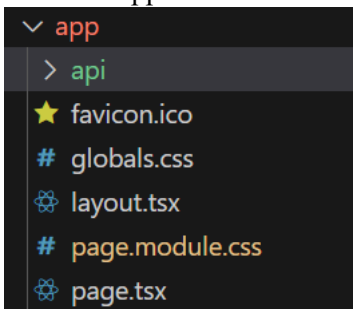
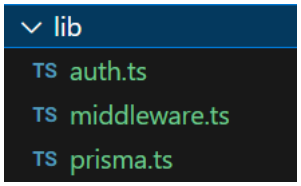
Nama : Indri Tri Maharani
NPM : 22312035
Kelas : IF 22 B

1. Membuat folder app dan lib pada folder src

- Isi folder app



- Isi folder lib

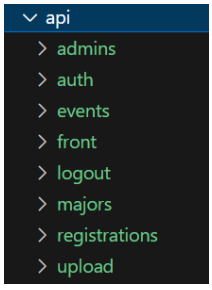


2. Melakukan instal jsonwebtoken

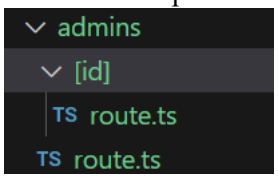
```
PS D:\KULIAHKU\SEMESTER 6\PEMROGRAMAN BERORIENTASI SERVIS\TUGAS AKHIR\TOOS-Acarain\acarain-backend> npm install jsonwebtoken

added 26 packages, removed 1 package, changed 16 packages, and audited 69 packages in 35s

8 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
npm notice
npm notice New minor version of npm available! 11.2.0 -> 11.4.1
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.4.1
npm notice To update run: npm install -g npm@11.4.1
npm notice
PS D:\KULIAHKU\SEMESTER 6\PEMROGRAMAN BERORIENTASI SERVIS\TUGAS AKHIR\TOOS-Acarain\acarain-backend>
PS D:\KULIAHKU\SEMESTER 6\PEMROGRAMAN BERORIENTASI SERVIS\TUGAS AKHIR\TOOS-Acarain\acarain-backend> npm run dev
```

jsonwebtoken (sering disebut JWT) adalah sebuah library JavaScript yang digunakan untuk membuat, memverifikasi, dan memproses token otentikasi berbasis JSON Web Token (JWT).

3. Membuat isi folder api pada foder app



4. Foder admins pada folder api berisi folder [id] dengan masing-masing file route.ts



- Admins>route.ts

```
import { NextResponse, NextRequest } from "next/server"
import prisma from "@/lib/prisma"
import { requireAdmin } from "@/lib/middleware"
import bcrypt from "bcryptjs"
```

Kode diatas digunakan untuk mengimpor modul yang dibutuhkan dalam membangun API route di Next.js.

- Modul NextResponse dan NextRequest digunakan untuk menangani permintaan dan respons HTTP.
- prisma digunakan untuk mengakses database, sedangkan requireAdmin adalah middleware yang memverifikasi apakah pengguna adalah admin.
- bcryptjs digunakan untuk mengenkripsi password sebelum disimpan ke database agar lebih aman.

```
export async function GET(req: NextRequest) {
    const { error } = requireAdmin(req)
    if (error) return error

    const admins = await prisma.admin.findMany({
        select: { id: true, name: true, email: true, createdAt: true, updatedAt: true },
        orderBy: { createdAt: "desc" }
    })
    return NextResponse.json(admins)
}
```

Digunakan untuk mengambil daftar data admin dari database dan mengembalikannya dalam format JSON, hanya jika pengguna adalah admin.

```
export async function POST(req: NextRequest) {
    const { error } = requireAdmin(req)
    if (error) return error

    try {
        const { name, email, password } = await req.json()

        const exists = await prisma.admin.findUnique({ where: { email } })
        if (exists) return NextResponse.json({ message: "Email already exists" }, { status: 400 })

        const hashed = await bcrypt.hash(password, 10)

        const admin = await prisma.admin.create({
            data: {
                name,
                email,
                password: hashed
            }
        })

        return NextResponse.json({
            id: admin.id,
            name: admin.name,
            email: admin.email,
            createdAt: admin.createdAt,
            updatedAt: admin.updatedAt
        }, { status: 201 })
    } catch (error: any) {
        return NextResponse.json({ message: error.message }, { status: 500 })
    }
}
```

Digunakan untuk menambahkan data admin baru ke dalam database, dengan validasi dan enkripsi password, dan hanya bisa diakses oleh pengguna yang sudah diverifikasi sebagai admin.

