LAPORAN UJIAN AKHIR SEMESTER

PEMROGRAMAN API

Judul : Implementasi REST API menggunakan Next.js, Prisma, dan PostgreSQL

Disusun Oleh :

Moh. Anwar Musadad	[23104410116]
Muhammad Riza Kumalasari	[23104410112]
Miftakul Huda	[23104410099]
Mahsunah Ulfah	[25104415001]

Dosen Pengampu :

Saiful Nur Budiman, S.Kom., M.Kom.

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ISLAM BALITAR

2025

KATA PENGANTAR

Puji syukur ke hadirat Allah SWT atas segala limpahan rahmat, taufik, dan hidayah- Nya sehingga kami dapat menyelesaikan laporan Ujian Akhir Semester (UAS) mata kuliah Pemrograman API dengan baik dan tepat waktu. Laporan ini disusun sebagai salah satu bentuk pertanggungjawaban akademik dalam memahami, merancang, dan mengimplementasikan layanan REST API menggunakan teknologi Next.js, Prisma ORM, dan PostgreSQL.

Penyusunan laporan ini tidak terlepas dari dukungan, bimbingan, serta arahan dari berbagai pihak. Oleh karena itu, kami menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Bapak Saiful Nur Budiman, S.Kom., M.Kom., selaku Dosen Pengampu mata kuliah Pemrograman API, yang telah memberikan pengetahuan, bimbingan, dan arahan selama proses perkuliahan dan pengerjaan tugas ini.
2. Orang tua dan keluarga yang senantiasa memberikan doa, dorongan, serta motivasi.
3. Teman-teman sekelompok dan seluruh rekan mahasiswa yang ikut berkontribusi dalam diskusi serta pelaksanaan tugas ini.

Kami menyadari bahwa laporan ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan demi penyempurnaan laporan ini di masa yang akan datang.

Semoga laporan ini dapat memberikan manfaat, khususnya dalam memahami konsep dan implementasi API modern, serta menjadi rujukan bagi pembaca yang ingin mempelajari lebih dalam mengenai pengembangan layanan web berbasis Next.js dan Prisma.

Blitar, Desember 2025

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi yang semakin pesat menuntut adanya sistem yang mampu menyediakan akses data secara cepat, aman, dan terstruktur. Salah satu teknologi yang berperan penting dalam pemenuhan kebutuhan tersebut adalah Application Programming Interface (API). API memungkinkan komunikasi antara berbagai aplikasi dan layanan sehingga mempermudah integrasi sistem, pengelolaan data, serta otomasi proses dalam dunia digital.

Dalam konteks akademik maupun kebutuhan industri, pemahaman mengenai cara membangun API yang aman, efisien, dan sesuai standar merupakan keterampilan yang sangat penting. Pengembangan API tidak hanya sekadar membuat endpoint, namun juga mencakup penerapan autentikasi, otorisasi, pengelolaan basis data, validasi data, serta penanganan error agar API yang dihasilkan dapat digunakan dengan baik dan aman oleh berbagai klien.

Mata kuliah Pemrograman API memberikan dasar dan praktik pengembangan layanan API modern. Salah satu teknologi yang kini banyak digunakan adalah Next.js dengan arsitektur App Router, yang menyediakan kemampuan server-side sekaligus fleksibilitas dalam membuat endpoint API. Kombinasi ini diperkuat dengan Prisma ORM sebagai penghubung antara aplikasi dan basis data, sehingga pengelolaan data menjadi lebih mudah, konsisten, dan aman. Sementara itu, PostgreSQL dipilih sebagai sistem manajemen basis data yang handal, stabil, serta mendukung kebutuhan transaksi data skala kecil hingga besar.

Dalam laporan ini, kami mengimplementasikan sistem REST API untuk manajemen data akademik, mencakup autentikasi menggunakan bcrypt dan JSON Web Token (JWT), middleware untuk pengamanan route, serta fitur CRUD berbasis role (User dan Admin). Implementasi ini tidak hanya bertujuan memenuhi tugas UAS, tetapi juga sebagai latihan penting dalam pengembangan API yang sesuai standar industri.

Dengan adanya proyek ini, diharapkan mahasiswa dapat memahami konsep dan praktik membangun API modern secara menyeluruh, mulai dari perancangan struktur data, pengembangan endpoint, hingga mekanisme pengamanan API. Selain itu, implementasi ini memberikan pengalaman langsung dalam mengintegrasikan berbagai tools backend seperti Next.js, Prisma, dan PostgreSQL dalam satu sistem yang utuh.

1.2 Rumusan Masalah

1. Bagaimana membangun API menggunakan Next.js?
2. Bagaimana mengimplementasikan autentikasi dengan JWT dan bcrypt?
3. Bagaimana mengatur role Admin & User?
4. Bagaimana mengelola data Student menggunakan CRUD?

1.3 Tujuan

1. Mengimplementasikan API modern dengan Next.js App Router.
2. Memanfaatkan Prisma untuk ORM dan migrasi database.
3. Menggunakan PostgreSQL sebagai basis data.
4. Membuat sistem autentikasi dan otorisasi berbasis JWT.
5. Melakukan CRUD Students sebagai studi kasus.

