SISTEM MANAJEMEN TUGAS MAHASISWA

Mata Kuliah: Pemrograman Web Lanjut

Disusun Oleh:

Muh. Ressa Arsy Ma'rif	F1D022137
Fadila Rahmania	F1D02310048
Fitri Nufa Dastana	F1D02310052

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS MATARAM

TAHUN 2025

BAB I - DESKRIPSI PROJECT

1. Latar Belakang

Perkembangan teknologi informasi yang semakin pesat telah membawa perubahan signifikan dalam berbagai aspek kehidupan, termasuk di bidang pendidikan tinggi. Mahasiswa dituntut untuk mampu mengelola berbagai aktivitas akademik secara mandiri, seperti mengerjakan tugas individu, tugas kelompok, hingga proyek akhir mata kuliah. Aktivitas tersebut sering kali memiliki tenggat waktu yang berbeda-beda serta melibatkan banyak pihak, sehingga membutuhkan pengelolaan yang terstruktur dan efisien.

Pada praktiknya, banyak mahasiswa masih mengalami kesulitan dalam mengatur dan memantau tugas yang sedang dikerjakan. Pengelolaan tugas yang dilakukan secara manual, seperti mencatat di buku, menggunakan pesan singkat, atau mengandalkan ingatan pribadi, sering kali menimbulkan permasalahan seperti lupa tenggat waktu, miskomunikasi dalam kerja kelompok, serta kesulitan memantau progres pekerjaan. Kondisi ini dapat berdampak pada menurunnya produktivitas, keterlambatan pengumpulan tugas, dan kurang optimalnya hasil pembelajaran.

Berdasarkan permasalahan tersebut, dikembangkan sebuah Sistem Manajemen Tugas Mahasiswa berbasis web. Sistem ini dirancang untuk membantu mahasiswa dalam mencatat tugas, mengatur jadwal, serta memantau progres pengerjaan. Dengan adanya sistem ini, diharapkan mahasiswa dapat meningkatkan kedisiplinan, efisiensi waktu, serta kualitas kerja dalam menyelesaikan tugas.

Pengembangan sistem berbasis web dipilih karena kemudahan aksesnya yang dapat digunakan kapan saja dan di mana saja melalui perangkat yang terhubung dengan internet. Selain itu, sistem ini juga diharapkan dapat menjadi solusi digital yang mendukung proses pembelajaran modern dan sejalan dengan penerapan teknologi informasi di lingkungan perguruan tinggi.

2. Tujuan Project

Tujuan dari pengembangan Sistem Manajemen Tugas Mahasiswa ini adalah sebagai berikut:

1. Mengembangkan sistem berbasis web yang dapat membantu mahasiswa dalam mengelola tugas secara terstruktur dan terintegrasi.
2. Menyediakan media pencatatan tugas yang memudahkan mahasiswa dalam mengatur jadwal, tenggat waktu, serta prioritas pekerjaan.
3. Membantu mahasiswa memantau progres pengerjaan tugas, sehingga dapat meningkatkan kedisiplinan dan efektivitas dalam menyelesaikan aktivitas akademik.

4. Mengurangi risiko keterlambatan dan kelalaian dalam penyelesaian tugas akibat pengelolaan yang masih dilakukan secara manual.

3. Teknologi yang Digunakan (*Backend*, *Frontend*, *Database*)

    1. *Backend*

        Backend berfungsi sebagai penyedia layanan API dan logika utama aplikasi. Teknologi yang digunakan meliputi:

        a. Node.js & Express.js

            Digunakan sebagai kerangka kerja utama server untuk menangani permintaan HTTP dan *routing*.

        b. JSON Web Token (JWT)

            Digunakan untuk autentikasi pengguna yang aman (sistem *login* dan proteksi *route*).

        c. Bcryptjs

            Digunakan untuk enkripsi (*hashing*) kata sandi pengguna sebelum disimpan ke dalam database demi keamanan.

        d. MySQL2

            Library driver untuk menghubungkan aplikasi Node.js dengan database MySQL.

        e. Dotenv

            Untuk manajemen variabel lingkungan (environment variables).

    2. *Frontend*

        Frontend bertugas menangani antarmuka pengguna (*User Interface*) dan interaksi pengguna. Teknologi yang digunakan meliputi:

        a. Vue.js 3

            Kerangka kerja JavaScript utama untuk membangun antarmuka yang reaktif dan berbasis komponen.

        b. Vite

            Build tool yang digunakan untuk mempercepat proses pengembangan dan kompilasi aset.

        c. Vue Router

            Mengatur navigasi antar halaman (seperti Dashboard, Login, Profile) tanpa memuat ulang halaman (*Single Page Application*).

        d. Axios

            Klien HTTP untuk melakukan permintaan data (GET, POST, PUT, DELETE) ke server Backend.

e. Chart.js / Vue-Chartjs

   Digunakan untuk menampilkan grafik statistik performa mingguan pengguna pada halaman dashboard.

3. *Database*

   a. MySQL

      Sistem manajemen basis data relasional (RDBMS) yang digunakan untuk menyimpan data pengguna (users), tugas utama (tasks), dan sub-tugas (subtasks). Struktur database mendukung relasi antar tabel (seperti *Foreign Key* user_id pada tabel tasks) untuk menjaga integritas data.

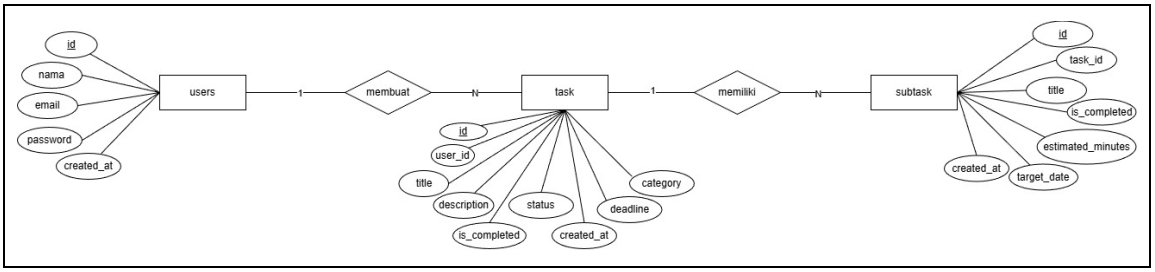
BAB II - PERANCANGAN SISTEM

1. Deskripsi Singkat Sistem

Sistem yang dirancang dalam proyek ini adalah sebuah manajemen tugas (*Task Management System*) berbasis web yang bertujuan untuk membantu pengguna mengorganisir aktivitas harian secara terstruktur guna meningkatkan produktivitas. Aplikasi ini dibangun menggunakan arsitektur *Client-Server* yang memisahkan logika bisnis di sisi *backend* (menggunakan Node.js dan Express.js) dengan antarmuka pengguna yang responsif di sisi *frontend* (menggunakan Vue.js), serta memanfaatkan MySQL sebagai basis data relasional untuk penyimpanan informasi. Demi menjaga privasi dan keamanan data, sistem menerapkan mekanisme autentikasi berbasis *JSON Web Token* (JWT), di mana setiap pengguna diwajibkan melakukan registrasi dan *login* untuk dapat mengakses serta mengelola data tugas pribadi mereka secara eksklusif.

Secara fungsional, sistem ini menyediakan fitur pengelolaan tugas yang komprehensif (CRUD) yang memungkinkan pengguna membuat tugas utama lengkap dengan atribut pendukung seperti deskripsi, kategori, tenggat waktu, dan status pengerjaan, serta kemampuan memecah pekerjaan besar menjadi sub-tugas yang lebih kecil. Untuk mendukung evaluasi kinerja, aplikasi juga dilengkapi dengan *dashboard* interaktif yang menampilkan ringkasan statistik dan grafik performa mingguan, sehingga pengguna dapat memantau progres penyelesaian tugas dan mengidentifikasi pekerjaan yang mendesak atau terlambat secara visual dan *real-time*.

