RANCANG BANGUN SISTEM INFORMASI PENGADUAN PENGGUNA BERBASIS WEB

Dosen pengampu: Pahrul Irfan, S.Kom., M.Kom.

Disusun Oleh:

Nurul Fatimah	**F1D022085**
Putra Heryan Gagah Perkasa	**F1D022087**
Baiq Alfia Zahira	**F1D02310042**

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS MATARAM

2025

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pelayanan publik yang berkualitas memerlukan saluran komunikasi yang efektif antara pengguna dan pihak pengelola. Salah satu bentuk komunikasi tersebut adalah penyampaian pengaduan atas permasalahan yang dialami pengguna, seperti keluhan di bidang akademik, kerusakan fasilitas, maupun aspek keamanan. Namun, dalam praktiknya, proses pengaduan masih sering dilakukan secara lisan atau melalui media sosial yang belum terkelola secara sistematis, sehingga data pengaduan tidak terdokumentasi dengan baik dan berpotensi terabaikan.

Kondisi tersebut menimbulkan berbagai permasalahan, di antaranya kesulitan pengguna dalam mengetahui perkembangan atau status laporan yang telah disampaikan. Di sisi lain, pihak admin juga menghadapi kendala dalam melakukan pencatatan, pengelompokan, serta rekapitulasi data pengaduan secara terpusat. Proses pengelolaan yang masih bersifat manual berpotensi menimbulkan ketidakakuratan data, keterlambatan dalam penanganan laporan, serta menurunkan tingkat transparansi dan kepercayaan pengguna terhadap layanan yang diberikan.

Seiring dengan perkembangan teknologi informasi, pemanfaatan sistem informasi berbasis web menjadi solusi yang relevan untuk mengatasi permasalahan tersebut. Sistem Informasi Pengaduan Pengguna Berbasis Web memungkinkan pengguna untuk menyampaikan laporan secara terstruktur berdasarkan kategori tertentu, mengakses riwayat pengaduan, serta memantau status laporan secara *real-time*. Selain itu, sistem ini membantu admin dalam mengelola, memproses, dan memperbarui status pengaduan secara lebih efektif dan terorganisir. Dengan diterapkannya sistem ini, diharapkan proses pengaduan dapat berjalan secara lebih transparan, akuntabel, dan efisien, serta mampu meningkatkan kualitas pelayanan secara keseluruhan.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka rumusan masalah dalam penelitian atau pengembangan sistem ini adalah sebagai berikut:

1. Bagaimana merancang dan membangun Sistem Informasi Pengaduan Pengguna berbasis web yang dapat memfasilitasi penyampaian pengaduan secara terstruktur dan terdokumentasi dengan baik?

2. Bagaimana sistem dapat membantu pengguna dalam melihat riwayat pengaduan serta memantau status laporan secara *real-time*?
3. Bagaimana sistem dapat mendukung admin dalam mengelola, memproses, dan memperbarui status pengaduan secara terpusat dan efisien?
4. Bagaimana penerapan sistem informasi pengaduan berbasis web dapat meningkatkan transparansi dan efektivitas proses pengelolaan pengaduan?

1.3 Batasan Masalah

Agar pembahasan dalam pengembangan sistem ini lebih terarah dan tidak meluas, maka ditetapkan beberapa batasan masalah sebagai berikut:

1. Sistem yang dikembangkan berbasis web dan hanya dapat diakses melalui peramban (*browser*).
2. Pengguna sistem dibatasi pada dua peran, yaitu pengguna (*user*) dan admin.
3. Jenis pengaduan dibatasi pada tiga kategori saja, seperti pengaduan akademik, kerusakan fasilitas, dan keamanan.
4. Fitur yang tersedia bagi pengguna meliputi *login*, pengajuan pengaduan, melihat riwayat pengaduan, memantau status laporan, serta melihat galeri perbaikan fasilitas.
5. Fitur yang tersedia bagi admin meliputi *login*, melihat daftar masuk pengaduan, pengelolaan data pengaduan dan memperbarui status laporan.

BAB II
ANALISIS DAN PERANCANGAN SISTEM

2.1 Analilsis Kebutuhan Sistem

Analisis kebutuhan sistem dilakukan untuk mengidentifikasi fungsi dan spesifikasi yang harus dimiliki oleh Sistem Informasi Pengaduan Pengguna Berbasis Web agar dapat berjalan sesuai dengan tujuan pengembangannya. Tahap ini bertujuan untuk memastikan bahwa sistem yang dibangun mampu memenuhi kebutuhan pengguna dan admin secara efektif, efisien, serta mudah digunakan.

2.1.1 Kebutuhan Fungsional

Kebutuhan fungsional merupakan kebutuhan yang berkaitan langsung dengan fungsi atau layanan yang harus disediakan oleh sistem. Adapun kebutuhan fungsional dalam sistem ini adalah sebagai berikut:

1. Sistem mampu menyediakan fitur autentikasi berupa *login* untuk pengguna dan admin.
2. Sistem memungkinkan pengguna untuk menambahkan pengaduan baru dengan mengisi kategori pengaduan, judul laporan, deskripsi detail, serta tanggal kejadian.
3. Sistem mampu menyimpan dan menampilkan riwayat pengaduan yang telah diajukan oleh pengguna.
4. Sistem menampilkan status pengaduan kepada pengguna, seperti *pending* dan selesai.
5. Sistem memungkinkan admin untuk melihat seluruh data pengaduan yang masuk.
6. Sistem memungkinkan admin untuk memproses pengaduan dan memperbarui status laporan.
7. Sistem menyimpan seluruh data pengaduan secara terpusat dalam basis data.

2.1.2 Kebutuhan *Non*-Fungsional

Kebutuhan non-fungsional berkaitan dengan kualitas sistem dalam mendukung kinerja dan kenyamanan pengguna. Kebutuhan *non*-fungsional pada sistem ini meliputi:

1. Kemudahan Penggunaan (*Usability*)

   Sistem dirancang dengan antarmuka yang sederhana dan mudah dipahami agar dapat digunakan oleh berbagai kalangan pengguna.

2. Keamanan (*Security*)

   Sistem menerapkan proses login untuk membatasi akses berdasarkan peran pengguna dan admin.

3. Kinerja (*Performance*)

   Sistem mampu menampilkan data pengaduan dan status laporan dengan waktu respon yang cepat.