1.4 Batasan Masalah

Agar pengembangan sistem dan pembahasan dalam laporan ini tetap terarah, fokus, dan sesuai dengan tujuan pembelajaran, maka penulis menetapkan beberapa batasan masalah sebagai berikut:

1. Sistem yang dibangun hanya berupa layanan REST API, tanpa menyediakan antarmuka pengguna (frontend). Pengujian dilakukan menggunakan Postman atau tool sejenis.
2. Entitas yang dikelola dibatasi pada dua jenis data, yaitu:
    - Data User (untuk autentikasi dan pengaturan role),
    - Data Student sebagai objek CRUD utama.
3. Autentikasi menggunakan JSON Web Token (JWT) hanya mencakup:
    - Register (mendaftarkan akun baru),
    - Login (menghasilkan token akses),
    - Penggunaan token untuk mengakses route yang dilindungi.

    Sistem tidak mencakup mekanisme refresh token, token revocation, maupun manajemen sesi lanjutan.
4. Otorisasi dibatasi pada dua role, yaitu:
    - User, yang dapat melakukan operasi GET, POST, dan PATCH terhadap data Student,
    - Admin, yang memiliki akses penuh termasuk hak untuk menghapus data (DELETE).

    Tidak ada implementasi level role tambahan selain kedua peran tersebut.

5. Validasi input dilakukan sesuai kebutuhan tugas, yaitu menggunakan skema validator berbasis Zod untuk data Student. Validasi lanjutan seperti sanitasi mendalam atau pengecekan kompleks tidak termasuk dalam ruang lingkup.
6. Database yang digunakan hanya PostgreSQL, dan sistem tidak mencakup integrasi dengan basis data lain ataupun mekanisme replikasi, backup, dan optimasi performa tingkat lanjut.
7. Sistem tidak mencakup fitur produksi, seperti rate limiting, logging terpusat, monitoring, maupun deployment ke layanan cloud. Fokus proyek adalah implementasi konsep dasar API sesuai ketentuan UAS.

Dengan batasan-batasan di atas, pengembangan API dalam laporan ini difokuskan pada pemahaman mekanisme autentikasi, otorisasi, desain endpoint, serta manajemen data menggunakan Prisma ORM dan Next.js sebagai server backend.

BAB II
PERANCANGAN SISTEM

2.1 Arsitektur Sistem

Sistem yang dikembangkan pada proyek ini merupakan aplikasi backend berbasis REST API yang dirancang menggunakan pendekatan modular dan terstruktur. Arsitektur yang digunakan memadukan beberapa komponen utama, yaitu Next.js App Router, Prisma ORM, PostgreSQL, middleware untuk keamanan, serta penerapan JWT Authentication dengan Role- Based Authorization (Admin dan User).

Secara umum, arsitektur sistem memisahkan tiga lapisan utama, yaitu:

1. Lapisan Routing & Middleware (Next.js App Router)

   Sistem API ini menggunakan Next.js App Router sebagai struktur routing utama. App Router memungkinkan developer mendefinisikan endpoint API melalui file-file route.js yang berada di dalam folder app/api/....

   Keunggulan App Router:

   - Mendukung arsitektur modular (setiap folder = 1 endpoint).
   - Berjalan sebagai serverless atau local server.
   - Mendukung TypeScript, async/await, dan integrasi middleware secara langsung.
   - Struktur mudah dibaca dan dipelajari.

   Pada proyek ini, setiap endpoint seperti /api/auth, /api/students, atau /api/students/[id] memiliki file route.js dengan fungsi GET, POST, PATCH, DELETE, dan sebagainya sesuai operasi CRUD.

2. Lapisan Logika bisnis dan ORM (Prisma)

   Prisma digunakan sebagai Object Relational Mapping (ORM) yang berfungsi menjembatani logika aplikasi dengan database PostgreSQL.

   Prisma mengatur hal-hal berikut:

   - Definisi skema data melalui schema.prisma
   - Proses migrasi database
   - Query data menggunakan Prisma Client
   - Validasi tipe secara otomatis

   Keunggulan Prisma untuk tugas ini:

   - Query lebih mudah dipahami dibanding SQL mentah
   - Kesalahan tipe data lebih mudah dicegah

- Integrasi langsung dengan Next.js

Pada proyek ini, Prisma mengelola dua model utama:

1. User — menyimpan akun, password hash, dan role
2. Student — menyimpan data mahasiswa untuk proses CRUD

3. PostgreSQL sebagai Database

   PostgreSQL dipilih karena stabil, aman, dan sering digunakan dalam aplikasi produksi. Database menyimpan seluruh data User dan Student yang kemudian diakses oleh Prisma. Pemilihan PostgreSQL juga sesuai dengan standar tugas UAS yang mengharuskan penggunaan sistem database yang kuat dan terstruktur.

4. Middleware untuk Keamanan

   Next.js menyediakan mekanisme *middleware* untuk memproses request sebelum mencapai endpoint API.

   Dalam proyek ini, middleware digunakan untuk:

   - Mengecek apakah request mengandung header *Authorization: Bearer <token>*
   - Memverifikasi JWT menggunakan library jose
   - Mencegah akses endpoint sensitif oleh user tanpa hak
   - Menyisipkan informasi user (id, email, role) ke dalam header request lanjutan

   Middleware juga berfungsi sebagai *gatekeeper*, memastikan hanya request yang valid yang bisa mencapai server.

5. JWT Authentication

   Otentikasi menggunakan JSON Web Token (JWT) diterapkan agar API tetap stateless dan aman.