2. Desain Database (ERD dan penjelasan singkat tabel)



Gambar 1. *Entity Relationship Diagram*

Berdasarkan *Entity Relationship Diagram* (ERD) yang dirancang, sistem ini memiliki tiga tabel utama, yaitu users, task, dan subtask. Tabel users digunakan untuk menyimpan data pengguna sistem, yang dalam hal ini adalah mahasiswa. Data yang tersimpan meliputi identitas pengguna seperti nama, email, kata sandi, serta waktu pembuatan akun. Tabel ini berperan sebagai entitas utama karena setiap aktivitas pengelolaan tugas dalam sistem selalu terkait dengan pengguna tertentu.

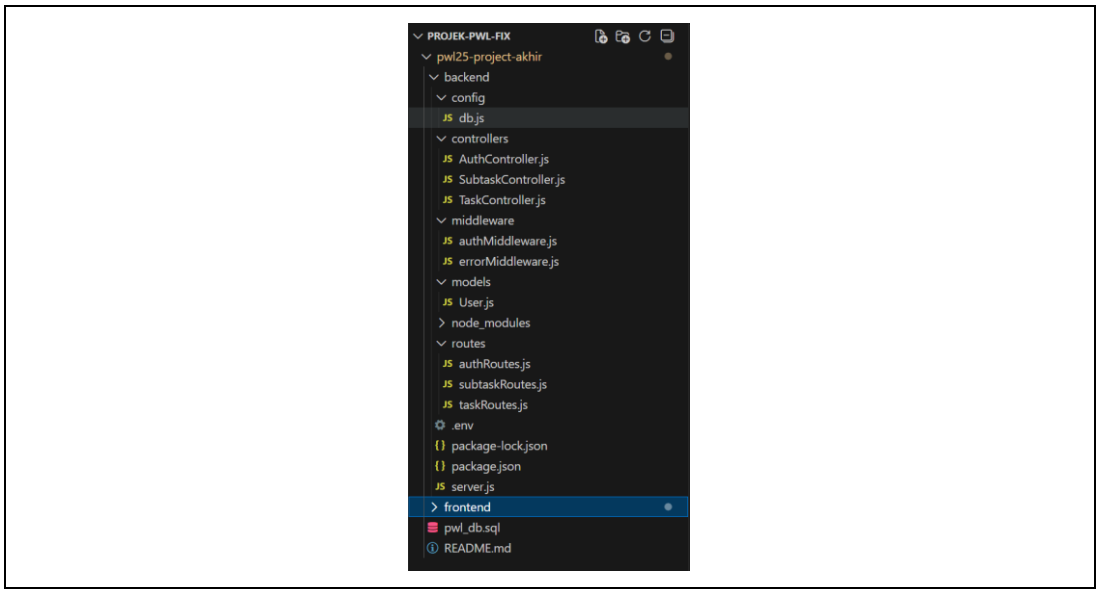
Tabel task digunakan untuk menyimpan informasi tugas atau proyek yang dibuat oleh pengguna. Setiap task terhubung dengan satu pengguna melalui atribut user_id, sehingga menunjukkan bahwa satu pengguna dapat memiliki banyak task. Informasi yang disimpan pada tabel ini meliputi judul task, deskripsi, kategori, status pengerjaan, penanda penyelesaian, batas waktu pengerjaan (deadline), serta waktu pembuatan task. Tabel ini berfungsi sebagai inti dalam pengelolaan aktivitas akademik mahasiswa.

Selanjutnya, tabel subtask digunakan untuk menyimpan rincian pekerjaan yang merupakan bagian dari sebuah task. Setiap subtask hanya terkait dengan satu task melalui atribut task_id, namun satu task dapat memiliki banyak subtask. Data yang tersimpan pada tabel ini mencakup judul subtask, status penyelesaian, estimasi waktu pengerjaan, target tanggal penyelesaian, serta waktu pembuatan subtask. Keberadaan tabel ini memungkinkan sistem untuk mengelola tugas secara lebih detail dan terstruktur.

Secara keseluruhan, relasi antar tabel dalam ERD ini membentuk hubungan one-to-many antara users dan task, serta antara task dan subtask. Struktur relasi tersebut mendukung pengelolaan tugas dan proyek mahasiswa secara hierarkis, mulai dari pengguna sebagai pemilik tugas, task sebagai pekerjaan utama, hingga subtask sebagai pembagian pekerjaan yang lebih rinci.

BAB III - IMPLEMENTASI

1. Implementasi *Backend* (struktur folder, API, *middleware*)
    1. Struktur Folder



Gambar 2. Struktur Folder *Backend*

Berikut adalah struktur direktori yang diterapkan:

a. config/

Berisi konfigurasi koneksi basis data menggunakan pustaka mysql2 untuk menghubungkan aplikasi dengan server database MySQL.

b. controllers/

Menyimpan logika bisnis utama. Setiap fungsi dalam *controller* bertanggung jawab menerima permintaan (*request*), memproses data, dan mengirimkan respons. Contohnya adalah TaskController.js untuk manajemen tugas dan AuthController.js untuk autentikasi.

c. middleware/

Berisi fungsi perantara yang dieksekusi sebelum mencapai *controller* utama, seperti verifikasi token JWT (authMiddleware.js) dan penanganan kesalahan global (errorMiddleware.js).

d. routes/

Mendefinisikan titik akhir (*endpoints*) API dan memetakan metode HTTP (GET, POST, PUT, DELETE) ke fungsi *controller* yang sesuai.

e. server.js

Titik masuk utama aplikasi yang menginisialisasi server Express, mengatur CORS, dan memuat variabel lingkungan.

2. Implementasi API

Sistem menyediakan layanan RESTful API yang memungkinkan komunikasi data antara klien dan server. Berikut adalah daftar *endpoint* utama yang telah diimplementasikan:

1. Autentikasi (/api/auth)
    a. POST /register: Mendaftarkan pengguna baru dengan enkripsi kata sandi.
    b. POST /login: Memvalidasi kredensial dan menerbitkan JSON Web Token (JWT).
    c. GET /profile-data: Mengambil informasi profil pengguna yang sedang login.
2. Manajemen Tugas (/api/tasks)
    a. GET /: Mengambil daftar tugas, mendukung filter berdasarkan waktu (hari ini/minggu ini) dan pengurutan status.
    b. POST /: Membuat tugas baru dengan detail kategori dan tenggat waktu.
    c. PATCH /:id/status: Memperbarui status tugas (misalnya: dari 'To Do' menjadi 'Done').
    d. DELETE /:id: Menghapus tugas dari sistem.
3. Penerapan Middleware
    1. Autentikasi (Auth Middleware)

    Middleware ini memverifikasi keberadaan dan kevalidan token JWT pada *header* permintaan. Jika token valid, informasi pengguna akan disisipkan ke dalam objek req.user, memastikan bahwa pengguna hanya dapat mengakses data milik mereka sendiri.

    2. Penanganan Kesalahan (Error Middleware)

    Middleware ini ditempatkan di akhir rantai eksekusi aplikasi (app.use(errorHandler)) untuk menangkap kesalahan yang tidak terduga dan mengembalikan respons format JSON yang standar kepada klien, mencegah aplikasi berhenti bekerja secara tiba-tiba.