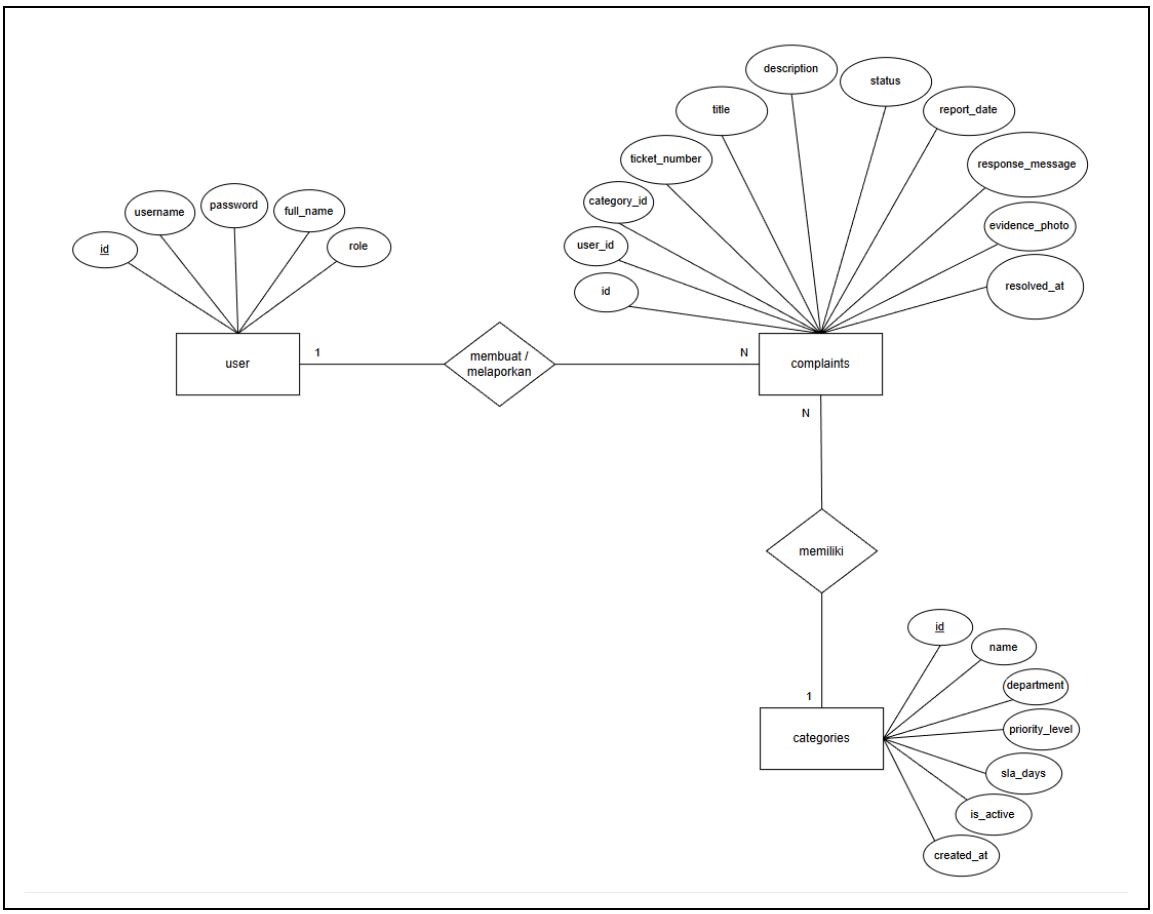
4. Keandalan (*Reliability*)

   Sistem dapat digunakan secara stabil selama proses pengajuan dan pengelolaan pengaduan.

5. Aksesibilitas

   Sistem berbasis web sehingga dapat diakses melalui perangkat yang memiliki peramban (*browser*) dan koneksi internet.

2.2 ERD (*Entity Relationship Diagram*)



Gambar 2.1 *Entity Relationship diagram*

Gambar di atas merupakan ERD dari Sistem Informasi Pengaduan Pengguna berbasis web yang terdiri dari tiga entitas utama, yaitu `Users`, `Complaints`, dan `Categories`.

Entitas `Users` menyimpan data pengguna sistem, baik mahasiswa maupun admin, dengan atribut utama seperti `id`, `username`, `password`, `full_name`, dan `role`.

Entitas `Complaints` merupakan inti dari sistem karena menyimpan seluruh data pengaduan yang dilaporkan oleh pengguna. Atribut yang dimiliki mencakup `id`, `user_id`, `category_id`, `ticket_number`, `title`, `description`, `status`, `report_date`, `response_message`, `evidence_photo`, `resolved_at`.

Entitas `Categories` berfungsi untuk mengelompokkan pengaduan berdasarkan jenis permasalahan. Atribut seperti `name`, `department`, `priority_level`, dan `sla_days` digunakan untuk mendukung proses penanganan yang lebih terstruktur, terukur, dan sesuai dengan tingkat prioritas serta batas waktu penyelesaian.

Secara keseluruhan, ERD ini mendukung pengelolaan pengaduan secara terstruktur dan terintegrasi dalam sistem pengaduan kampus.

2.3 Perancangan Alur Kerja (*Business Logic Flow*)

Perancangan alur kerja sistem dilakukan untuk menggambarkan proses bisnis yang terjadi dalam Sistem Informasi Pengaduan Pengguna Berbasis Web. Alur kerja dimulai ketika pengguna melakukan *login* ke dalam sistem. Setelah berhasil masuk, pengguna dapat mengajukan pengaduan dengan mengisi data yang telah disediakan. Data pengaduan tersebut kemudian disimpan ke dalam basis data dan ditampilkan pada halaman riwayat pengaduan dengan status awal *pending*.

Selanjutnya, admin mengakses sistem melalui halaman *login* admin dan melihat daftar pengaduan yang masuk. Admin melakukan proses verifikasi dan penanganan pengaduan, kemudian memperbarui status laporan sesuai dengan hasil penanganan. Perubahan status tersebut dapat langsung dilihat oleh pengguna melalui *dashboard*, sehingga proses pengaduan menjadi lebih transparan dan terpantau.

2.4 Perancangan Basis Data

Perancangan basis data pada Sistem Informasi Pengaduan Pengguna Berbasis Web menggunakan model relasional (*Relational Database Management System*). Basis data dirancang untuk menyimpan dan mengelola data pengguna, kategori pengaduan, serta laporan pengaduan secara terstruktur dan terintegrasi. *Database* yang digunakan adalah MariaDB/MySQL, dengan tujuan memudahkan proses penyimpanan, pencarian, serta pengelolaan data pengaduan secara terpusat.

Struktur basis data dalam sistem ini terdiri dari beberapa tabel utama, yaitu sebagai berikut:

1. Tabel *Users*

   Tabel ini digunakan untuk menyimpan data pengguna sistem, baik mahasiswa maupun admin. Atribut yang disimpan meliputi `id`, `username`, `password`, `full_name`, `role`, dan `created_at`. Kolom role menggunakan tipe `ENUM` dengan nilai admin dan mahasiswa untuk membedakan hak akses pengguna dalam sistem.
2. Tabel *Categories*

   Tabel ini digunakan untuk menyimpan data laporan pengaduan yang diajukan oleh pengguna. Atribut yang disimpan meliputi `id`, `user_id`, `category_id`, `ticket_number`, `title`, `description`, `status`, `report_date`, `evidence_photo`, `response_message`, dan `resolved_at`.
3. Tabel *Complaints*

   Tabel *complaints* memiliki relasi dengan tabel `users` melalui `user_id` sebagai f*oreign key*, serta relasi dengan tabel `categories` melalui `category_id`. Relasi ini bertujuan untuk menunjukkan siapa pengaju pengaduan dan kategori pengaduan yang dipilih.

2.5 Perancangan Antarmuka (*User Interface*)

Perancangan antarmuka pada Sistem Informasi Pengaduan Pengguna Berbasis Web dilakukan dengan mengedepankan prinsip kesederhanaan, kejelasan informasi, dan kemudahan penggunaan. Antarmuka dirancang agar dapat digunakan dengan baik oleh pengguna awam tanpa memerlukan pemahaman teknis yang mendalam.

1. *Dashboard* Pengguna

   Halaman dashboard pengguna menampilkan ringkasan informasi terkait pengaduan yang telah diajukan, seperti jumlah pengaduan dan status laporan (*pending*, proses, selesai, atau ditolak). Informasi disajikan secara ringkas agar mudah dipahami oleh pengguna.
2. Form Pengajuan Pengaduan

   Form pengaduan dirancang secara sederhana dan terstruktur, meliputi pemilihan kategori pengaduan, pengisian judul laporan, deskripsi permasalahan, serta tanggal kejadian. Desain form ini bertujuan untuk memudahkan pengguna dalam menyampaikan laporan secara jelas dan lengkap.
3. Riwayat Pengaduan

   Halaman riwayat pengaduan menampilkan daftar laporan yang telah diajukan oleh pengguna dalam bentuk tabel. Setiap laporan dilengkapi dengan informasi status,

tanggapan petugas, dan bukti foto sehingga pengguna dapat memantau perkembangan penanganan pengaduan.