   Alur kerjanya:

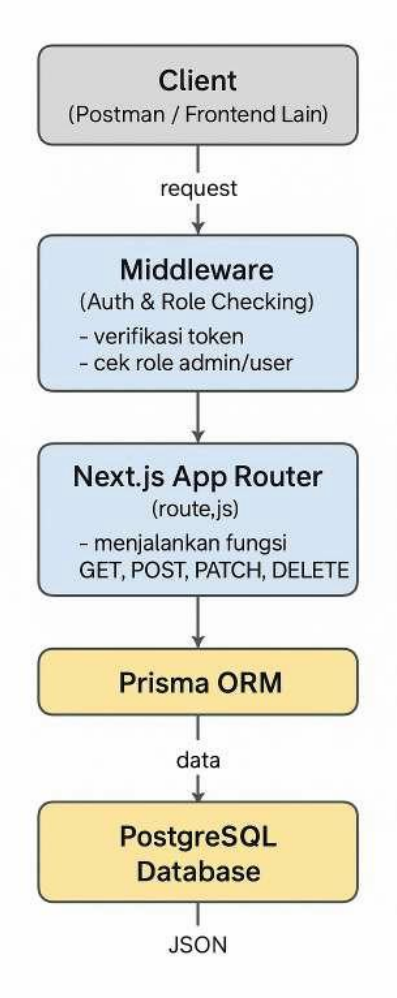
   1. User melakukan login
   2. Server menghasilkan token berisi *id, email,* dan *role*
   3. Token dikirimkan ke klien
   4. Klien memasukkan token pada setiap request yang membutuhkan akses
   5. Middleware memverifikasi token sebelum request diproses

   Kelebihan JWT:

   - Tidak membutuhkan session di server
   - Mudah diintegrasikan pada API

- Format ringkas dan aman jika kunci rahasia

dijaga Diagram Arsitektur Sistem



Penjelasan:

Diagram arsitektur secara umum menggambarkan:

1. Client (Postman / Frontend Lain)
   - mengirim request
2. Middleware (Auth & Role Checking)
   - ➢ verifikasi token
   - ➢ cek role admin/user

   ↓
3. Next.js App Router (route.js)
   - menjalankan fungsi GET, POST, PATCH, DELETE
4. Prisma ORM
   - memproses query ke database

   ↓
5. PostgreSQL Database
   - menyimpan dan mengembalikan data

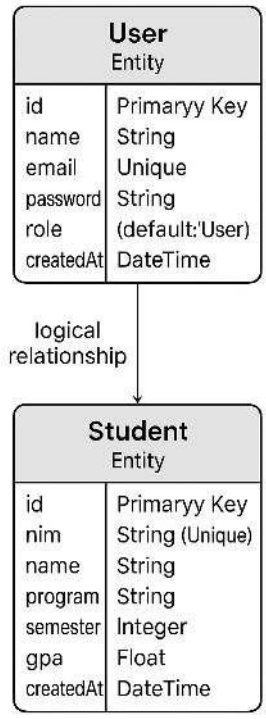
Kemudian data mengalir kembali ke client dengan format JSON.

2.2 ERD Database

Entity Relationship Diagram (ERD) merupakan representasi visual dari struktur database yang digunakan pada sistem. ERD membantu menggambarkan entitas apa saja yang terlibat, atribut yang dimiliki, serta hubungan antar entitas tersebut. Pada proyek ini, terdapat dua tabel utama yang menjadi dasar sistem, yaitu User dan Student.

Secara umum, kedua tabel ini dirancang untuk memenuhi kebutuhan autentikasi, otorisasi, dan pengelolaan data mahasiswa. Relasi antar entitas bersifat independen (tidak saling terhubung secara langsung), karena sistem autentikasi berdiri sendiri dan tidak memerlukan keterhubungan langsung terhadap data mahasiswa.

Gambaran ERD



Penjelasan Entitas :

1. Entitas User

   Tabel User digunakan untuk menyimpan data akun yang dapat mengakses sistem.

   Tabel ini mencakup informasi identitas, kredensial, dan hak akses (role).

   Atribut-atribut dalam entitas ini meliputi:

Nama Atribut	Tipe Data	Keterangan
id	Integer (Primary Key)	Nomor unik tiap user
name	String	Nama lengkap user
email	String (Unique)	Username untuk login
password	String	Password yang di-*hash*
role	String (default: "User")	Hak akses: *Admin* / *User*
createAt	DateTime	Waktu pembuatan data

Fungsi tabel User:

- Menangani autentikasi (login) menggunakan JWT
- Menentukan hak akses (authorization) pada endpoint API
- Menjamin keamanan sistem melalui pembagian role

User *Admin* memiliki akses penuh terhadap API, sedangkan User biasa hanya memiliki akses terbatas

2. Entitas Student

Tabel Student merupakan tabel utama yang digunakan untuk menyimpan data mahasiswa. Tabel ini diakses dan dimodifikasi melalui berbagai endpoint CRUD. Atribut-atribut dalam entitas ini meliputi:

Nama Atribut	Tipe Data	Keteranagn
id	Integer (Primary Key)	Nomor unik tiap mahasiswa
nim	String (Unique)	Nomr Induk Mahasiswa
name	String	Nama mahasiswa
program	String	Program jurusan
semester	Integer	Semester aktif
gpa	Float	Indeks Prestasi Kumulatif (IPK)
createdAt	DateTime	Waktu pembuatan data

Fungsi tabel Student:

- Menyimpan data mahasiswa secara terstruktur
- Menjadi objek utama operasi CRUD pada API
- Menjadi bagian dari studi kasus untuk memenuhi kebutuhan tugas UAS