2. Implementasi *Frontend* (halaman, *routing*, konsumsi API)
    1. Halaman (Views)

    Antarmuka aplikasi dibagi menjadi beberapa komponen halaman utama (*Views*) yang berada di direktori src/views:

    a. TaskDashboard.vue

    Merupakan halaman utama yang menampilkan ringkasan statistik (Total, Selesai, Progress) dan daftar tugas. Halaman ini dilengkapi dengan fitur interaktif seperti

checkbox penyelesaian tugas, indikator warna untuk tenggat waktu (*urgent/overdue*), dan formulir modal untuk menambah tugas baru.

b. LoginView.vue & RegisterView.vue

Halaman formulir untuk akses masuk dan pendaftaran pengguna.

c. PerformanceView.vue

Halaman yang menyajikan visualisasi data produktivitas pengguna dalam bentuk grafik mingguan.

d. ProfileView.vue

Halaman untuk menampilkan informasi akun pengguna dan opsi keluar (*logout*).

2. Manajemen Routing

Navigasi antar halaman dikelola oleh Vue Router, yang memungkinkan perpindahan tampilan tanpa memuat ulang halaman browser. Konfigurasi rute didefinisikan dalam berkas router/index.js, yang memetakan URL (seperti /, /login, /performance) ke komponen Vue yang sesuai.

3. Konsumsi API dan Integrasi Data

Komunikasi data dengan *backend* dilakukan menggunakan pustaka HTTP Client Axios. Pola integrasi yang diterapkan meliputi:

a. Penyisipan Token Otomatis

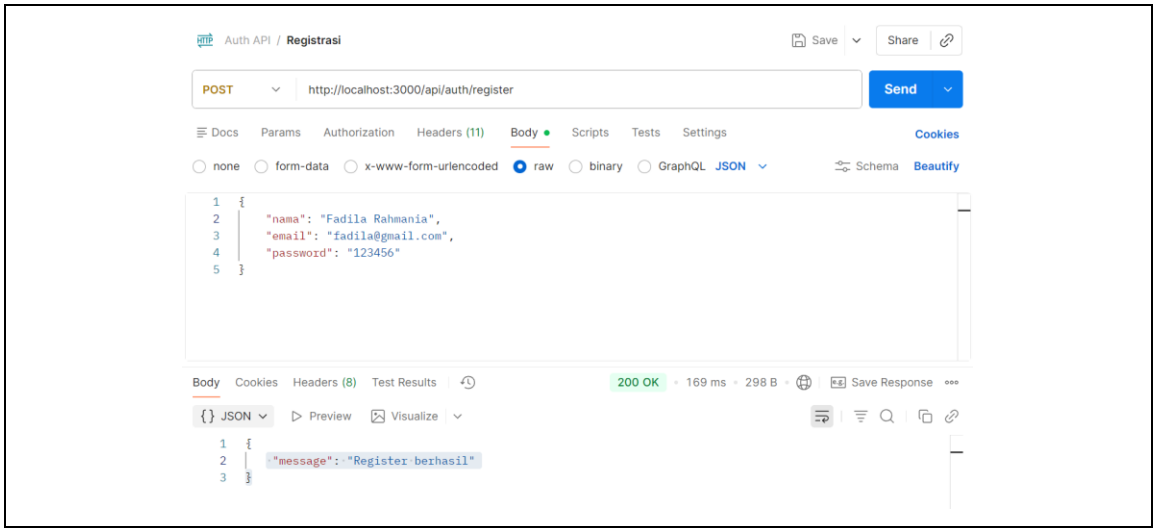
Pada halaman yang membutuhkan otorisasi (seperti Dashboard), token JWT yang tersimpan di localStorage diambil dan disisipkan ke dalam *header* Authorization pada setiap permintaan HTTP (Bearer ${token}). Hal ini terlihat pada implementasi fungsi fetchTasks dan authHeader.

b. Reaktivitas Data

Data yang diterima dari API (misalnya daftar tugas dalam JSON) disimpan dalam variabel reaktif Vue (data()). Perubahan pada data ini secara otomatis memicu pembaruan tampilan (DOM), seperti saat pengguna menandai tugas selesai, status dan statistik pada UI akan diperbarui secara *real-time* tanpa perlu *refresh* halaman.

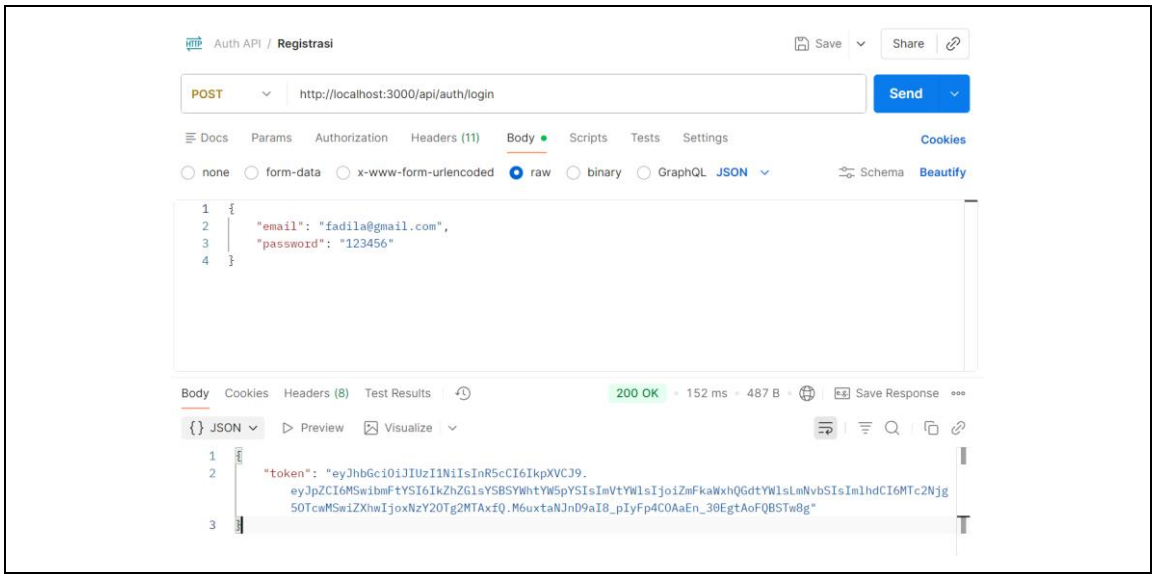
BAB IV - PENGUJIAN

1. Pengujian API Menggunakan Postman



Gambar 4. Tampilan *Register User* pada Postman

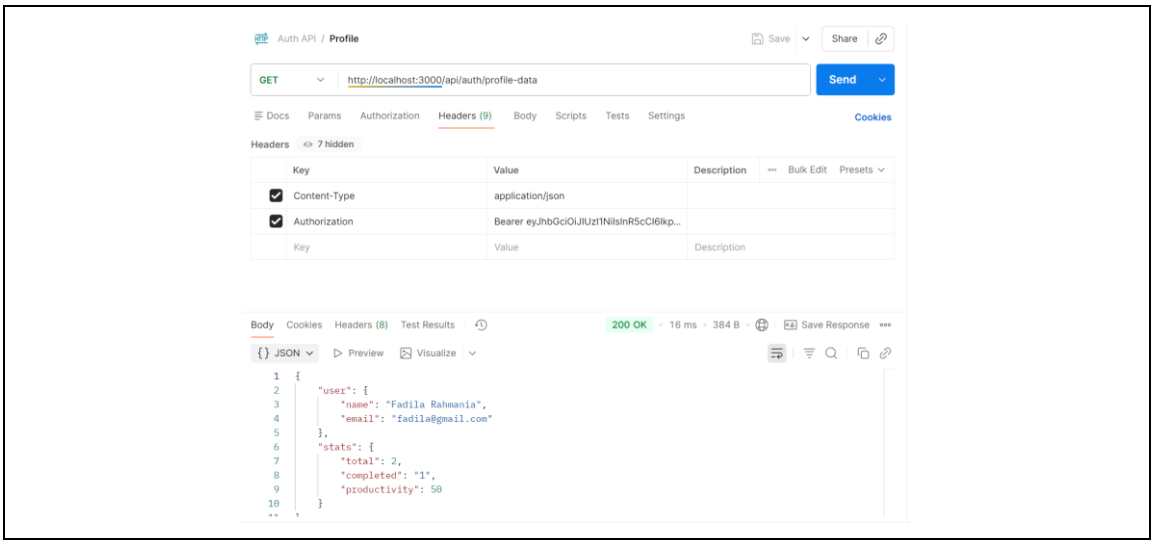
Pengujian endpoint **POST /api/auth/register** melalui Postman berhasil dilakukan dengan mengirim data *username* "Fadila Rahmania" dan email terkait dalam format JSON ke localhost:3000. Server memberikan *respons* sukses berstatus 200 OK beserta pesan "Registrasi berhasil", yang menunjukkan proses pendaftaran *user* baru telah tersimpan dengan baik di *database*. Hal ini mengonfirmasi bahwa sistem autentikasi pada aplikasi Express berjalan optimal.