4. Antarmuka Admin

   Antarmuka admin menampilkan daftar seluruh pengaduan yang masuk dalam bentuk tabel yang informatif. Admin dapat melihat detail pengaduan, memberikan tanggapan, serta memperbarui status laporan melalui tombol aksi yang tersedia. Desain ini bertujuan untuk meningkatkan efisiensi admin dalam memproses pengaduan.

BAB III
IMPLEMENTASI DAN EVALUASI SISTEM

3.1 Implementasi *Backend*

3.1.1 Struktur Folder *Backend*

Gambar 3.1 struktur folder *backend*

Pada struktur folder `uas-backend`, terdapat beberapa folder dan file yang penting dalam pengembangan aplikasi Sistem Pengaduan. Berikut adalah penjelasan mengenai isi folder dan file yang ada:

1. **config/**: Menyimpan file konfigurasi aplikasi, seperti pengaturan koneksi *database*.
2. **controllers/**: Berisi file-file yang menangani logika bisnis, seperti `authController.js` untuk autentikasi dan `complaintController.js` untuk pengelolaan pengaduan.

3. **middleware/**: Menyimpan *middleware* yang digunakan untuk memproses permintaan, seperti `authMiddleware.js` untuk autentikasi dan `validate.js` untuk validasi *input*.
4. **models/**: Berisi file untuk mendefinisikan struktur data (*model*), seperti `userModel.js` dan `complaintModel.js`, yang berinteraksi dengan *database*.
5. **routes/**: Menyimpan file rute API, seperti `authRoutes.js` untuk autentikasi dan `complaintRoutes.js` untuk pengaduan.
6. **.env**: Menyimpan variabel lingkungan yang sensitif, seperti JWT *secret key* dan informasi konfigurasi lainnya.
7. **server.js**: Merupakan *entry point* aplikasi yang mengonfigurasi dan menjalankan *server*.

3.1.2 Konfigurasi Environment (.env)

File .env sangat penting untuk menyimpan informasi sensitif yang tidak boleh dimasukkan langsung ke dalam kode sumber, seperti JWT *secret key*, *database connection strings*, dan *environment* variabel lainnya. Di mana konfigurasi paa sistem ini yaitu:

```
PORT=5000
DB_HOST=localhost
DB_USER=root
DB_PASS=
DB_NAME=campus_complaint_db
JWT_SECRET=rahasia_negara_api
```

3.1.3 *Middleware* Autentikasi

Di folder *middleware*, terdapat `authMiddleware.js` yang bertugas untuk memverifikasi token JWT yang dikirimkan dalam *header Authorization* pada *request*. Token ini digunakan untuk memastikan bahwa pengguna yang mengakses *endpoint* tertentu telah terautentikasi dengan benar. *Middleware* ini memastikan bahwa hanya pengguna yang telah terautentikasi yang dapat mengakses rute-rute tertentu di aplikasi, serta `validate.js` digunakan untuk memvalidasi data yang dikirimkan oleh pengguna di *request body*, terutama untuk rute-rute yang memerlukan *input* dari pengguna, seperti registrasi atau pembuatan pengaduan.

Secara keseluruhan `authMiddleware.js` memastikan pengguna yang mengakses rute dilindungi oleh autentikasi token JWT yang valid. Dan `validate.js` memastikan bahwa data yang dikirimkan oleh pengguna (seperti saat registrasi atau membuat pengaduan) valid sesuai dengan aturan yang ditentukan. Kedua *middleware* ini bekerja sama untuk memastikan aplikasi *backend* aman, hanya menerima data yang valid, dan hanya memberikan akses kepada pengguna yang terautentikasi dengan benar.

3.1.4 Implementasi Logika Bisnis (*Controller* & *Services*)

Di folder *controllers*, terdapat file `authController.js` dan `complaintController.js` yang menangani logika bisnis untuk pendaftaran pengguna dan pengaduan. Di dalam *controller* ini, `authController.js` dan `complaintController.js` bertanggung jawab untuk menangani *request* yang datang dari klien, memproses data yang diterima, berinteraksi dengan model (*database*), dan mengembalikan respon yang sesuai.

Contoh *code* `authController.js` (Implementasi *Login* & *Register*):

```
const bcrypt = require('bcryptjs');
const jwt = require('jsonwebtoken');
const UserModel = require('../models/userModel');

const AuthController = {

    register: async (req, res) => {
        try {
            const { username, password, full_name, role } = req.body;



            const existingUser = await UserModel.findByUsername(username);
            if (existingUser) {
                return res.status(400).json({ message: "Username sudah dipakai" });
            }



            const salt = await bcrypt.genSalt(10);
```

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

            const userRole = role || 'mahasiswa';
            await UserModel.create(username, hashedPassword, full_name, userRole);

            res.status(201).json({ message: "Registrasi berhasil, silakan login" });

        } catch (error) {
            res.status(500).json({ message: "Server Error", error: error.message });
        }
    },

    login: async (req, res) => {
        try {
            const { username, password } = req.body;

            const user = await UserModel.findByUsername(username);
            if (!user) {
                return res.status(404).json({ message: "User tidak ditemukan" });
            }

            const isMatch = await bcrypt.compare(password, user.password);
            if (!isMatch) {
                return res.status(400).json({ message: "Password salah" });
            }

            const token = jwt.sign(
                { id: user.id, role: user.role },
                process.env.JWT_SECRET,
                { expiresIn: '1d' }
            );
```

```
            res.json({
                message: "Login berhasil",
                token: token,
                user: {
                    id: user.id,
                    username: user.username,
                    role: user.role
                }
            });

        } catch (error) {
            res.status(500).json({ message: "Server Error",
error: error.message });
        }
    }
};

module.exports = AuthController;
```

3.2 Implementasi *Frontend* (Vue.js 3)

Proyek *frontend* ini menggunakan Vue.js 3 untuk membangun antarmuka pengguna. Struktur foldernya dirancang dengan jelas untuk memisahkan komponen-komponen, rute, dan pengelolaan status aplikasi. Berikut adalah penjelasan lebih rinci untuk setiap bagian yang relevan.

3.2.1 *Routing* dan Navigasi

Navigasi antar halaman dikelola oleh *Vue Router*. Contohnya ialah `router/index.js` file ini berisi konfigurasi untuk *routing* di aplikasi `Vue.js`. Di sini, kita akan mendefinisikan rute yang mengarahkan pengguna ke halaman-halaman yang berbeda dalam aplikasi, seperti *dashboard*, *login*, dan pengaduan

Contoh kode `index.js`:

```
import { createRouter, createWebHistory } from 'vue-
router';
import Login from '../views/Login.vue';
import Dashboard from '../views/Dashboard.vue';
import ComplaintForm from '../views/ComplaintForm.vue';
import ComplaintDetail from
'../views/ComplaintDetail.vue';
import EditComplaint from '../views/EditComplaint.vue';
import AdminDashboard from
'../views/AdminDashboard.vue';
import NewsFeed from '../views/NewsFeed.vue';
```

```
const routes = [
    { path: '/', name: 'Login', component: Login },
    {
        path: '/dashboard',
        name: 'Dashboard',
        component: Dashboard,
        meta: { requiresAuth: true }
    },
    {
        path: '/complaint/create',
        name: 'CreateComplaint',
        component: ComplaintForm,
        meta: { requiresAuth: true }
    },
    {
        path: '/complaint/:id',
        name: 'ComplaintDetail',
        component: ComplaintDetail,
        meta: { requiresAuth: true }
    },
    {
        path: '/complaint/:id/edit',
        name: 'EditComplaint',
        component: EditComplaint,
        meta: { requiresAuth: true }
    },