Hubungan Antar Entitas

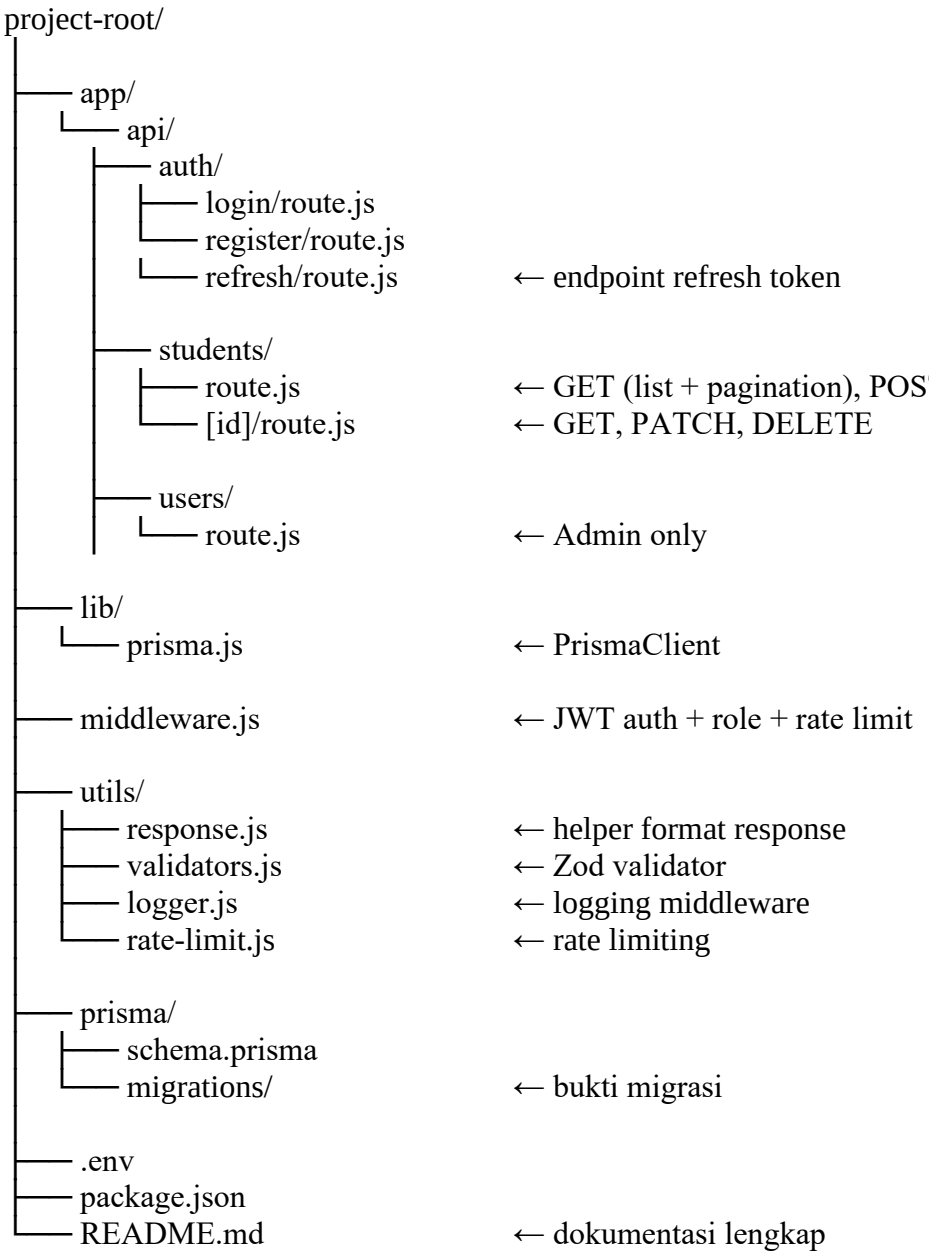
Sistem ini tidak menggunakan foreign key antar tabel, karena konteks tugas tidak mengharuskan hubungan langsung antara User dan Student. User berfungsi sebagai pengakses API, sedangkan Student adalah objek data yang dikelola.

Namun, hubungan fungsionalnya dapat dijelaskan sebagai berikut:

- User bertindak sebagai aktor, bukan sebagai pemilik data Student.
- Admin memiliki izin untuk melakukan seluruh operasi CRUD pada tabel Student.
- User biasa hanya dapat melakukan operasi tertentu (misalnya GET & POST).

Dengan demikian, hubungan antar entitas bersifat logical relationship, bukan physical relationship, sesuai kebutuhan sistem dan spesifikasi UAS.

2.3 File & Folder Structure (Next.js)



2.4 Struktur Project dan Penjelasan Setiap File

Untuk memastikan kemudahan pengembangan, pemeliharaan, serta keterbacaan kode, proyek REST API ini dirancang menggunakan struktur folder yang terorganisir sesuai standar Next.js App Router. Setiap folder dan file memiliki peran dan tanggung jawab yang jelas, sehingga mendukung penerapan prinsip modular dan separation of concern. Berikut merupakan struktur project secara lengkap beserta penjelasan fungsi dari masing-masing file utama.

app/api/auth/login/route.js

- Menerima email & password
- Verifikasi password (bcrypt)
- Generate access token + refresh token
- Simpan refresh token di database (opsional)

app/api/auth/register/route.js

- Registrasi user baru
- Hash password
- Role default: “User”

app/api/auth/refresh/route.js (BARU)

- Menerima refresh token
- Verifikasi valid
- Generate access token baru tanpa login ulang

app/api/students/route.js

- GET → menampilkan semua student + pagination
- POST → menambah student (User & Admin)

app/api/students/[id]/route.js

- GET → detail student
- PATCH → update student
- DELETE → hanya Admin

middleware.js

- Memverifikasi JWT
- Memvalidasi role
- Mengecualikan route /auth/*
- Menjalankan logging middleware
- Menjalankan rate limiting

utils/logger.js

Middleware logging:

- Method
- URL
- Waktu eksekusi
- User ID (jika tersedia)

utils/rate-limit.js

Rate limiting sederhana per IP:

- Contoh: max 60 request / menit
- Response 429 jika limit dilampaui

utils/validators.js

- Validator Zod untuk Student dan User.

utils/response.js

- Helper untuk format response JSON.

lib/prisma.js

- Inisialisasi PrismaClient & koneksi database.

prisma/schema.prisma

- Model User + Student.