Gambar 5. Tampilan *Login User* pada Postman

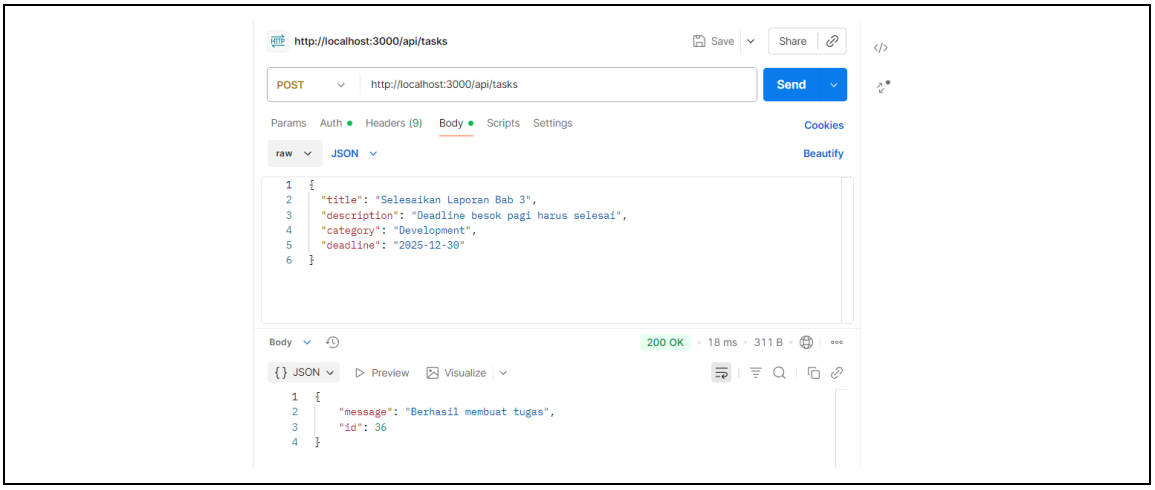
Gambar di menunjukkan hasil pengujian endpoint **POST /api/auth/login** menggunakan Postman untuk proses autentikasi *user*. Pada pengujian ini, sistem menerima data email dan *password* yang telah terdaftar sebelumnya. Jika data valid, server mengembalikan respons dengan status code 200 OK dan menampilkan token JWT (JSON

Web Token) pada atribut token. Token ini digunakan sebagai bukti autentikasi yang sah dan wajib disertakan pada header Authorization (dengan format Bearer Token) untuk mengakses endpoint yang dilindungi. Hasil ini membuktikan bahwa proses login berjalan dengan baik dan mekanisme pembuatan token JWT berfungsi sesuai harapan.



Gambar 6. Tampilan *Profile User* pada Postman

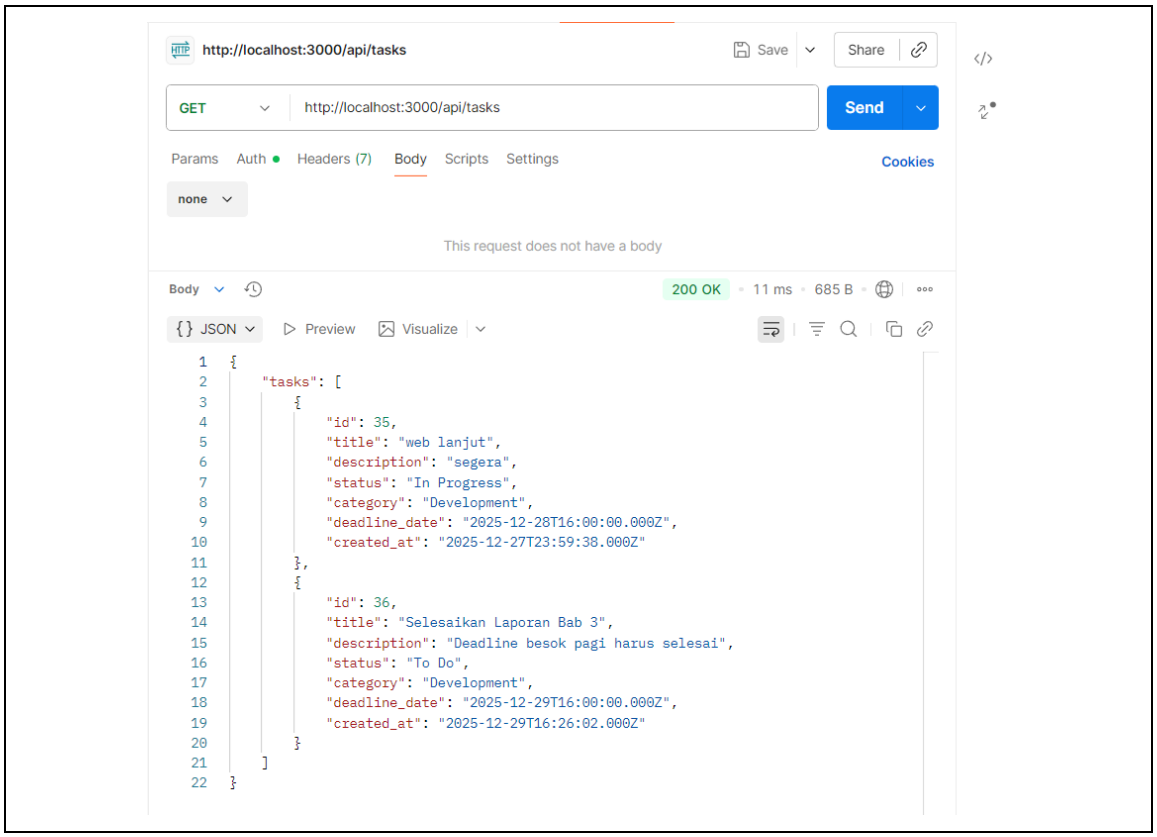
Gambar di atas menampilkan hasil pengujian endpoint **GET /api/auth/profile-data** menggunakan Postman untuk mengakses route yang dilindungi (*protected route*). Pada pengujian ini, token JWT yang sebelumnya diperoleh dari proses *login* dikirim melalui *header Authorization* dengan format Bearer Token. Hasil respons menampilkan data user yang berhasil di-decode dari token (berisi *user* dan *stats*). Respons dengan status code 200 OK ini menandakan bahwa middleware autentikasi JWT berfungsi dengan benar, di mana hanya *user* yang memiliki token valid yang dapat mengakses endpoint *profile*.