    {
        path: '/admin-dashboard',
        name: 'AdminDashboard',
        component: AdminDashboard,
        meta: { requiresAuth: true }
    },

    {
        path: '/news',
        name: 'NewsFeed',
        component: NewsFeed,
        meta: { requiresAuth: true }
    }
];

const router = createRouter({
    history: createWebHistory(),
    routes
});

// Middleware Frontend (Proteksi Halaman)
router.beforeEach((to, from, next) => {
    const token = localStorage.getItem('token');

    if (to.meta.requiresAuth && !token) {
        next('/'); // Kalau gak ada token, tendang ke
Login
    } else {
        next();
    }
});
```

```
export default router;
```

3.2.2 Integrasi API

Integrasi API dilakukan melalui Postman untuk membuat permintaan HTTP ke *backend*. Misalnya, ketika pengguna *login*, permintaan dikirim ke *endpoint* `/api/auth/login` untuk memverifikasi kredensial, dan token yang diterima akan disimpan di status global. Dengan token tersebut, *frontend* akan mengautentikasi permintaan selanjutnya ke *backend* dengan menyertakan token JWT dalam *header Authorization*.

3.2.3 Komponen Manajemen Data

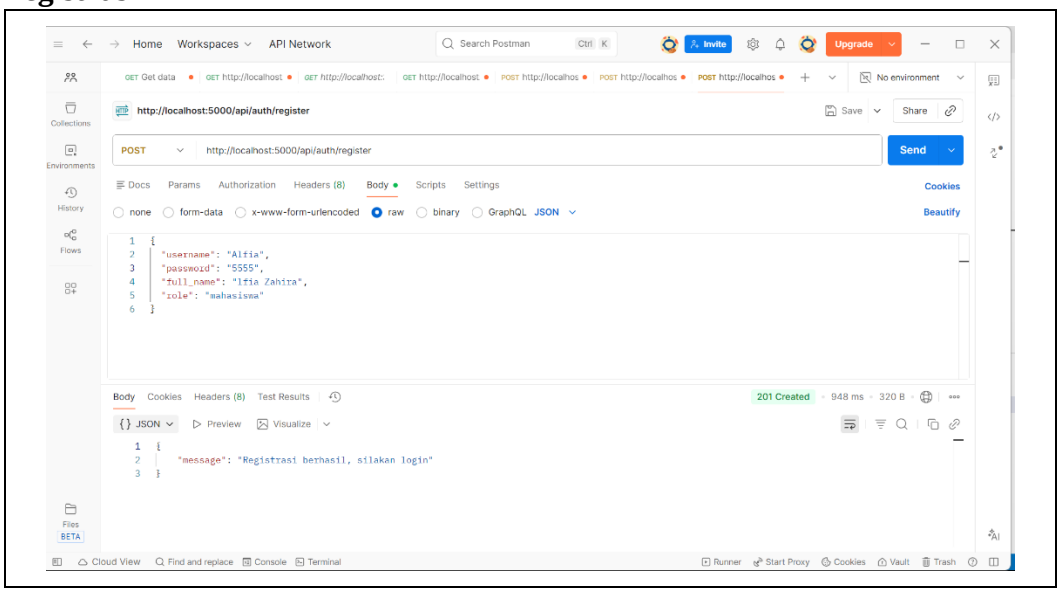
1. Components/: Komponen-komponen di dalam folder ini menangani tampilan dan interaksi pengguna. Di dalamnya ada beberapa file seperti `HelloWorld.vue` yang berfungsi untuk tampilan awal atau bisa digunakan untuk menampilkan data dinamis yang diambil dari API.
2. Views/: Folder ini berisi tampilan utama halaman yang digunakan oleh rute-rute yang telah didefinisikan di *router*. Setiap halaman memiliki komponen Vue-nya sendiri, misalnya:
    - `AdminDashboard.vue`: Berisi tampilan untuk admin yang bisa melihat dan mengelola pengaduan.
    - `ComplaintDetail.vue`: Menampilkan detail pengaduan tertentu.
    - `ComplaintForm.vue`: Formulir untuk membuat pengaduan baru.
    - `EditComplaint.vue`: Halaman untuk mengedit pengaduan yang sudah ada