BAB III

IMPLEMENTASI API

3.1 Instalasi dan Setup

Lingkungan Tools yang

digunakan:

- Node.js
- PostgreSQL
- VSCode
- Prisma CLI
- Postman

Langkah

Instalasi:

- Clone repository dari GitHub
- Install dependency (npm install)
- Buat file .env (modifikasi file .env.example)
- Setup database dan migrasi Prisma (npx prisma migrate dev)
- Jalankan server (npm run dev)

3.2 Implementasi Database dengan

Prisma Kode schema.prisma:

```
generator client {
  provider = "prisma-client-js"
}
datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}
model User {
  id        Int      @id @default(autoincrement())
  name      String
  email     String
            @unique password
            String
  role      String   @default("User")
  createdAt DateTime @default(now())
}
model Student {
  id        Int      @id @default(autoincrement())
  nim       String   @unique
  name      String
  program   String
  semester  Int gpa
```

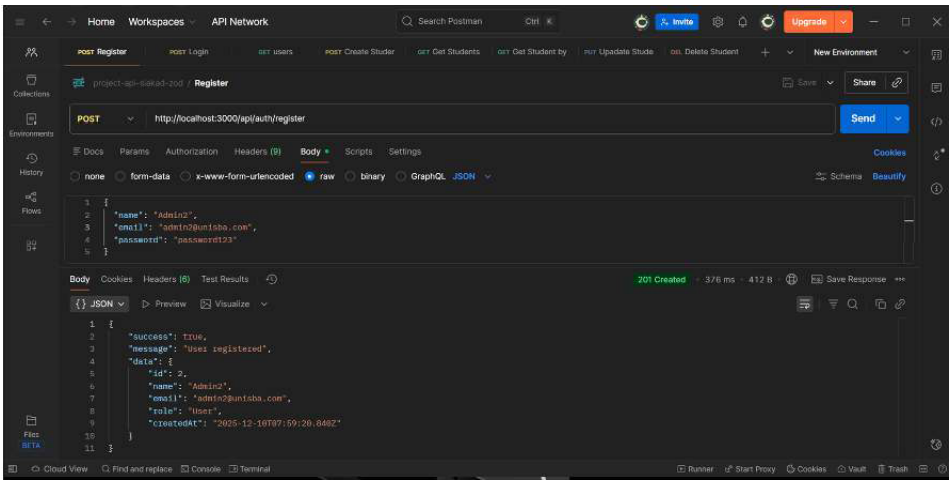
Float
createdAt DateTime @default(now())
}

3.3 Implementasi Autentikasi (Register & Login) Penjelasan:

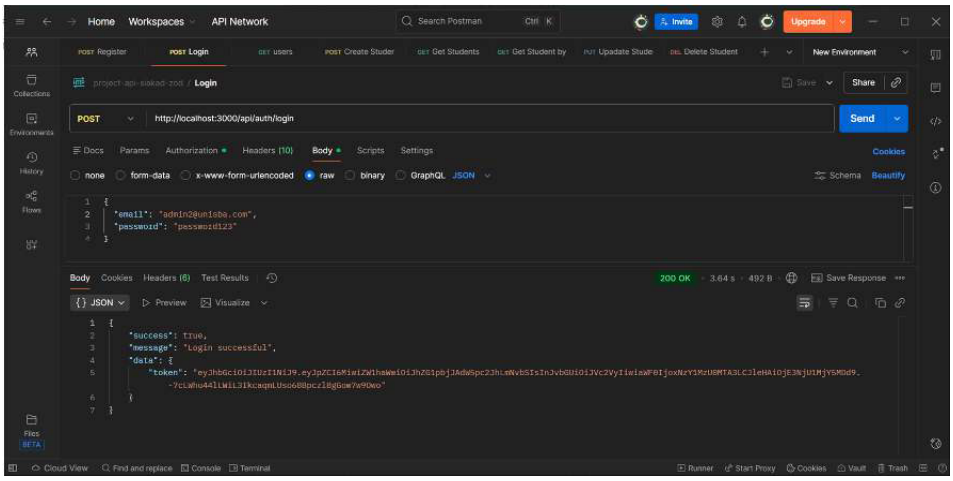
- Hash password menggunakan bcrypt
- Verifikasi login
- Generate token JWT
- Role default:

User Kode penting

- route.js register
- route.js login



Gambar Postman Register (POST | http://localhost:3000/api/auth/register)



Gambar Postman Login (POST | http://localhost:3000/api/auth/login)

3.4 Daftar dan Penjelasan Endpoint API

Setelah proses autentikasi berhasil diimplementasikan, langkah selanjutnya adalah mendefinisikan dan menguji endpoint API yang digunakan untuk mengelola data pada sistem. Setiap endpoint dirancang mengikuti prinsip RESTful API dan diakses melalui metode HTTP yang sesuai dengan fungsinya. Tabel berikut menjelaskan daftar endpoint yang tersedia beserta hak aksesnya.