Gambar 7. Tampilan Tambah Tugas pada Postman

Gambar di atas menampilkan hasil pengujian *endpoint* **POST /api/tasks** menggunakan Postman untuk menambahkan data tugas baru ke dalam sistem. Pada

pengujian ini, data tugas dikirim melalui **Body** permintaan dalam format **JSON** yang mencakup atribut *title*, *description*, *category*, dan *deadline*. Hasil *respons* menampilkan pesan sukses "Berhasil membuat tugas" beserta id tugas yang baru saja dibuat. *Respons* dengan status *code* **200 OK** ini menandakan bahwa *endpoint* pembuatan tugas berfungsi dengan benar, di mana sistem berhasil memvalidasi *input*, menyimpannya ke dalam *database*, dan memberikan umpan balik yang sesuai kepada pengguna.

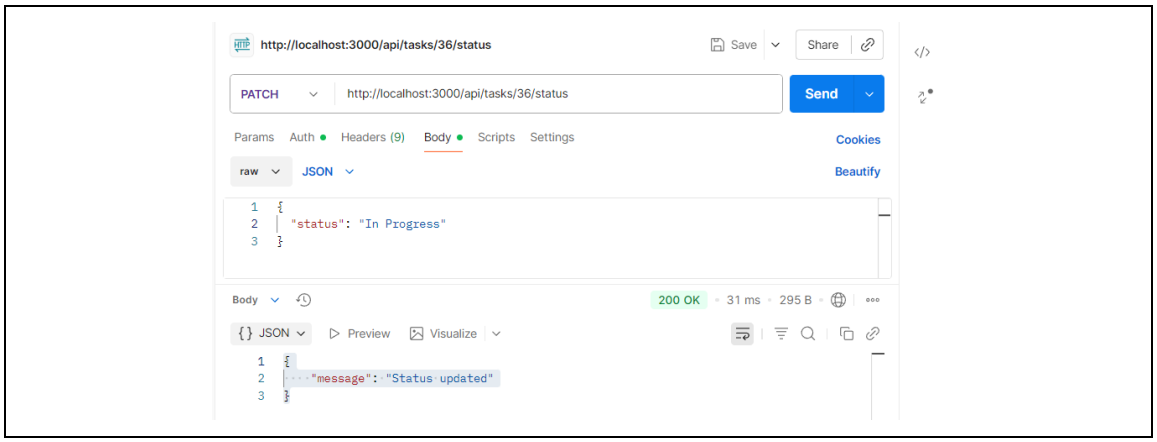


Gambar 8. Tampilan Melihat Semua Tugas pada Postman

Gambar di atas menampilkan hasil pengujian endpoint **GET /api/tasks** menggunakan Postman untuk mengambil seluruh daftar tugas yang tersimpan di dalam sistem. Pada pengujian ini, permintaan dikirim tanpa menyertakan *body* untuk mendapatkan data dari server. Hasil *respons* menampilkan array objek dalam format **JSON** yang berisi daftar tugas lengkap dengan detail seperti id, *title*, *description*, *status*, *category*, *deadline_date*, dan *created_at*. Respons dengan status *code* **200 OK** ini menandakan bahwa *endpoint* pengambilan data tugas berfungsi dengan benar, di mana sistem berhasil mengambil data dari *database* dan menyajikannya kembali kepada pengguna dalam format yang sesuai.

Gambar 9. Tampilan *Update* Tugas pada Postman

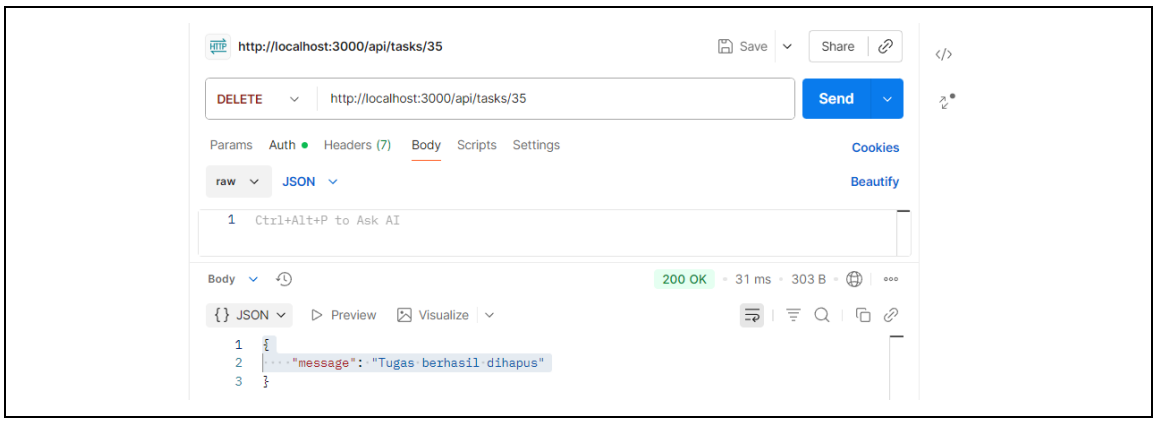
Gambar di atas menampilkan hasil pengujian *endpoint* **PUT /api/tasks/35** menggunakan Postman untuk memperbarui data tugas secara menyeluruh pada ID 35. Pada pengujian ini, data baru yang mencakup *title*, *description*, *category*, *deadline*, dan status dikirim melalui **Body** permintaan dalam format JSON. Hasil *respons* menampilkan pesan sukses "Tugas berhasil diupdate" yang mengonfirmasi bahwa seluruh informasi tugas tersebut telah diperbarui di server. *Respons* dengan status *code* **200 OK** ini menandakan bahwa *endpoint* pembaruan data berfungsi dengan benar dan sistem berhasil memproses sinkronisasi data terbaru ke dalam *database*.



Gambar 10. Tampilan *Update* Tugas pada Postman

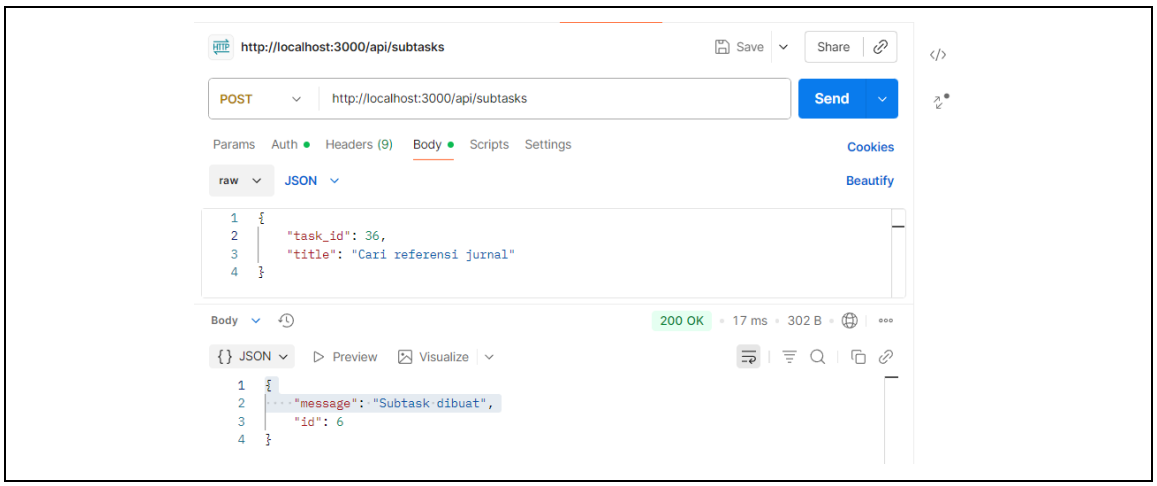
Gambar di atas menampilkan hasil pengujian *endpoint* **PATCH /api/tasks/36/status** menggunakan Postman untuk memperbarui status pada data tugas tertentu secara spesifik. Pada pengujian ini, data perubahan status dikirim melalui **Body** permintaan dalam format **JSON** dengan nilai baru yaitu "In Progress". Hasil *respons* menampilkan pesan sukses "Status updated" yang mengonfirmasi bahwa perubahan telah berhasil diterapkan pada server. *Respons* dengan status *code* **200 OK** ini menandakan bahwa *endpoint* pembaruan

status berfungsi dengan benar, di mana sistem berhasil mengidentifikasi tugas berdasarkan ID dan memperbarui kolom status sesuai instruksi yang diberikan.