BAB IV

PENGUJIAN PROGRAM

4.1 Pengujian API (Postman)

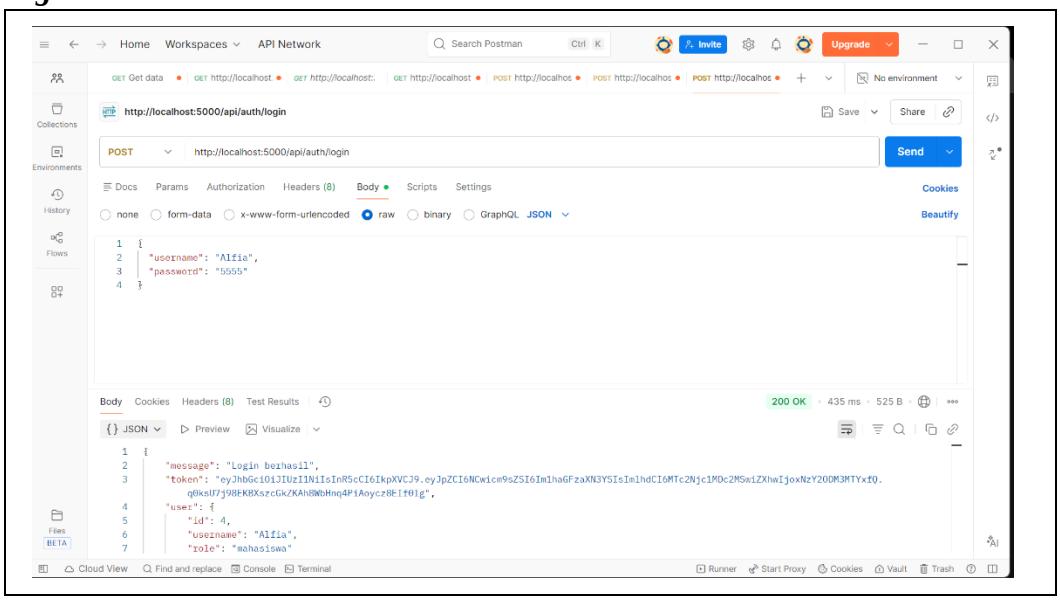
4.1.1 Registrasi



Gambar 4.1 Registrasi

Gambar di atas merupakan menunjukkan pengujian *endpoint* `POST /api/auth/register` menggunakan Postman untuk proses registrasi pengguna. Data dikirim dalam format JSON berisi `username`, `password`, `nama lengkap`, dan `role`.

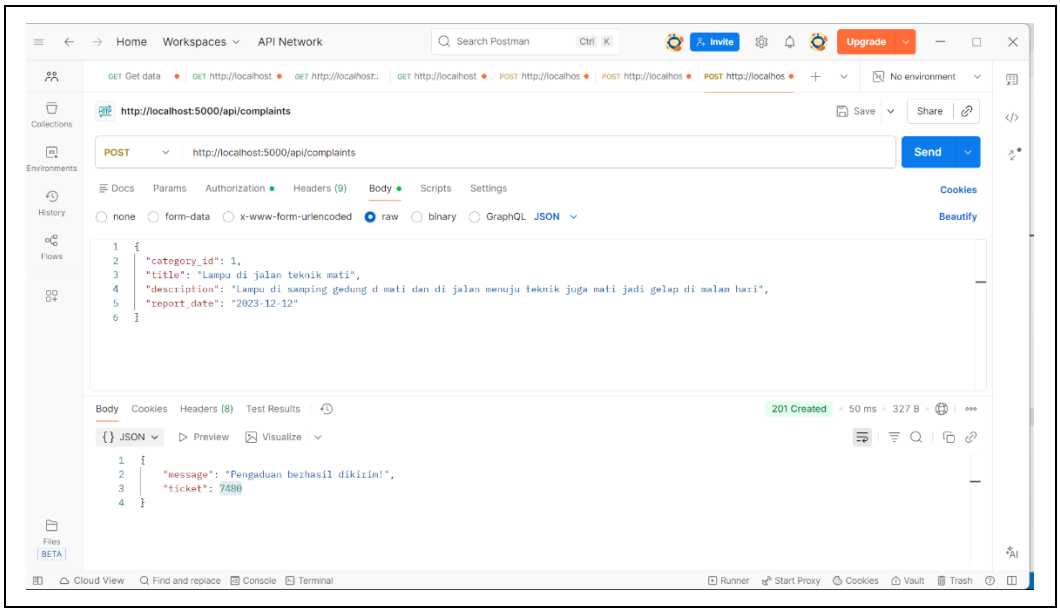
4.1.2 *Login*



Gambar 4.2 *Login*

Gambar tersebut menunjukkan pengujian *endpoint* `POST /api/auth/login` menggunakan Postman untuk proses autentikasi pengguna. *Username* dan *password* dikirim dalam format JSON.

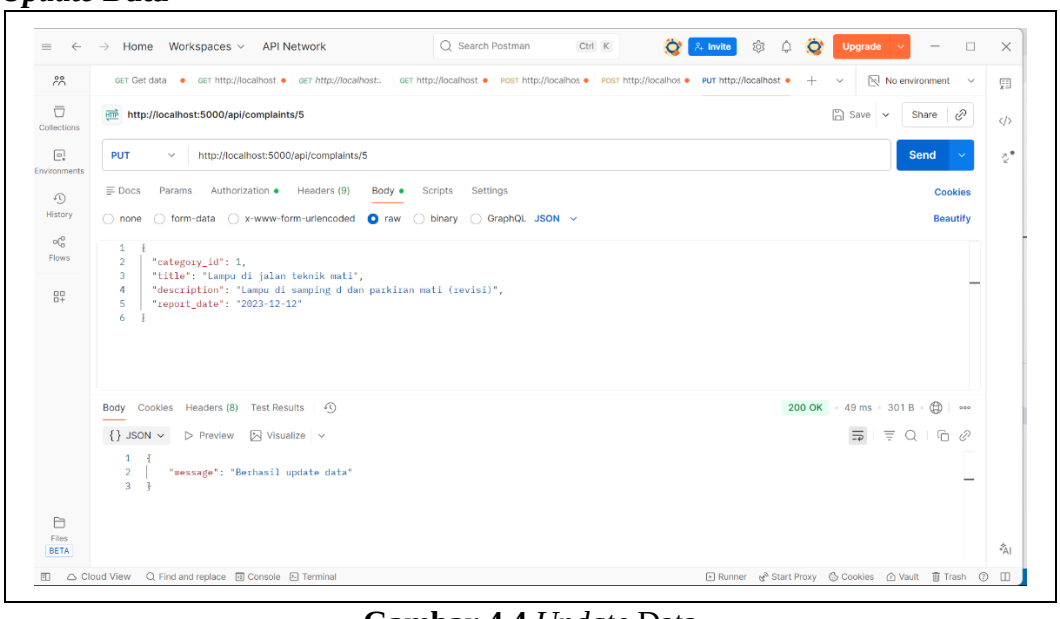
4.1.3 *Create* Data



Gambar 4.3 *Create* Data

Gambar tersebut menunjukkan pengujian *endpoint* `POST /api/complaints` menggunakan Postman untuk mengirim data pengaduan. Data dikirim dalam format JSON yang memuat kategori, judul, deskripsi, dan tanggal laporan dengan autentikasi pengguna.

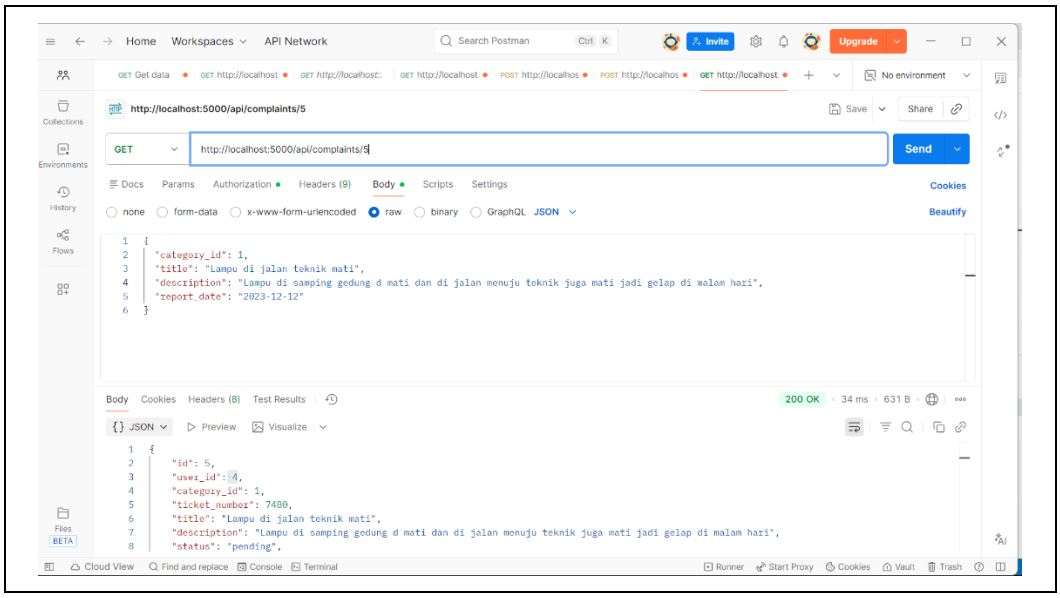
4.1.4 *Update* Data



Gambar 4.4 *Update* Data

Gambar tersebut menunjukkan pengujian *endpoint* `PUT /api/complaints/{id}` menggunakan Postman untuk memperbarui data pengaduan. Perubahan data dikirim dalam format JSON berdasarkan ID pengaduan tertentu dengan autentikasi pengguna.

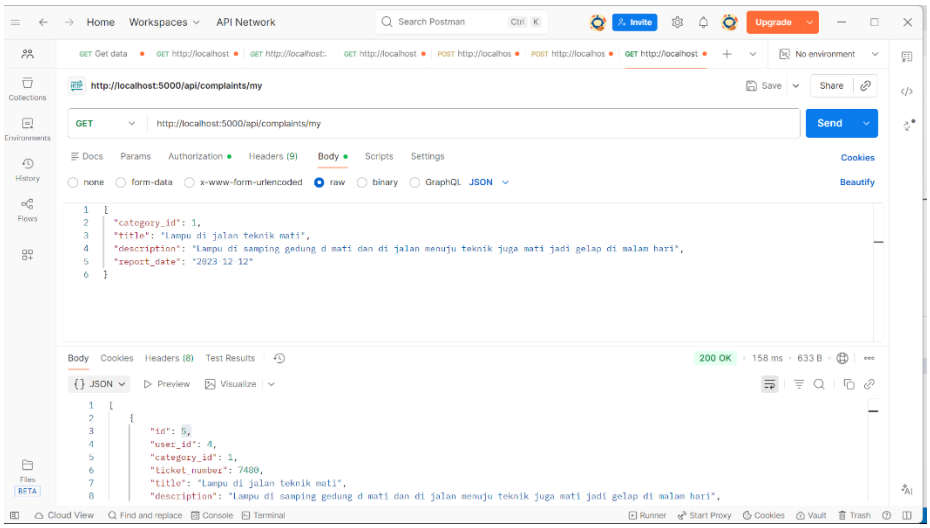
4.1.5 *Read* Detail Data



Gambar 4.5 *Read* Detail Data

Gambar tersebut menunjukkan pengujian *endpoint* `GET /api/complaints/{id}` menggunakan Postman untuk mengambil detail data pengaduan. *Request* dikirim ke ID pengaduan tertentu dan server mengembalikan respon dalam format JSON berisi informasi lengkap pengaduan.

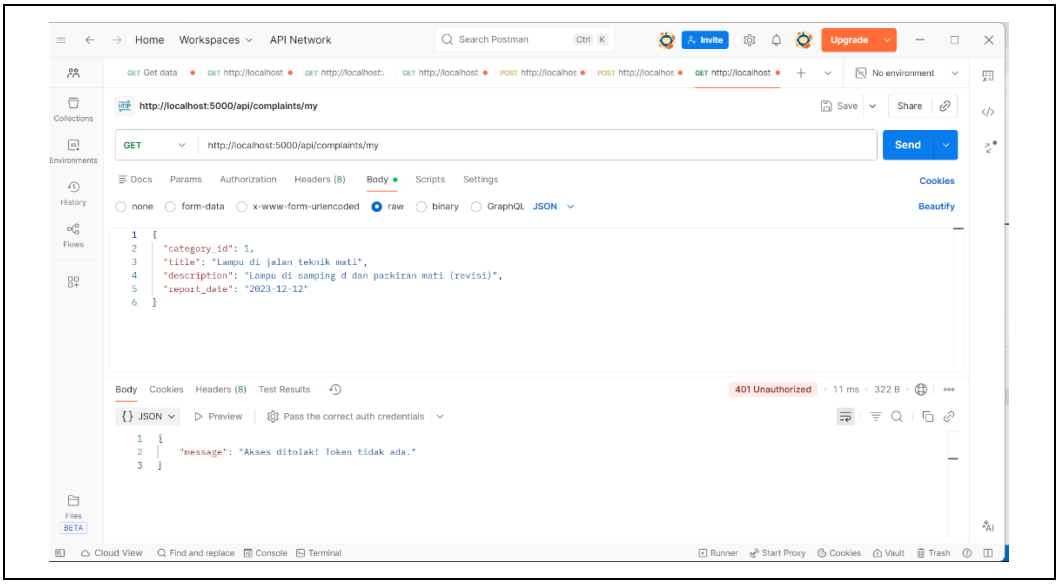
4.1.6 *Read List* Data



Gambar 4.6 *Read List* Data