Auth

Method	Endpoint	Deskripsi
POST	/api/auth/register	Registrasi user
POST	/api/auth/login	Login & generate token
POST	/api/auth/refresh	Generate access token baru

Students

Method	Endpoint	Deskripsi
GET	/api/students	List student + pagination
POST	/api/students	Tambah student
GET	/api/students/:id	Detail student
PATCH	/api/students/:id	Edit student
DELETE	/api/students/:id	Hanya Admin

User (Admin only)

Method	Endpoint
GET	/api/users

3.5 Cara Kerja Middleware Autentikasi dan Otorisasi

Middleware berperan sebagai lapisan pengaman yang dieksekusi sebelum request diproses oleh endpoint API. Pada sistem ini, middleware digunakan untuk memverifikasi JSON Web Token (JWT), memeriksa hak akses berdasarkan role pengguna, serta mencegah akses tidak sah terhadap endpoint tertentu. Mekanisme middleware ini memastikan bahwa hanya request yang valid dan memiliki izin yang dapat diteruskan ke lapisan berikutnya.

Dengan adanya middleware, sistem dapat memastikan bahwa setiap request:

1. Dikirim oleh pengguna yang telah login (autentikasi JWT).
2. Memiliki hak akses yang sesuai (otorisasi berdasarkan role).

Dalam implementasi proyek ini, terdapat beberapa aturan akses yang ditetapkan:

1. Akses Public – /api/auth/*

Seluruh endpoint pada route /api/auth bersifat publik dan tidak membutuhkan token JWT. Ini termasuk:

POST /api/auth/register

POST /api/auth/login

Alasannya karena pengguna belum memiliki token ketika pertama kali mendaftar dan login. Middleware akan mendeteksi URL yang dimulai dengan /api/auth dan langsung mengizinkannya tanpa pengecekan token.

2. Akses Login Wajib – /api/students/*

Seluruh endpoint CRUD student mengharuskan pengguna sudah login, yang berarti header wajib memiliki: **Authorization: Bearer <token>**

Jika token tidak ada atau tidak valid:

- Response akan 401

Unauthorized. Jika token valid,

middleware akan:

- Memverifikasi JWT
- Menyisipkan header:
  - x-user-id
  - x-user-email
  - x-user-role

Meneruskan request ke endpoint tujuan

Endpoint ini dapat diakses oleh User dan Admin, kecuali operasi DELETE.

3. Admin Only – /api/users/*

Seluruh endpoint yang mengakses manajemen User (misalnya GET users atau update role) hanya boleh diakses oleh role Admin.

Jika user yang login role-nya “User”, maka middleware otomatis memberikan respon:

403 Forbidden

Dengan demikian, fitur manajemen User benar-benar dijaga keamanannya.

4. Khusus Admin – DELETE Student

Walaupun route /api/students/* dapat diakses oleh semua role yang login, operasi DELETE dibatasi hanya untuk Admin.

Jadi contoh: ***DELETE /api/students/1***

Jika dilakukan oleh User biasa → 403 Forbidden,

jika dilakukan oleh Admin → 200 OK dan data berhasil dihapus.

Cara kerja Autentikasi :

1. Login
    a) User login → email + password
    b) Server memverifikasi password
    c) Server menghasilkan:
        - Access token (15 menit)
        - Refresh token (7 hari)

    Access token dipakai untuk semua request.
2. Refresh Token
    a) Jika access token expired:
        - ✓ POST /api/auth/refresh
        - ✓ Authorization: Bearer <refresh token>
    b) Server memverifikasi refresh token
    c) Server mengirim access token baru
3. Logout (opsional)
    - ✓ Refresh token dihapus dari database

A. Cara Kerja Middleware

Middleware melakukan:

1. Jika route termasuk /api/auth/* → dilewati
2. Cek header Authorization
3. Verifikasi JWT
4. Tambahkan data user ke request (id, email, role)
5. Cek role jika endpoint butuh izin khusus
6. Melewati logging middleware
7. Melewati rate limiting middleware

B. Pagination

Hasil:

- ✓ data
- ✓ totalPages
- ✓ currentPage
- ✓ nextPage
- ✓ prevPage

Pagination ini dilakukan dengan:

```
skip: (page - 1) * limit
take: limit
```

C. Logging Middleware => Log bermanfaat untuk debugging & audit.

D. Deployment ke Vercel + Neon

Database (Neon PostgreSQL)

1. Buat project di https://neon.tech
2. Copy connection string
3. Masukkan ke .env:

   DATABASE_URL="postgresql://..."

Deploy ke Vercel

1. GitHub → Import ke Vercel
2. Tambahkan environment variables
3. Jalankan:

```
npx prisma migrate deploy
npx prisma generate
```

