rute dan fungsi API yang digunakan dalam sistem Hexa School, meliputi autentikasi, manajemen data akademik, pengelolaan pengguna, presensi, hingga entitas biometrik.

Tabel 3.4. Daftar Endpoint API pada Sistem Hexa School

No	Endpoint	Method	Deskripsi
1	`/api/auth/login`	POST	Melakukan autentikasi pengguna dan menghasilkan token akses atau JWT.
2	`/api/auth/register`	POST	Melakukan registrasi untuk *user* baru baik untuk *role teacher, employee*, dan *student*.
3	`/api/student/get`	GET	Mengambil daftar seluruh siswa dari *database*.
4	`/api/student/get/:id`	GET	Mengambil informasi siswa dari *database* per ID siswa.
5	`/api/student/:id/scheduled`	GET	Mengambil daftar jadwal pelajaran siswa per ID siswa.
6	`/api/student/update/:id`	PUT	Melakukan perubahan atau meng-*update* informasi siswa.
7	`/api/student/delete/:id`	DELETE	Menghapus data siswa sesuai dengan ID siswa.
8	`/api/classes/post`	POST	Membuat kelas baru.
9	`/api/classes/get`	GET	Menampilkan semua kelas yang sudah dibuat.
10	`/api/classes/class:id`	GET	Menampilkan daftar siswa dalam sebuah kelas.
11	`/api/classes/delete/:id`	DELETE	Menghapus kelas sesuai dengan ID.

Bersambung ke halaman berikutnya

Lanjutan Tabel 3.4 dari halaman sebelumnya

No	Endpoint	Method	Deskripsi
12	`/api/classes/put/:id`	PUT	Mengubah atau meng-*update* informasi kelas yang sudah dibuat.
13	`/api/role/post`	POST	Membuat *role* baru dalam sistem, seperti *admin*, *teacher*, atau *student*.
14	`/api/role/get`	GET	Mengambil daftar seluruh *role* yang tersedia di dalam sistem.
15	`/api/role/delete/:id`	DELETE	Menghapus data *role* berdasarkan ID tertentu.
16	`/api/academic/post`	POST	Menambahkan data tahun ajaran baru ke dalam sistem.
17	`/api/academic/get`	GET	Mengambil daftar seluruh tahun ajaran yang tersimpan di basis data.
18	`/api/academic/delete/:id`	DELETE	Menghapus data tahun ajaran berdasarkan ID tertentu.
19	`/api/attendance/get`	GET	Mengambil seluruh catatan presensi siswa dan karyawan.
20	`/api/attendance/post`	POST	Menambahkan catatan presensi baru, misalnya hasil pemindaian dari *palm scanner*.
21	`/api/attendance/put/:id`	PUT	Memperbarui data presensi berdasarkan ID.
22	`/api/attendance/delete/:id`	DELETE	Menghapus catatan presensi berdasarkan ID tertentu.
23	`/api/employee/get`	GET	Mengambil daftar seluruh karyawan dari basis data.
24	`/api/employee/get/:id`	GET	Mengambil informasi detail karyawan berdasarkan ID.

Bersambung ke halaman berikutnya

Lanjutan Tabel 3.4 dari halaman sebelumnya

No	Endpoint	Method	Deskripsi
25	`/api/employee/shift`	POST	Menetapkan atau membuat jadwal kerja untuk karyawan tertentu.
26	`/api/employee/:id`	DELETE	Menghapus data karyawan berdasarkan ID.
27	`/api/schedule/post`	POST	Menambahkan jadwal pelajaran baru ke dalam sistem.
28	`/api/schedule/get`	GET	Mengambil daftar seluruh jadwal yang tersedia.
29	`/api/schedule/delete/:id`	DELETE	Menghapus jadwal berdasarkan ID tertentu.
30	`/api/schedule/update/:id`	PATCH	Memperbarui jadwal pelajaran berdasarkan ID.
31	`/api/subject/post`	POST	Menambahkan mata pelajaran baru.
32	`/api/subject/get`	GET	Mengambil daftar seluruh mata pelajaran yang tersedia.
33	`/api/subject/delete/:id`	DELETE	Menghapus data mata pelajaran berdasarkan ID.
34	`/api/subject/update/:id`	PUT	Memperbarui informasi mata pelajaran berdasarkan ID.
35	`/api/teacher/get`	GET	Mengambil daftar seluruh guru dari basis data.
36	`/api/teacher/get/:id`	GET	Mengambil informasi detail guru berdasarkan ID.
37	`/api/teacher/update/:id`	PUT	Memperbarui data guru berdasarkan ID.
38	`/api/teacher/delete/:id`	DELETE	Menghapus data guru berdasarkan ID.

3.4 Kendala yang Ditemukan

Selama pelaksanaan kegiatan magang dan pengembangan sistem *backend* pada proyek *Hexa School*, beberapa kendala ditemui, terutama pada tahap integrasi antara sistem dengan perangkat *palm scanner*. Proses integrasi ini menuntut pemahaman yang mendalam terhadap *Software Development Kit* (SDK) yang disediakan oleh perangkat, termasuk mekanisme kerja serta struktur *endpoint* API yang digunakan.

Kendala utama muncul pada saat menghubungkan hasil pemindaian telapak tangan dengan basis data melalui logika pemrosesan di sisi *backend*. Integrasi tidak dapat dilakukan secara langsung karena setiap fungsi dalam SDK memiliki aturan serta format komunikasi yang spesifik serta, dibutuhkan pencocokan *port* yang sesuai untuk komunikasi antara sistem *backend* dengan *palm scanner* dapat berjalan.

3.5 Solusi atas Kendala yang Ditemukan

Untuk mengatasi kendala-kendala tersebut, dilakukan sejumlah langkah perbaikan dan penyesuaian. Upaya yang dilakukan meliputi:

1. Mempelajari secara mendalam dokumentasi resmi serta contoh implementasi SDK yang disediakan oleh perangkat *palm scanner*.
2. Melakukan pengujian komunikasi antara sistem *backend* dan perangkat menggunakan aplikasi *Postman* guna memastikan kesesuaian format, *port* komunikasi *request* dan *response* API.