Gambar tersebut menunjukkan pengujian *endpoint* `GET /api/complaints/my` menggunakan Postman untuk mengambil daftar pengaduan milik pengguna yang sedang login. *Request* berhasil dan server mengembalikan respon berupa *list* data pengaduan dalam format JSON berdasarkan autentikasi pengguna.

4.1.7 *Protected Route*

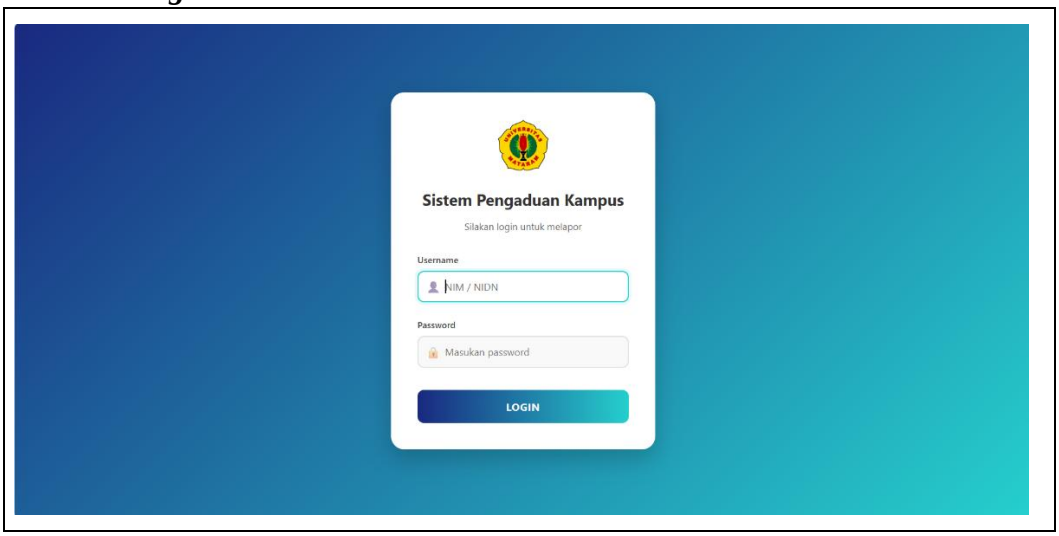


Gambar 4.7 *Protected Route*

Gambar tersebut menunjukkan pengujian *endpoint* `GET /api/complaints/my` yang merupakan *protected route*. Pengguna harus menyertakan token akses yang valid agar dapat mengakses data pengaduan miliknya, sehingga keamanan data tetap terjaga.

4.2 Pengujian Aplikasi

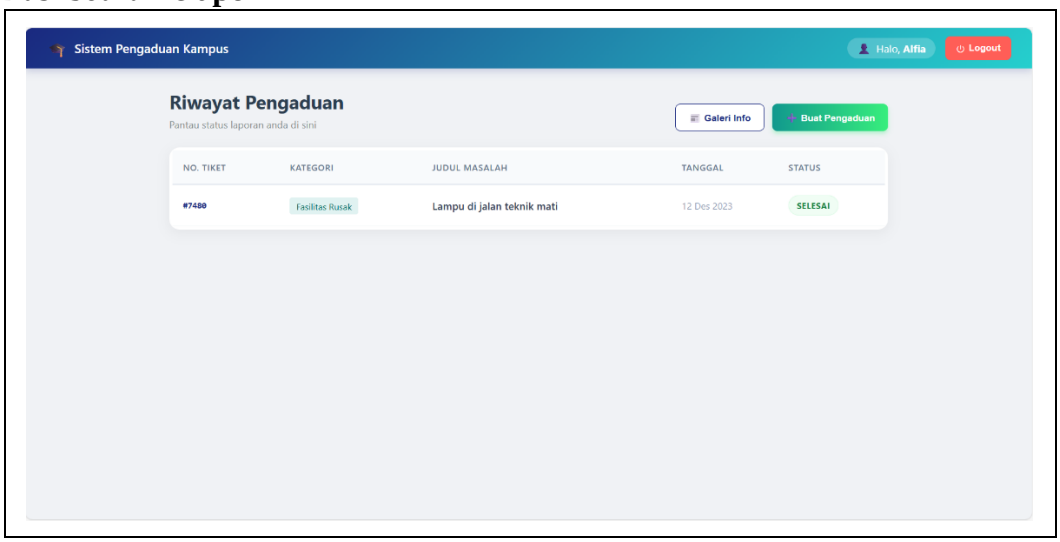
4.2.1 Halaman *Login User*



Gambar 4.8 Halaman *Login user*

Gambar tersebut menampilkan halaman *login* pengguna pada Sistem Pengaduan Kampus. Pengguna diminta memasukkan NIM/NIDN dan *password* sebagai proses autentikasi sebelum dapat mengakses dan mengelola layanan pengaduan.

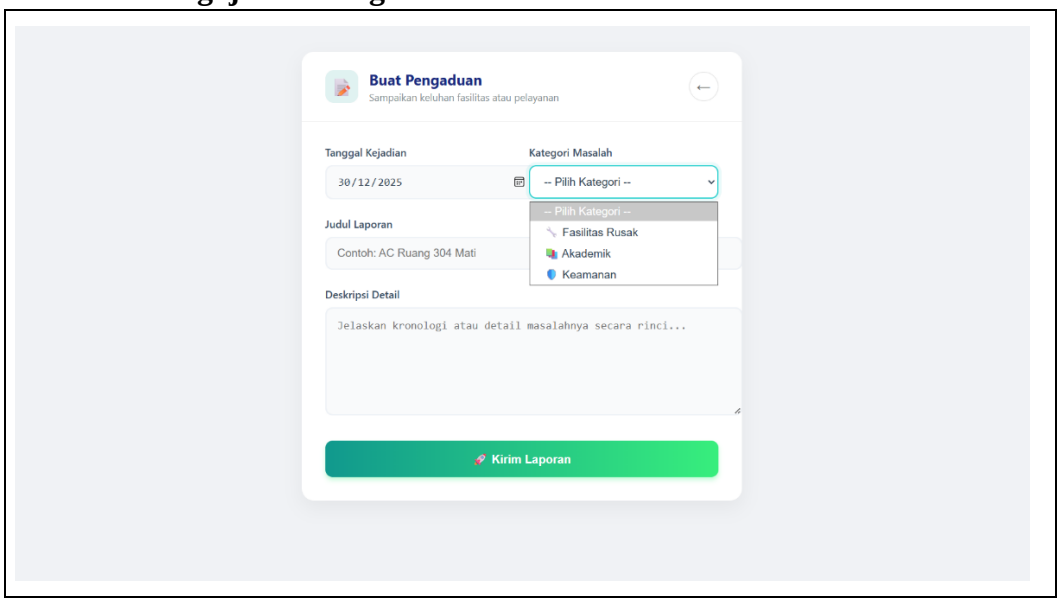
4.2.2 *Dashboard* Pelapor



Gambar 4.9 *Dashboard* Pelapor

Gambar tersebut menampilkan halaman *dashboard* Riwayat Pengaduan pada Sistem Pengaduan Kampus. Halaman ini digunakan pengguna untuk melihat daftar pengaduan yang telah dibuat beserta kategori, tanggal, dan status penyelesaian, serta menyediakan fitur membuat pengaduan baru.

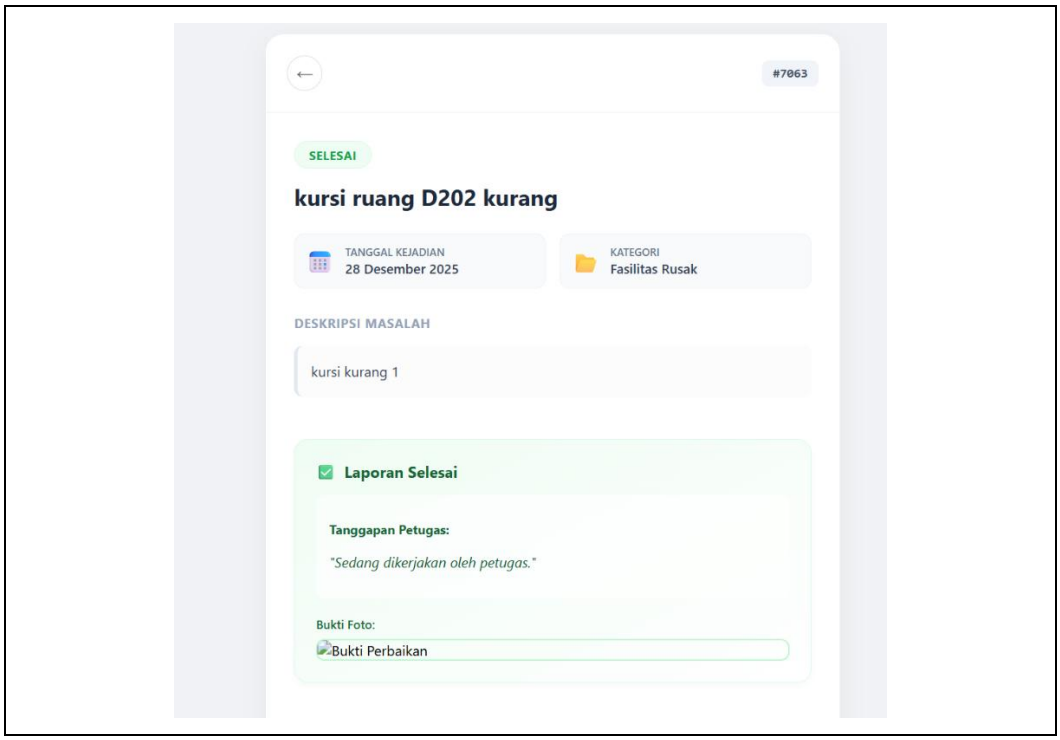
4.2.3 Halaman Mengajukan Pengaduan



Gambar 4.10 Halaman Melngajukan Pegaduan

Gambar tersebut menampilkan halaman untuk Pengaduan dimana pengguna mengisi tanggal kejadian, kategori masalah, judul laporan, dan deskripsi detail. Tersedia tombol Kirim Laporan untuk mengirimkan pengaduan.

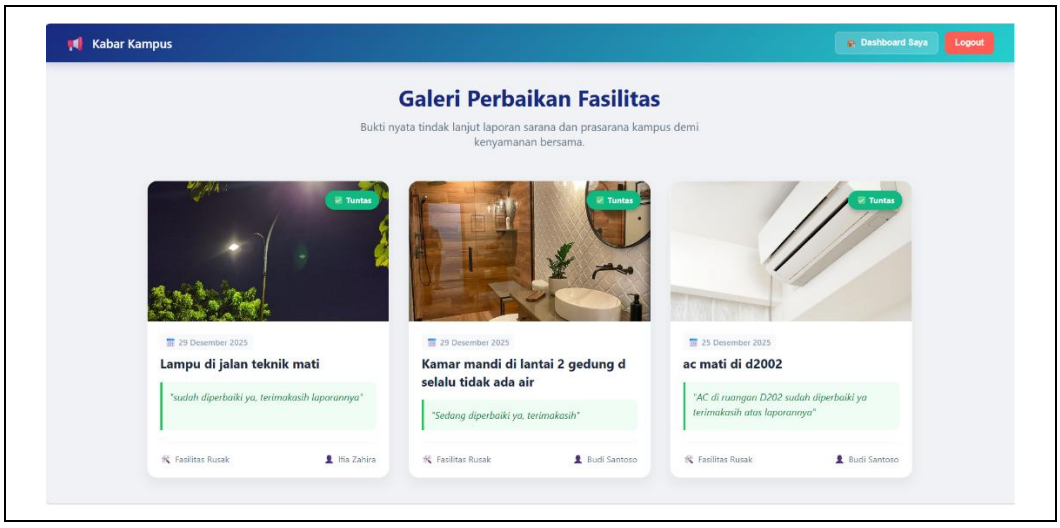
4.2.4 Melihat *Feedback* Dari Admin



Gambar 4.11 Melihat *Feedback* Dari Admin

Gambar tersebut menampilkan halaman detail laporan pengaduan yang telah selesai. Terlihat judul laporan, tanggal kejadian, kategori masalah, dan deskripsi singkat. Serta tanggapan dan bukti foto sebagai hasil tindak lanjut laporan yang diberikan oleh admin.