- Admins>[id]>route.ts

```ts
export async function GET(req: NextRequest, { params }: { params: { id: string } }) {
    const { error } = requireAdmin(req)
    if (error) return error

    const { id } = await params

    try {
        const admin = await prisma.admin.findUnique({
            where: { id },
            select: {
                id: true,
                name: true,
                email: true,
                createdAt: true,
                updatedAt: true
            }
        })
        return NextResponse.json(admin)
    } catch (error: any) {
        return NextResponse.json({ message: error.message }, { status: 500 })
    }
}
```

Digunakan untuk mengambil data admin berdasarkan id, dengan proteksi otentikasi admin.

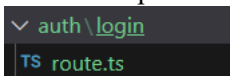
```ts
export async function PUT(req: NextRequest, { params }: { params: { id: string } }) {
    const { error } = requireAdmin(req)
    if (error) return error

    const { id } = await params
    const { name, email, password } = await req.json()
    const data: any = {}
    if (name) data.name = name
    if (email) data.email = email
    if (password) data.password = await bcrypt.hash(password, 10)

    try {
        const updated = await prisma.admin.update({ where: { id }, data })
        return NextResponse.json(updated)
    } catch (error: any) {
        return NextResponse.json({ message: error.message }, { status: 500 })
    }
}
```

Digunakan untuk mengupdate data admin berdasarkan id, dengan autentikasi admin.

```ts
Windsurf: Refactor | Explain | Generate JSDoc | X
export async function DELETE(req: NextRequest, { params }: { params: { id: string } }) {
    const { error } = requireAdmin(req)
    if (error) return error

    const { id } = await params

    try {
        await prisma.admin.delete({ where: { id } })
        return NextResponse.json({ message: 'Deleted successfully' })
    } catch (error: any) {
        return NextResponse.json({ error: error.message }, { status: 500 })
    }
}
```

Digunakan untuk menghapus data admin dari database menggunakan Prisma ORM di framework Next.js.

5. Foder auth pada folder api berisi folder login dengan file route.ts

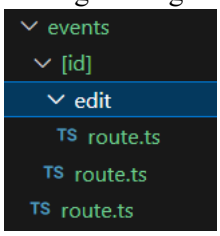


Auth>login>route.ts

```
Windsurf: Refactor | Explain | Generate JSDoc | ×
export async function POST(request: NextRequest) {
    const { email, password } = await request.json()
    const admin = await prisma.admin.findUnique({ where: { email } })

    if (!admin) return NextResponse.json({ message: "Invalid credentials" }, { status: 401 })

    const isMatch = await bcrypt.compare(password, admin.password)

    if (!isMatch) return NextResponse.json({ message: "Invalid credentials" }, { status: 401 })

    const token = jwt.sign({ id: admin.id }, process.env.JWT_SECRET as string, { expiresIn: "7d" })
    return NextResponse.json({ token })
}
```

Digunakan untuk proses login admin menggunakan email dan password.

6. Foder events pada folder api berisi folder [id] yang didalamnya terdapat folder edit dan disertai dengan masing-masing file route.ts



- events>route.ts

```
Windsurf: Refactor | Explain | Generate JSDoc | ×
export async function GET(req: NextRequest) {
    const { error } = requireAdmin(req)
    if (error) return error

    const events = await prisma.event.findMany({
        select: {
            id: true,
            title: true,
            location: true,
            date: true,
            quota: true,
            registrations: {
                where: { status: 'APPROVED' },
                select: { id: true }
            }
        },
        orderBy: { createdAt: "desc" },
    })

    const eventsWithQuota = events.map(event => ({
        ...event,
        currentQuota: event.quota - event.registrations.length,
        totalRegistered: event.registrations.length
    }))
    return NextResponse.json(eventsWithQuota)
}
```