Gambar 11. Tampilan *Delete* Tugas pada Postman

Gambar di atas menampilkan hasil pengujian *endpoint* **DELETE /api/tasks/35** menggunakan Postman untuk menghapus data tugas tertentu berdasarkan ID. Pada pengujian ini, permintaan dikirim langsung ke URL *endpoint* yang menyertakan parameter ID tugas yang ingin dihapus. Hasil *respons* menampilkan pesan "Tugas berhasil dihapus" yang menandakan data telah terhapus dari sistem. *Respons* dengan status *code* **200 OK** ini menandakan bahwa *endpoint* penghapusan berfungsi dengan benar, di mana sistem berhasil memproses instruksi untuk menghapus *record* dari *database* secara permanen.



Gambar 12. Tampilan *Create* Sub-Task pada Postman

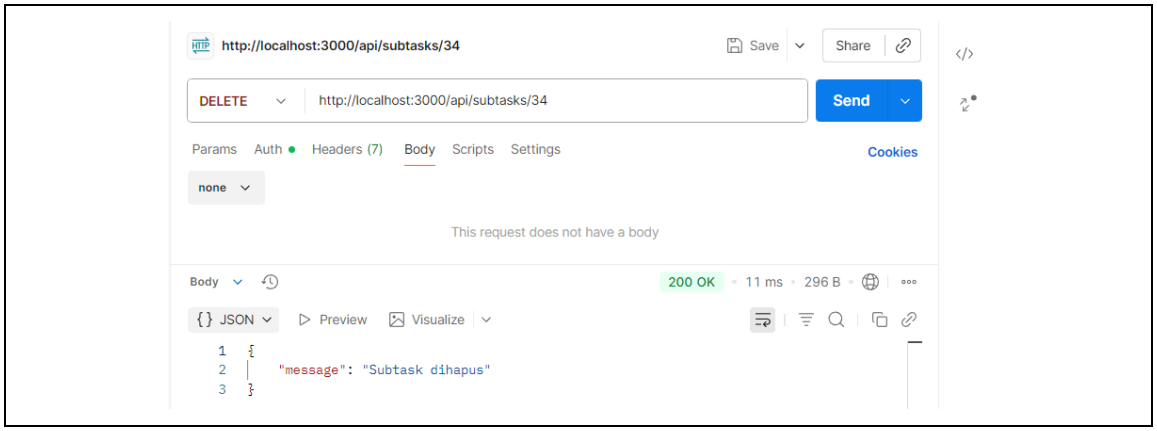
Gambar di atas menampilkan hasil pengujian *endpoint* **POST /api/subtasks** menggunakan Postman untuk membuat *sub*-tugas baru yang terelasi dengan tugas utama. Pada pengujian ini, data berupa *task_id* dan *title* dikirim melalui **Body** permintaan dalam format JSON. Hasil *respons* menampilkan pesan "Subtask dibuat" beserta id baru dari *sub*-tugas tersebut. *Respons* dengan status *code* **200 OK** ini menandakan bahwa *endpoint* pembuatan *sub*-tugas berfungsi dengan benar dan berhasil menyimpan relasi data ke sistem.

Gambar 13. Tampilan *Update* Sub-Task pada Postman

Gambar di atas menampilkan hasil pengujian *endpoint* **PATCH /api/subtasks/6** menggunakan Postman untuk memperbarui status penyelesaian *sub*-tugas dengan ID 6. Pada pengujian ini, data perubahan *is_completed* dengan nilai 1 (*true*) dikirim melalui **Body** permintaan dalam format JSON. Hasil *respons* menampilkan pesan "Status subtask diupdate" yang mengonfirmasi bahwa status *sub*-tugas telah berhasil diubah. *Respons* dengan status *code* **200 OK** ini menandakan bahwa *endpoint* pembaruan *sub*-tugas berfungsi dengan benar sesuai dengan ID yang dituju.

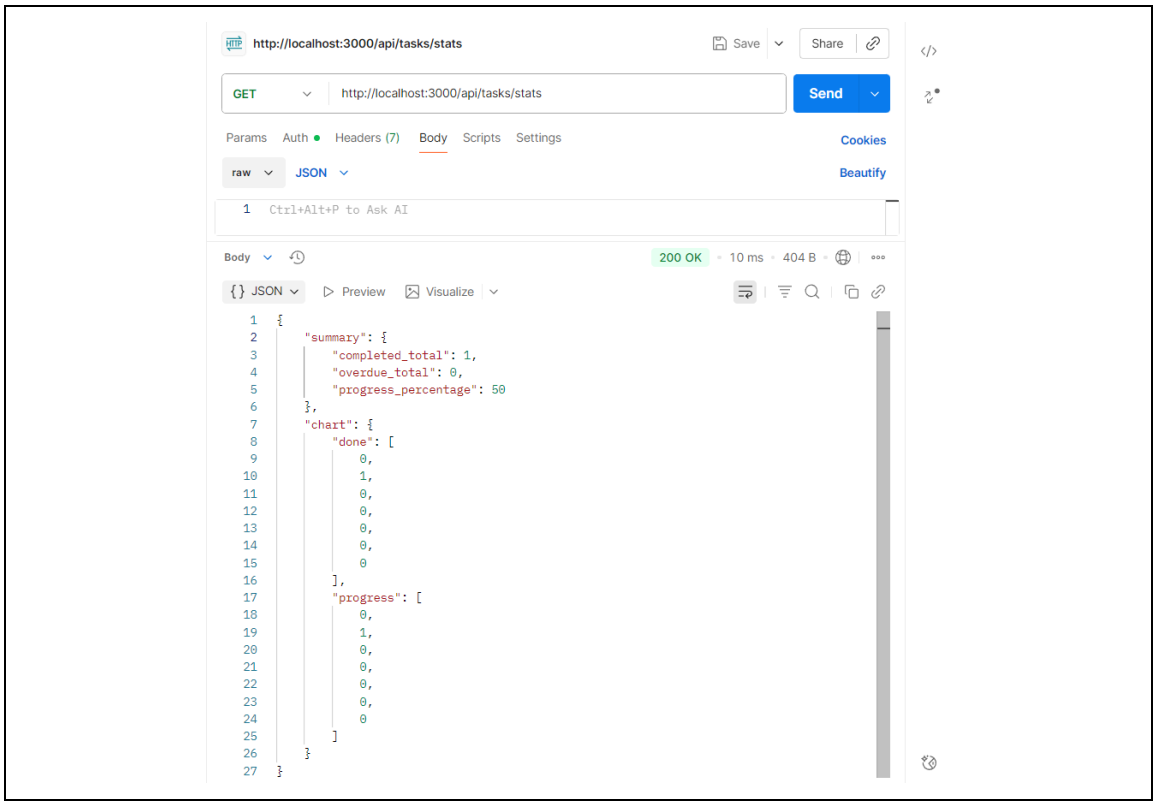
Gambar 14. Tampilan Mengambil Sub-Task Pada Task pada Postman

Gambar di atas menampilkan hasil pengujian *endpoint* **GET /api/subtasks/37** menggunakan Postman untuk mengambil daftar *sub*-tugas yang terkait dengan tugas utama (Task ID 37). Pada pengujian ini, permintaan dikirim untuk memfilter *sub*-tugas berdasarkan ID tugas induknya. Hasil *respons* menampilkan array JSON yang berisi detail *sub*-tugas seperti id, *task_id*, *title*, dan status *is_completed*. *Respons* dengan status code **200 OK** ini menandakan bahwa sistem berhasil melakukan relasi data dan menampilkan daftar *sub*-tugas yang sesuai dengan ID tugas yang diminta.