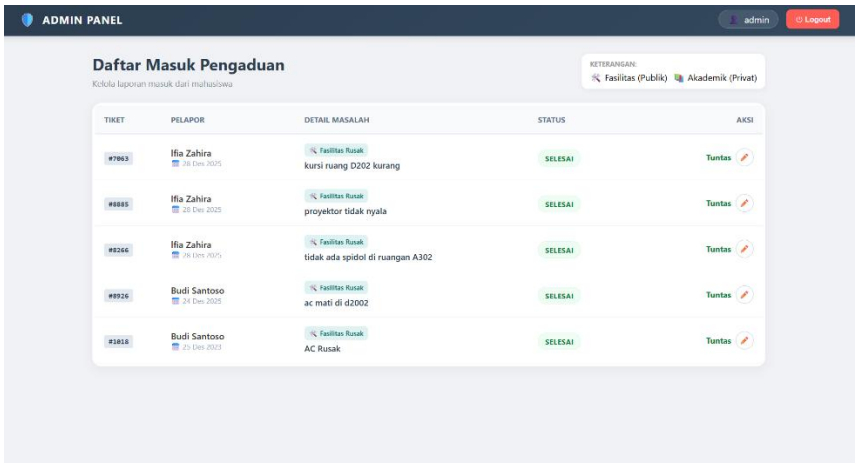
4.2.5 Galeri Perbaikan Fasilitas



Gambar 4.12 Galeri Perbaikan Fasilitas

Gambar tersebut menampilkan halaman Galeri Perbaikan Fasilitas pada aplikasi Kabar Kampus. Halaman ini berisi daftar laporan fasilitas yang sudah tuntas lengkap dengan foto, tanggal, dan tanggapan petugas. Tujuannya sebagai bukti tindak lanjut perbaikan sarana dan prasarana kampus.

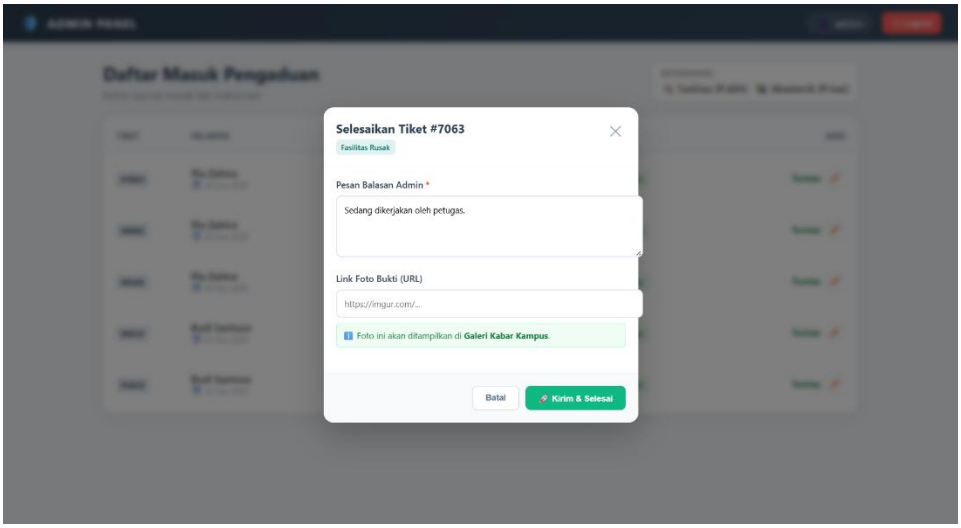
4.2.6 *Dashboard* Admin



Gambar 4.13 *Dashboard* Admin

Gambar tersebut menampilkan Admin *Panel* dengan halaman Daftar Masuk Pengaduan dari mahasiswa. Terdapat tabel berisi tiket, pelapor, detail masalah fasilitas, status laporan, dan aksi.

4.2.7 *Feedback* Dari Admin



Gambar 4.14 *Feedback* Dari Admin

Halaman ini menampilkan *form feedback* admin untuk menyelesaikan tiket pengaduan. Admin dapat menuliskan pesan balasan, menambahkan link foto bukti, lalu mengirim dan menandai laporan sebagai selesai. Foto bukti akan ditampilkan di Galeri Kabar Kampus sebagai dokumentasi.

BAB V
PENUTUP

5.1 Kesimpulan

Pada proyek ini, telah berhasil dikembangkan sebuah "Sistem Pengaduan Mahasiswa berbasis Web" yang memanfaatkan teknologi Vue.js 3 di sisi frontend dan Node.js dengan *Express* di sisi *backend*. Sistem ini memungkinkan mahasiswa untuk mengajukan pengaduan terkait masalah yang mereka hadapi di kampus, serta memberikan solusi bagi pihak yang berwenang untuk menindaklanjuti pengaduan tersebut.

Implementasi *frontend* menggunakan Vue.js 3 memungkinkan pengembangan antarmuka yang interaktif dan responsif. Vue Router digunakan untuk mengatur navigasi antar halaman, sedangkan Vuex berfungsi untuk manajemen status aplikasi, termasuk autentikasi pengguna dan pengelolaan token JWT.