Digunakan untuk mengambil daftar event (acara) dari database dan menghitung sisa kuota pendaftaran untuk masing-masing event, tetapi hanya bisa diakses oleh admin.

```ts
Windsurf: Refactor | Explain | Generate JSDoc | ×
export async function POST(req: NextRequest) {
    const { error } = requireAdmin(req)
    if (error) return error

    try {
        const {
            title,
            description,
            location,
            date,
            quota,
            thumbnail
        } = await req.json().then(data => data.form)

        const slug = slugify(title, { lower: true, strict: true })

        const exists = await prisma.event.findUnique({ where: { slug } })
        if (exists) return NextResponse.json({ message: "Slug already exists, use different title" }, { status: 400 })

        const event = await prisma.event.create({
            data: {
                title,
                slug,
                description,
                location,
                date: new Date(date),
                quota: Number(quota),
                thumbnail
            }
        })

        return NextResponse.json(event, { status: 201 })
    } catch (error: any) {
        return NextResponse.json({ message: error.message }, { status: 500 })
    }
}
```

Digunakan untuk menambahkan event baru ke database, dan hanya bisa diakses oleh admin. Fungsi ini juga melakukan validasi agar judul event tidak duplikat berdasarkan slug.

- event>[id]>route.ts

```ts
Windsurf: Refactor | Explain | Generate JSDoc | ×
export async function GET(req: NextRequest, { params }: { params: { id: string } }) {
    const { error } = requireAdmin(req)
    if (error) return error

    const { id } = await params

    try {
        const event = await prisma.event.findFirst({
            where: { id },
            include: {
                registrations: {
                    include: {
                        major: {
                            select: { name: true }
                        }
                    }
                }
            }
        })

        if (!event) return NextResponse.json({ message: 'Event not found' }, { status: 404 })

        const eventWithQuota = {
            ...event,
            currentQuota: event.quota - event.registrations.length,
            totalRegistered: event.registrations.length
        }

        return NextResponse.json(eventWithQuota)
    } catch (error: any) {
        return NextResponse.json({ message: error.message }, { status: 500 })
    }
}
```

Digunakan untuk mengambil detail event berdasarkan ID, hanya bisa diakses oleh admin, dan akan menampilkan juga informasi registrasi beserta jurusan (major) peserta, serta menghitung sisa kuota.

- event>[id]>edit>route.ts

```ts
Windsurf: Refactor | Explain | Generate JSDoc | ×
export async function GET(req: NextRequest, { params }: { params: { id: string } }) {
    const { error } = requireAdmin(req)
    if (error) return error

    const { id } = await params

    try {
        const event = await prisma.event.findFirst({
            where: { id }
        })

        return NextResponse.json(event)
    } catch (error: any) {
        return NextResponse.json({ message: error.message }, { status: 500 })
    }
}
```

Digunakan untuk mengambil detail satu event berdasarkan ID, dan hanya bisa diakses oleh admin.

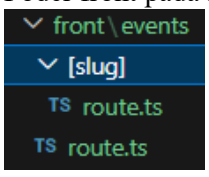
```
Windsurf: Refactor | Explain | Generate JSDoc | ×
export async function PUT(req: NextRequest, { params }: { params: { id: string } }) {
    const { error } = requireAdmin(req)
    if (error) return error

    const { id } = await params

    try {
        const {
            title,
            description,
            location,
            date,
            quota,
            thumbnail
        } = await req.json()

        const slug = slugify(title, { lower: true, strict: true })

        const exists = await prisma.event.findFirst({
            where: {
                slug,
                NOT: { id }
            }
        })
        if (exists) return NextResponse.json({ message: "Slug already exists, use different title" }, { status: 400 })
```