Gambar 15. Tampilan *Delete* Sub-Task pada Postman

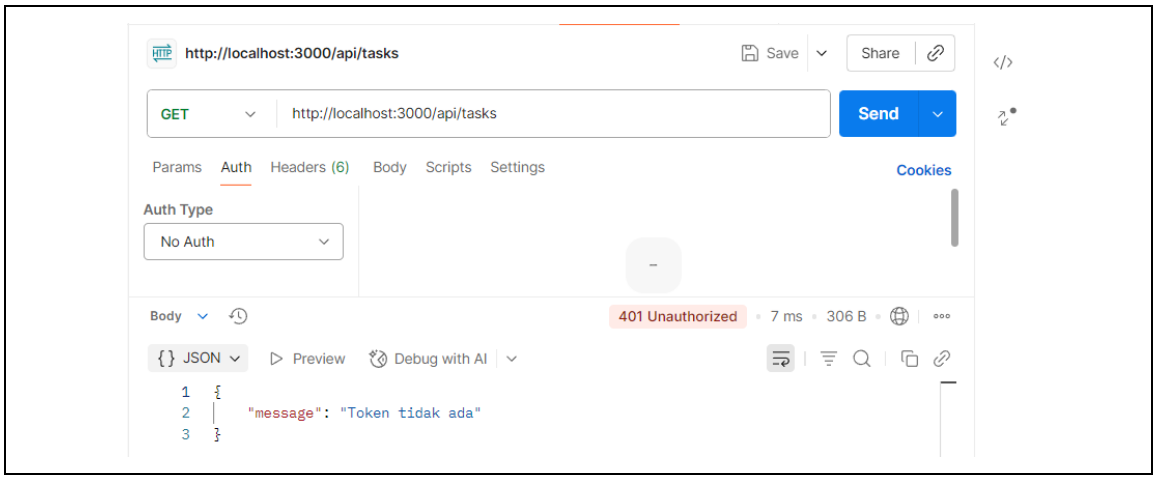
Gambar di atas menampilkan hasil pengujian *endpoint* **DELETE /api/subtasks/34** menggunakan Postman untuk menghapus *sub*-tugas tertentu berdasarkan ID-nya. Pada pengujian ini, permintaan dikirim langsung ke URL *endpoint* yang menargetkan ID *sub*-tugas 34. Hasil respons menampilkan pesan "Subtask dihapus" yang mengonfirmasi bahwa data tersebut telah dibuang dari sistem. *Respons* dengan status *code* **200 OK** ini menandakan bahwa *endpoint* penghapusan *sub*-tugas berfungsi dengan benar dan tepat sasaran.



Gambar 16. Tampilan Statistik Performa Tugas di *Postman*

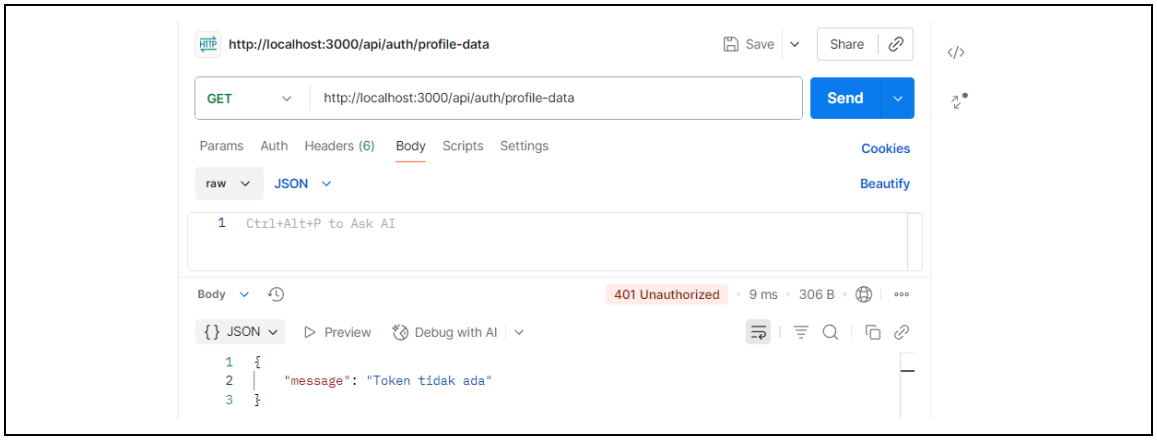
Gambar di atas menampilkan hasil pengujian *endpoint* **GET /api/tasks/stats** menggunakan Postman untuk mendapatkan ringkasan data performa tugas. Pada pengujian ini, sistem mengembalikan data berupa *summary* yang mencakup total tugas selesai, tugas

terlambat, dan persentase *progres*, serta data *chart* untuk visualisasi. *Respons* dengan status *code* **200 OK** ini menandakan bahwa logika kalkulasi statistik pada server berfungsi dengan benar dalam mengolah data tugas untuk kebutuhan laporan performa pengguna.



Gambar 17. Tampilan *Protected Route* pada *Postman*

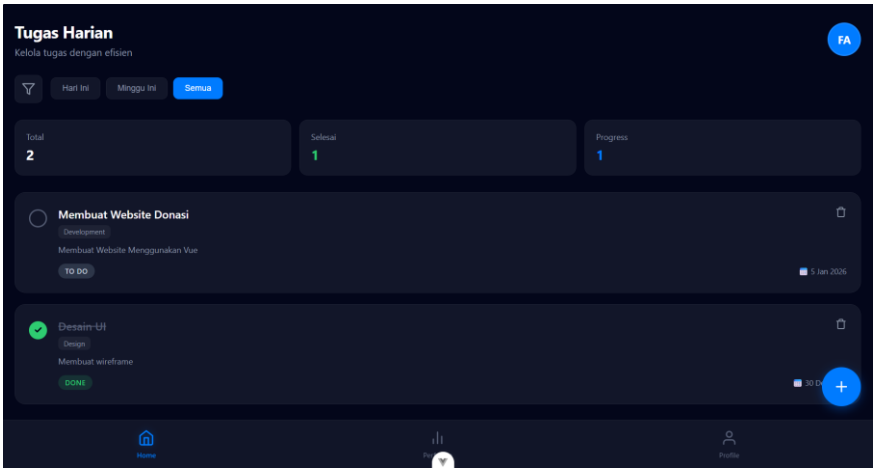
Gambar di atas menampilkan hasil pengujian *endpoint* **GET /api/auth/profile-data** menggunakan Postman untuk menguji keamanan *protected route* tanpa menyertakan token autentikasi. Hasil *respons* menampilkan pesan kesalahan "*T*oken tidak ada" dengan status code **401 Unauthorized**. Hal ini menandakan bahwa middleware keamanan berfungsi dengan benar dalam membatasi akses, di mana permintaan ditolak jika pengguna tidak menyediakan kredensial token JWT yang valid.