Sisi *backend* dibangun menggunakan Node.js dan *Express*, dengan JWT sebagai mekanisme autentikasi yang mengamankan akses pengguna. *Backend* juga mengelola komunikasi dengan *database*, mengelola pengaduan yang diajukan, serta memastikan integritas dan keamanan data yang dikirim dan diterima antara *frontend* dan *backend.*

Struktur sistem ini memastikan pemisahan yang jelas antara *frontend* dan *backend*, sehingga aplikasi lebih mudah untuk dikembangkan dan dipelihara. Implementasi *middleware* yang mengatur verifikasi token, serta validasi data *input*, memberikan lapisan keamanan yang penting untuk aplikasi ini.

Secara keseluruhan, proyek ini berhasil mengintegrasikan berbagai teknologi web untuk menciptakan sistem pengaduan yang tidak hanya fungsional, tetapi juga aman dan mudah diakses oleh pengguna.

5.2 Saran

Meskipun sistem ini sudah berfungsi dengan baik, terdapat beberapa saran yang dapat meningkatkan kinerja dan kualitas sistem, baik dari sisi teknis maupun pengalaman pengguna:

1. Notifikasi Pengaduan

   Pengguna dan admin dapat diberi notifikasi terkait status pengaduan mereka, misalnya ketika pengaduan diproses, diselesaikan, atau membutuhkan tindakan lebih lanjut. Hal ini dapat meningkatkan interaksi dan keterlibatan pengguna dengan aplikasi.
2. Fitur Pencarian Pengaduan

Penambahan fitur pencarian berdasarkan kategori atau kata kunci akan membantu pengguna dan admin untuk mencari pengaduan dengan lebih efisien, terutama ketika jumlah pengaduan meningkat.

3. *Feedback* dari Pengguna

   Fitur *feedback* atau rating mengenai pengaduan yang telah diselesaikan dapat membantu admin untuk mengevaluasi kualitas respon terhadap pengaduan dan meningkatkan pelayanan di masa depan

Dengan memperhatikan saran-saran di atas, aplikasi Sistem Pengaduan Mahasiswa ini dapat ditingkatkan baik dari segi fungsionalitas, keamanan, kinerja, dan pengalaman pengguna, sehingga dapat memberikan manfaat yang lebih besar bagi pengguna di masa yang akan datang.