Digunakan untuk mengupdate data event berdasarkan ID, khusus untuk admin.

```
Windsurf: Refactor | Explain | Generate JSDoc | ×
export async function DELETE(req: NextRequest, { params }: { params: { id: string } }) {
    const { error } = requireAdmin(req)
    if (error) return error

    const { id } = await params

    try {
        const event = await prisma.event.findUnique({ where: { id } })
        if (!event) return NextResponse.json({ message: 'Event not found' }, { status: 404 })

        await prisma.event.delete({ where: { id } })

        if (event.thumbnail) {
            try {
                const filePath = path.join(process.cwd(), 'public', event.thumbnail)
                await fs.unlink(filePath)
            } catch (err) {
                console.warn('Failed to delete thumbnail', err)
            }
        }

        return NextResponse.json({ message: 'Deleted successfully' })
    } catch (error: any) {
        return NextResponse.json({ error: error.message }, { status: 500 })
    }
}
```

Digunakan untuk menghapus data event berdasarkan ID, dan juga akan menghapus thumbnail event dari sistem file jika ada. Hanya admin yang diizinkan.

7. Foder front pada folder api berisi folder events dan foder [slug] dengan masing-masing file route.ts



- front>events>route.ts

```
import { NextResponse, NextRequest } from "next/server"
import prisma from "@/lib/prisma"
```

  - NextRequest merupakan Representasi request pada Next.js middleware atau route handler.
  - NextResponse digunakan untuk membuat response HTTP (seperti json, redirect, status, dll).
  - Import prisma from “@/lib/prisma” digunakan untuk mengimpor instance Prisma Client dari file lib/prisma.ts.

```ts
export async function GET(req: NextRequest) {
    const events = await prisma.event.findMany({
        select: {
            id: true,
            title: true,
            thumbnail: true,
            date: true,
            quota: true,
            slug: true,
            registrations: {
                where: { status: 'APPROVED' },
                select: { id: true }
            }
        },
        orderBy: { createdAt: "desc" },
    })

    const eventsWithQuota = events.map(event => ({
        ...event,
        currentQuota: event.quota - event.registrations.length,
        totalRegistered: event.registrations.length
    }))
    return NextResponse.json(eventsWithQuota)
}
```

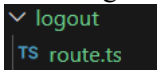
Digunakan untuk mengambil semua event yang memiliki pendaftaran (registrations) dengan status 'APPROVED' dari database menggunakan Prisma. Fungsi ini juga menghitung kuota tersisa dan jumlah peserta yang terdaftar untuk masing-masing event.

- front>events>[slug]>route.ts

```ts
export async function GET(req: NextRequest, { params }: { params: { slug: string } }) {

    const { slug } = await params

    try {
        const event = await prisma.event.findFirst({
            where: { slug },
            include: {
                registrations: {
                    where: { status: 'APPROVED' },
                    include: {
                        major: {
                            select: { name: true }
                        }
                    }
                }
            }
        })

        if (!event) return NextResponse.json({ message: 'Event not found' }, { status: 404 })

        const eventWithQuota = {
            ...event,
            currentQuota: event.quota - event.registrations.length,
            totalRegistered: event.registrations.length
        }

        return NextResponse.json(eventWithQuota)
    } catch (error: any) {
        return NextResponse.json({ message: error.message }, { status: 500 })
    }
}
```

Digunakan untuk mengambil data satu event berdasarkan slug, lengkap dengan pendaftar yang disetujui (status: 'APPROVED') dan informasi major (jurusan) mereka. Fungsi ini juga menghitung kuota tersisa (currentQuota) dan total peserta (totalRegistered).

8. Foder logout pada folder api berisi file route.ts



logout>route.ts

```ts
import { NextRequest, NextResponse } from "next/server"
import { serialize } from "cookie"
```

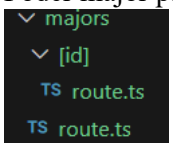
- NextRequest merupakan objek request khusus Next.js yang mirip dengan Request standar dari fetch, tapi dengan tambahan utilitas seperti akses cookie (request.cookies) dan url.