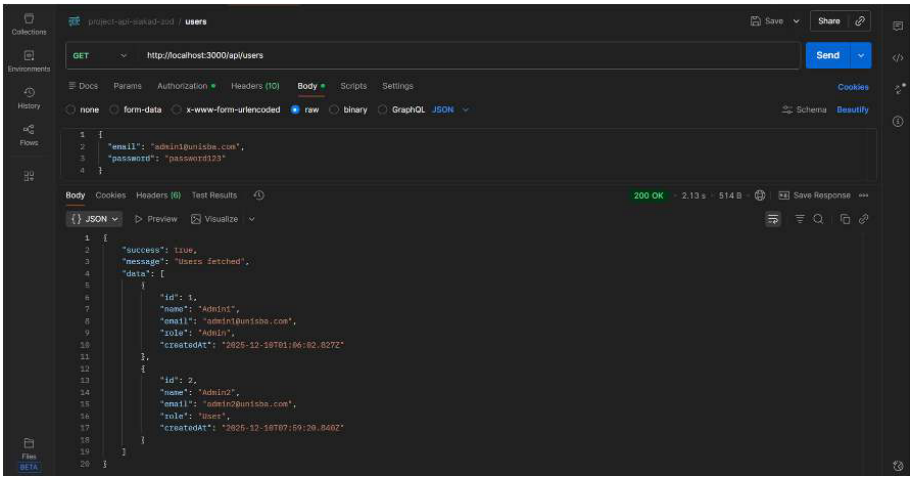
Next.js App Router otomatis menjadi serverless API.

BAB IV
PENGUJIAN DAN HASIL

4.1 Pengujian Penambahan/Pembuatan User Baru (role default: “User”)

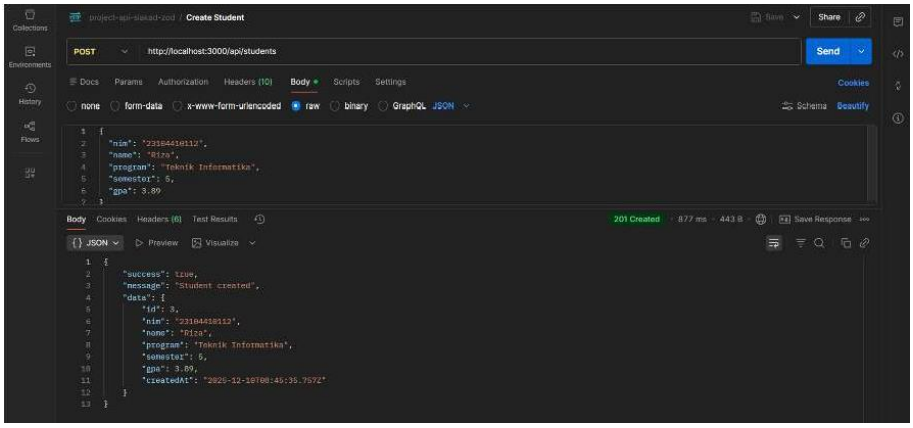
Kenapa defaultnya “User”? Yaitu agar sistem tidak memperbolehkan seseorang mendaftar sebagai Admin lewat /register, sehingga sistem tidak mudah dieksploitasi.

Selain proses login, sistem ini juga mendukung mekanisme refresh token untuk memperbarui access token yang telah kedaluwarsa tanpa mengharuskan pengguna melakukan login ulang.



Gambar Postman Users (GET | http://localhost:3000/api/users)

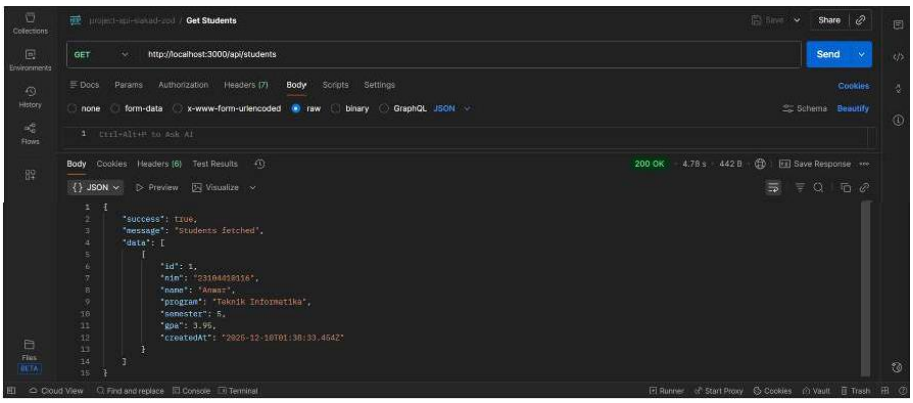
4.2 Pengujian Penambahan/Pembuatan Student



Gambar Postman Create Student (POST | http://localhost:3000/api/students)

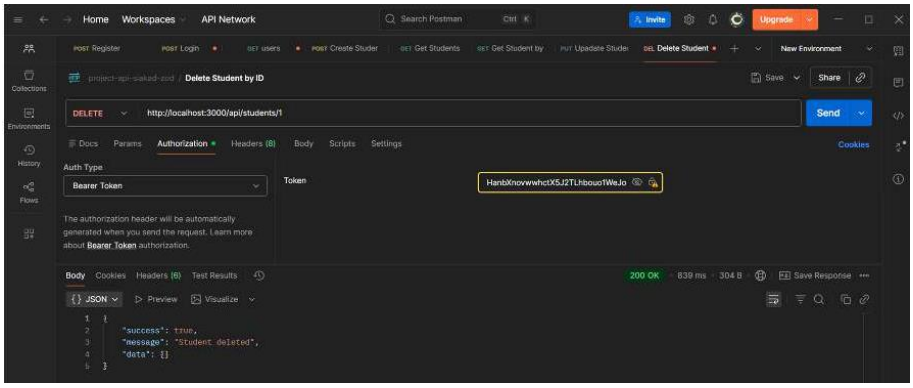
4.3 Pengujian Untuk Melihat Data Student Yang Sudah Dibuat

Untuk menghindari pengambilan data dalam jumlah besar sekaligus, endpoint GET Student dilengkapi dengan fitur pagination. Pengujian dilakukan menggunakan parameter page dan limit melalui Postman.