Gambar 18. Tampilan *Protected Route* pada *Postman*

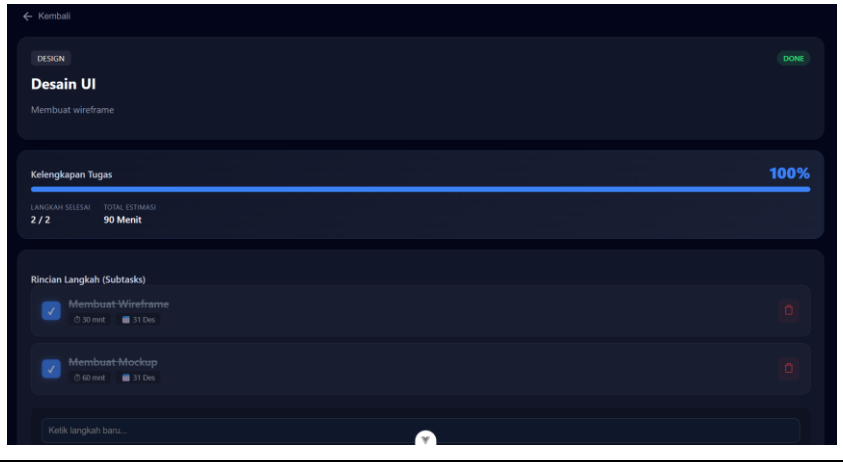
Gambar di atas menampilkan hasil pengujian *endpoint* **GET /api/auth/profile-data** menggunakan Postman untuk menguji keamanan *protected route* tanpa menyertakan token autentikasi. Hasil *respons* menampilkan pesan kesalahan "Token tidak ada" dengan status code **401 Unauthorized**. Hal ini menandakan bahwa middleware keamanan berfungsi dengan benar dalam membatasi akses, di mana permintaan ditolak jika pengguna tidak menyediakan kredensial token JWT yang valid.

2. Pengujian Aplikasi (*screenshoot* fitur utama)



Gambar 19. Tampilan Website Halaman Home

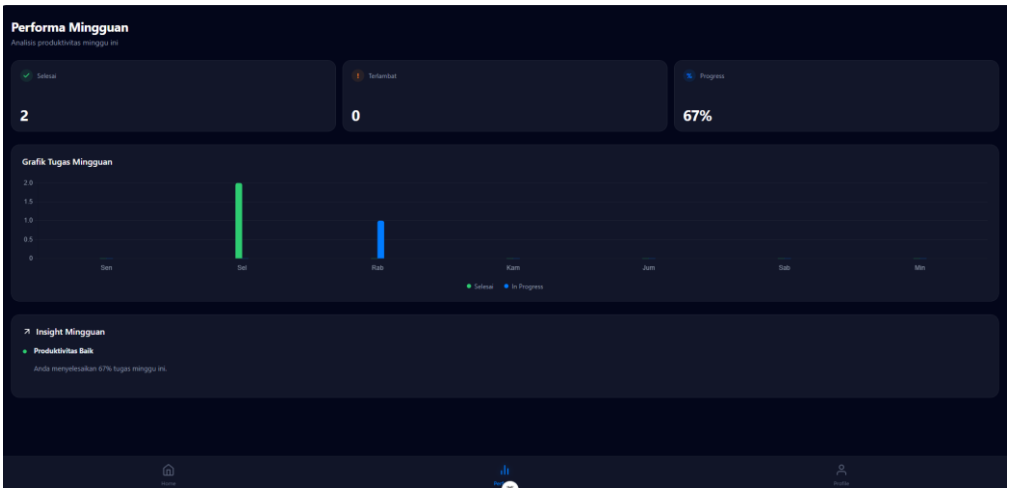
Gambar di atas menampilkan halaman dashboard sistem manajemen tugas harian yang digunakan untuk memantau dan mengelola tugas pengguna. Pada halaman ini terdapat judul dan deskripsi singkat, fitur filter tugas berdasarkan waktu, serta ringkasan statistik berupa jumlah total tugas, tugas yang telah selesai, dan tugas yang masih dalam proses. Daftar tugas ditampilkan dalam bentuk kartu yang memuat informasi judul, kategori, deskripsi singkat, status pengerjaan, dan tenggat waktu. Di bagian bawah halaman tersedia navigasi menu untuk berpindah halaman serta tombol tambah yang berfungsi untuk menambahkan tugas baru, sehingga memudahkan pengguna dalam mengelola tugas secara efisien.



Gambar 20. Tampilan Website Halaman Tugas dan Sub-Tugas

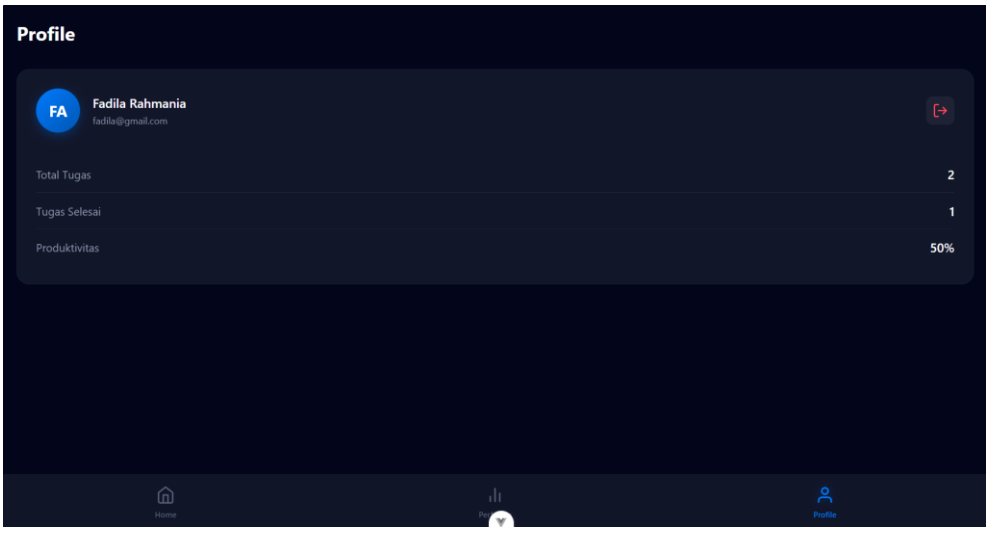
Gambar di atas menampilkan halaman detail tugas, yang berisi informasi lengkap mengenai sebuah tugas, seperti judul, kategori, deskripsi, dan status penyelesaian. Pada halaman ini juga ditampilkan tingkat kelengkapan tugas dalam bentuk progress bar, total estimasi waktu, serta daftar subtask yang dapat dicentang untuk menandai langkah-

langkah yang telah diselesaikan, sehingga pengguna dapat memantau progres pengerjaan tugas secara rinci.



Gambar 21. Tampilan Website Halaman Performa

Gambar di atas menunjukkan halaman performa mingguan yang berfungsi untuk menampilkan ringkasan produktivitas pengguna dalam satu minggu. Informasi yang disajikan meliputi jumlah tugas yang selesai, tugas yang tertunda, persentase progres, serta grafik visual yang menggambarkan aktivitas penyelesaian tugas per hari, sehingga memudahkan pengguna dalam mengevaluasi kinerja mingguan.



Gambar 22. Tampilan Website Halaman Profile

Gambar di atas menampilkan halaman profil pengguna yang berisi informasi identitas pengguna serta ringkasan statistik aktivitas, seperti total tugas, jumlah tugas yang telah diselesaikan, dan tingkat produktivitas. Halaman ini memberikan gambaran umum mengenai performa pengguna secara keseluruhan dalam menggunakan sistem manajemen tugas.

LAMPIRAN

Link GitHub Repository: https://github.com/ressaarsy/pwl25-project-akhir.git