- NextResponse digunakan untuk mengembalikan respons dari handler API (mirip res.json() di Express).
- serialize dari package cookie digunakan untuk mengubah data menjadi format string cookie yang valid agar bisa ditambahkan ke header Set-Cookie.

```
Windsurf: Refactor | Explain | Generate JSDoc | ×
export async function POST(req: NextRequest) {
  const expiredToken = serialize("token", "", {
    httpOnly: true,
    path: "/",
    expires: new Date(0),
  })

  return new NextResponse(
    JSON.stringify({ message: "Logout berhasil" }),
    {
      status: 200,
      headers: {
        "Set-Cookie": expiredToken,
        "Content-Type": "application/json",
      },
    }
  )
}
```

Merupakan handler logout pada API route atau App Router (Next.js) yang melakukan proses logout pengguna dengan cara menghapus cookie token.

9. Foder major pada folder api berisi folder [id] dengan masing-masing file route.ts



- majors>route.ts

```
import { NextResponse, NextRequest } from "next/server"
import prisma from "@/lib/prisma"
import { requireAdmin } from "@/lib/middleware"
```

  - NextRequest merupakan Representasi request pada Next.js middleware atau route handler.
  - NextResponse digunakan untuk membuat response HTTP (seperti json, redirect, status, dll).
  - Import prisma from “@/lib/prisma” digunakan untuk mengimpor instance Prisma Client dari file lib/prisma.ts.
  - import { requireAdmin } from "@/lib/middleware" digunakan untuk mengimpor middleware custom bernama requireAdmin, yang kemungkinan besar untuk mengecek apakah user memiliki akses admin dan biasanya akan mengembalikan error atau memblokir akses jika bukan admin.

```
Windsurf: Refactor | Explain | Generate JSDoc | ×
export async function GET(req: NextRequest) {
    const majors = await prisma.major.findMany({ orderBy: { createdAt: "desc" } })
    return NextResponse.json(majors)
}
```

  - GET handler berungsi untuk menangani request HTTP GET.
  - Mengambil data dari tabel major di database menggunakan Prisma.
  - Mengurutkan data berdasarkan kolom createdAt secara menurun (descending).
  - Mengembalikan data ke klien dalam format JSON.

```
Windsurf: Refactor | Explain | Generate JSDoc | ×
export async function POST(req: NextRequest) {
    const { error } = requireAdmin(req)
    if (error) return error

    try {
        const { name } = await req.json()
        const major = await prisma.major.create({ data: { name } })
        return NextResponse.json(major, { status: 201 })
    } catch (error: any) {
        return NextResponse.json({ message: error.message }, { status: 500 })
    }
}
```

Digunakan oleh admin untuk menambahkan jurusan baru. Dapat digunakan pada halaman dashboard admin (form tambah jurusan). Otomatis akan memvalidasi bahwa user memiliki hak akses admin terlebih dahulu.

- major>[id]>route.ts

```
Windsurf: Refactor | Explain | Generate JSDoc | ×
export async function GET(req: NextRequest, { params }: { params: { id: string } }) {
    const { error } = requireAdmin(req)
    if (error) return error

    const { id } = await params

    try {
        const major = await prisma.major.findUnique({ where: { id } })
        return NextResponse.json(major)
    } catch (error: any) {
        return NextResponse.json({ message: error.message }, { status: 500 })
    }
}
```

Digunakan untuk mengambil data satu jurusan (major) berdasarkan ID. Fungsi ini juga hanya dapat diakses oleh admin.

```
Windsurf: Refactor | Explain | Generate JSDoc | ×
export async function PUT(req: NextRequest, { params }: { params: { id: string } }) {
    const { error } = requireAdmin(req)
    if (error) return error

    const { id } = await params
    const { name } = await req.json()

    try {
        const updated = await prisma.major.update({ where: { id }, data: { name } })
        return NextResponse.json(updated)
    } catch (error: any) {
        return NextResponse.json({ message: error.message }, { status: 500 })
    }
}
```

Digunakan untuk mengedit/memperbarui data jurusan (major) berdasarkan ID. Fungsi ini hanya dapat diakses oleh admin.

```
Windsurf: Refactor | Explain | Generate JSDoc | ×
export async function DELETE(req: NextRequest, { params }: { params: { id: string } }) {
    const { error } = requireAdmin(req)
    if (error) return error

    const { id } = await params

    try {
        await prisma.major.delete({ where: { id } })
        return NextResponse.json({ message: 'Deleted successfully' })
    } catch (error: any) {
        return NextResponse.json({ error: error.message }, { status: 500 })
    }
}
```

Digunakan u ntuk menghapus data jurusan (major) berdasarkan ID, dan hanya dapat diakses oleh admin.