Gambar Postman Get Student (GET | http://localhost:3000/api/students)

4.4 Pengujian Untuk Menghapus Data Student



Gambar Postman Delete Student by ID (DELETE | http://localhost:3000/api/students)

BAB V
PENUTUP

Melalui proses perancangan, implementasi, dan pengujian yang telah dilaksanakan, dapat disimpulkan bahwa tugas UAS ini telah memberikan pengalaman praktis dalam membangun API modern yang aman, terstruktur, dan sesuai standar industri. Pemanfaatan Next.js, Prisma, dan PostgreSQL terbukti mampu mendukung pengembangan API yang stabil dan mudah dipelihara. Selain itu, penerapan autentikasi berbasis JWT dan middleware otorisasi berdasarkan role menambah nilai penting dalam pemahaman konsep keamanan sistem.

Kami berharap laporan ini dapat menjadi referensi dan pembelajaran bagi mahasiswa lain yang ingin memahami pengembangan sistem API secara lebih mendalam. Segala keterbatasan dan kekurangan yang ada dalam proyek ini dapat menjadi dasar evaluasi untuk perbaikan dan pengembangan pada implementasi berikutnya. Dengan demikian, diharapkan keterampilan dalam membangun API dan mengelola backend dapat terus berkembang dan bermanfaat ke depannya.

5.1 KESIMPULAN

Berdasarkan proses perancangan, implementasi, dan pengujian yang telah dilakukan, dapat disimpulkan bahwa sistem API berhasil dibangun menggunakan teknologi Next.js (App Router) sebagai server backend, Prisma ORM sebagai pengelola basis data, dan PostgreSQL sebagai sistem manajemen database. Integrasi ketiga komponen ini mampu menghasilkan API yang terstruktur, mudah dikembangkan, serta memiliki kinerja yang stabil dalam pengelolaan data.

Fitur autentikasi yang dikembangkan menggunakan bcrypt untuk hashing password dan JSON Web Token (JWT) untuk manajemen sesi autentikasi telah berjalan dengan baik. Mekanisme ini memastikan bahwa setiap akses API hanya dapat dilakukan oleh pengguna yang memiliki token valid, sehingga meningkatkan keamanan sistem secara keseluruhan. Proses login dan register berhasil diuji melalui Postman dan menunjukkan hasil yang sesuai dengan skenario yang diharapkan.

Selain itu, middleware yang diimplementasikan berhasil melakukan proses otorisasi berdasarkan role pengguna. Middleware mampu membedakan antara akses User dan Admin, terutama dalam endpoint yang sensitif seperti manajemen user dan proses penghapusan data mahasiswa. Dengan hal ini, sistem dapat menerapkan kontrol akses yang ketat dan mencegah terjadinya misuse atau unauthorized access pada endpoint tertentu.

Seluruh fungsionalitas CRUD Student yang mencakup akses data, penambahan, pembaruan, dan penghapusan data berhasil diuji melalui Postman. Pengujian menunjukkan bahwa setiap operasi berjalan dengan respons yang benar, termasuk ketika token tidak valid, role tidak sesuai, atau data tidak ditemukan. Dengan demikian, API yang dibangun telah memenuhi seluruh spesifikasi dan tujuan dari tugas UAS yang diberikan.

5.2 SARAN

Meskipun sistem API yang dibangun telah berfungsi sesuai kebutuhan tugas UAS, terdapat beberapa pengembangan yang dapat dilakukan di masa mendatang. Pertama, sistem dapat ditingkatkan dengan menambahkan fitur refresh token, sehingga manajemen autentikasi menjadi lebih aman dan efisien. Selain itu, integrasi rate limiting, request logging, dan API monitoring juga dapat diterapkan untuk meningkatkan keamanan serta memudahkan proses debugging.

Dari sisi arsitektur, penggunaan role dapat dikembangkan lebih fleksibel dengan penerapan role-based access control (RBAC) atau permission-based access control (PBAC) agar sistem dapat mengatur hak akses secara lebih detail. Pengembangan dokumentasi API dengan Swagger atau Postman Collection juga sangat disarankan agar mempermudah proses integrasi dan pengujian di masa depan. Pada lingkup data, entitas Student dapat diperluas dengan tabel tambahan seperti Course atau Enrollment untuk membentuk sistem akademik yang lebih lengkap.

DAFTAR PUSTAKA

Next.js Documentation. (2025). Next.js 14 App Router Documentation. Vercel. Diakses dari: https://nextjs.org/docs

Prisma Documentation. (2025). Prisma ORM: Database Toolkit & ORM for Node.js. Prisma Data Inc. Diakses dari: https://www.prisma.io/docs

PostgreSQL Global Development Group. (2025). PostgreSQL Official Documentation.
Diakses dari: https://www.postgresql.org/docs

bcrypt Documentation. (2025). bcrypt NPM Package. OpenJS Foundation. Diakses dari: https://www.npmjs.com/package/bcrypt

JSON Web Token (JWT). (2025). JWT.io: JSON Web Tokens Introduction. Auth0. Diakses dari: https://jwt.io/introduction

Zod Documentation. (2025). Zod: TypeScript-first schema validation library. Diakses dari: https://zod.dev/

Mozilla Developer Network (MDN). (2025). HTTP Authentication & Authorization. Mozilla Foundation. Diakses dari: https://developer.mozilla.org/

OpenAPI Initiative. (2024). REST API Design Principles. Linux Foundation. Diakses dari: https://www.openapis.org/