10. Foder registrations pada folder api berisi folder [id] yang berisi folder approve, reject, kemudian ada juga folder attend dan cancel, dengan masing-masing file route.ts

- registrations>route.ts

```
import { NextResponse, NextRequest } from "next/server"
import prisma from "@/lib/prisma"
import { requireAdmin } from "@/lib/middleware"
import { nanoid } from "nanoid"
```

  - NextRequest merupakan Representasi request pada Next.js middleware atau route handler.
  - NextResponse digunakan untuk membuat response HTTP (seperti json, redirect, status, dll).
  - Import prisma from “@/lib/prisma” digunakan untuk mengimpor instance Prisma Client dari file lib/prisma.ts.
  - import { nanoid } from "nanoid" digunakan untuk mengimpor fungsi nanoid dari library nanoid, dan untuk membuat ID unik (biasanya untuk identifier yang random, seperti token, kode unik, dll).

```
Windsurf: Refactor | Explain | Generate JSDoc | X
export async function GET(req: NextRequest) {
    const { error } = requireAdmin(req)
    if (error) return error

    const registrations = await prisma.registration.findMany()
    return NextResponse.json(registrations)
}
```

Digunakan untuk mengambil seluruh data registrasi dari database dan hanya dapat diakses oleh user dengan hak akses admin

```
Windsurf: Refactor | Explain | Generate JSDoc | X
export async function POST(req: NextRequest) {
    try {
        const { name, npm, majorId, whatsapp, eventId } = await req.json()

        const cancelToken = nanoid(24)
        const qrCode = nanoid(32)

        const registration = await prisma.registration.create({
            data: {
                name,
                npm,
                majorId,
                whatsapp,
                eventId,
                cancelToken,
                qrCode
            }
        })
        return NextResponse.json({ id: registration.id }, { status: 201 })
    } catch (error: any) {
        return NextResponse.json({ message: error.message }, { status: 500 })
    }
}
```

Digunakan untuk menerima input registrasi dari pengguna, generate cancelToken dan qrCode, simpan data ke tabel registration, dan mengembalikan ID registrasi jika berhasil.

- registrations>[id]>route.ts

```
Windsurf: Refactor | Explain | Generate JSDoc | ×
export async function GET(req: NextRequest, { params }: { params: { id: string } }) {
    const { id } = await params

    try {
        const registration = await prisma.registration.findFirst({
            where: { id },
            include: {
                event: {
                    select: {
                        title: true,
                        slug: true,
                        location: true,
                        date: true
                    }
                },
                major: {
                    select: { name: true }
                }
            }
        })

        if (!registration) return NextResponse.json({ message: 'Registration not found' }, { status: 404 })

        const { qrCode, cancelToken, status, attendance, ...rest } = registration

        const response: any = {
            ...rest,
            status,
            attendance,
            event: registration.event,
            major: registration.major
        }

        if (status === 'APPROVED' && attendance === 'ABSENT') {
            response.qrCode = qrCode
        }

        if (status !== 'CANCELLED' && status !== 'REJECTED' && attendance === 'ABSENT') {
            response.cancelToken = cancelToken
        }

        return NextResponse.json(response)
    } catch (error: any) {
        return NextResponse.json({ message: error.message }, { status: 500 })
    }
}
```

Fungsi ini digunakan untuk mengambil detail lengkap registrasi berdasarkan ID, dengan fitur-fitur tambahan seperti; ambil info event dan major terkait, tampilkan qrCode jika disetujui dan belum hadir, dan tampilkan cancelToken jika belum dibatalkan atau ditolak dan belum hadir.

```
Windsurf: Refactor | Explain | Generate JSDoc | ×
export async function PUT(req: NextRequest, { params }: { params: { id: string } }) {
    const { error } = requireAdmin(req)
    if (error) return error

    const { id } = await params

    try {
        const { status, attendance } = await req.json()

        const registration = await prisma.registration.update({
            where: { id },
            data: { status, attendance },
            select: { status: true, attendance: true }
        })
        return NextResponse.json(registration)
    } catch (error: any) {
        return NextResponse.json({ message: error.message }, { status: 500 })
    }
}
```

Digunakan untuk menandai kehadiran peserta (attendance: 'PRESENT', 'ABSENT'), dan mengubah status registrasi (status: 'APPROVED', 'REJECTED', 'CANCELLED', dsb)

```
Windsurf: Refactor | Explain | Generate JSDoc | ×
export async function DELETE(req: NextRequest, { params }: { params: { id: string } }) {
    const { error } = requireAdmin(req)
    if (error) return error

    const { id } = await params

    try {
        await prisma.registration.delete({ where: { id } })
        return NextResponse.json({ message: 'Deleted successfully' })
    } catch (error: any) {
        return NextResponse.json({ error: error.message }, { status: 500 })
    }
}
```

Digunakan untuk menghapus registrasi pengguna berdasarkan ID tertentu pada dashboard admin untuk mengelola peserta/event.

- registrations>[id]>approve>route.ts

```
Windsurf: Refactor | Explain | Generate JSDoc | ×
export async function PATCH(req: NextRequest, { params }: { params: { id: string } }) {
    const { error } = requireAdmin(req)
    if (error) return error

    const { id } = await params

    try {
        const registration = await prisma.registration.update({
            where: { id },
            data: { status: 'APPROVED' }
        })
        return NextResponse.json(registration)
    } catch (error: any) {
        return NextResponse.json({ message: error.message }, { status: 500 })
    }
}
```

Digunakan dalam sistem administrasi untuk menyetujui pendaftaran peserta dalam event/kegiatan. Dan hanya bisa digunakan oleh user dengan hak akses admin (via requireAdmin).

- registrations>[id]>reject>route.ts

```
Windsurf: Refactor | Explain | Generate JSDoc | ×
export async function PATCH(req: NextRequest, { params }: { params: { id: string } }) {
    const { error } = requireAdmin(req)
    if (error) return error

    const { id } = await params

    try {
        const registration = await prisma.registration.update({
            where: { id },
            data: { status: 'REJECTED' }
        })
        return NextResponse.json(registration)
    } catch (error: any) {
        return NextResponse.json({ message: error.message }, { status: 500 })
    }
}
```

Digunakan untuk memberikan keputusan penolakan terhadap sebuah pendaftaran event (registrasi), hanya oleh admin. Biasanya bagian dari sistem approval.

- registrations>attend>route.ts

```ts
Windsurf: Refactor | Explain | Generate JSDoc | ×
export async function GET(req: NextRequest) {
    const { error } = requireAdmin(req)
    if (error) return error

    const { searchParams } = new URL(req.url)
    const code = searchParams.get('code')

    if (!code) return NextResponse.json({ message: "Code required" }, { status: 400 })

    try {
        const reg = await prisma.registration.findUnique({
            where: { qrCode: code },
            select: {
                name: true,
                npm: true,
                whatsapp: true,
                major: { select: { name: true } },
                attendance: true,
                status: true
            }
        })
        if (!reg) return NextResponse.json({ message: "Registration not found" }, { status: 404 })

        return NextResponse.json(reg)
    } catch (error: any) {
        return NextResponse.json({ message: error.message }, { status: 500 })
    }
}
```

Digunakan oleh admin untuk mengambil data registrasi peserta berdasarkan kode QR (qrCode).

```ts
Windsurf: Refactor | Explain | Generate JSDoc | ×
export async function POST(req: NextRequest) {
    const { error } = requireAdmin(req)
    if (error) return error

    const { searchParams } = new URL(req.url)
    const code = searchParams.get('code')

    if (!code) return NextResponse.json({ message: "Code required" }, { status: 400 })

    try {
        const reg = await prisma.registration.findUnique({ where: { qrCode: code } })
        if (!reg || reg.status === 'REJECTED' || reg.status === 'CANCELLED') return NextResponse.json(
            { message: "Registration has been rejected or cancelled" }, { status: 403 })

        if (reg.attendance === 'ATTENDED') return NextResponse.json({ message: "Registration already attended" }, { status: 403 })

        await prisma.registration.update({
            where: { qrCode: code, status: 'APPROVED' },
            data: { attendance: 'ATTENDED' }
        })
        return NextResponse.json({ success: true })
    } catch (error: any) {
        return NextResponse.json({ message: error.message }, { status: 500 })
    }
}
```

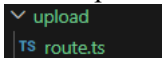
Digunakan untuk menandai kehadiran peserta menggunakan kode QR, dan hanya bisa diakses oleh admin.

- registrations>cancel>route.ts

```ts
Windsurf: Refactor | Explain | Generate JSDoc | ×
export async function POST(req: NextRequest) {
    const { searchParams } = new URL(req.url)
    const token = searchParams.get('token')

    if (!token) return NextResponse.json({ message: "Token required" }, { status: 400 })

    try {
        await prisma.registration.update({
            where: { cancelToken: token },
            data: { status: 'CANCELLED' }
        })
        return NextResponse.json({ success: true })
    } catch (error: any) {
        return NextResponse.json({ message: error.message }, { status: 500 })
    }
}
```

Handler POST yang digunakan untuk membatalkan pendaftaran berdasarkan cancelToken.

11. Foder upload pada folder api berisi file route.ts



upload>routes

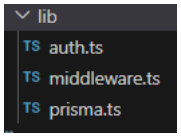
```ts
import { writeFile } from "fs/promises"
import path from "path"
import { NextResponse, NextRequest } from "next/server"
import { requireAdmin } from "@/lib/middleware"
```

- Mengimpor fungsi writeFile dari module fs/promises, yang digunakan untuk menulis file secara asynchronous dalam Node.js. Ini versi promise dari fs.writeFile.
- Mengimpor module path, yang digunakan untuk memanipulasi dan menangani path file dan direktori secara cross-platform.
- NextRequest merupakan Representasi request pada Next.js middleware atau route handler.
- NextResponse digunakan untuk membuat response HTTP (seperti json, redirect, status, dll).
- import { requireAdmin } from "@/lib/middleware" digunakan untuk mengimpor middleware custom bernama requireAdmin, yang kemungkinan besar untuk mengecek apakah user memiliki akses admin dan biasanya akan mengembalikan error atau memblokir akses jika bukan admin.

```
export async function POST(req: NextRequest) {
    const { error } = requireAdmin(req)
    if (error) return error

    const formData = await req.formData()
    const file = formData.get('file') as File

    if (!file) return NextResponse.json({ message: 'No file uploaded' }, { status: 400 })

    const buffer = Buffer.from(await file.arrayBuffer())
    const filename = `${Date.now()}-${file.name.replace(/\s/g, '_')}`
    const uploadDir = path.join(process.cwd(), 'public/uploads/thumbnails')
    const filePath = path.join(uploadDir, filename)

    await writeFile(filePath, buffer)

    return NextResponse.json({ url: `/uploads/thumbnails/${filename}` })
}
```

Digunakan untuk memverifikasi apakah pengguna adalah admin, menerima file dari request POST., menyimpan file ke folder local, dan mengembalikan URL lokasi file.

12. Membuat folder lib pada folder app



- lib>auth.ts

```
import jwt from "jsonwebtoken"

const JWT_SECRET = process.env.JWT_SECRET!
```

  - import jwt from "jsonwebtoken" digunakan untuk mengimpor library jsonwebtoken, yang digunakan untuk membuat (sign) token JWT, memverifikasi (verify) token JWT, dan mendekode (decode) token JWT.
  - const JWT_SECRET = process.env.JWT_SECRET! Digunakan untuk mengambil nilai secret key dari variabel environment JWT_SECRET.
  - Tanda seru ! di akhir (JWT_SECRET!) digunakan dalam TypeScript untuk menandakan bahwa nilai tidak akan undefined, jadi TypeScript tidak akan memberikan error pada baris ini.

```
Windsurf: Refactor | Explain | Generate JSDoc | ×
export function verifyAdminToken(token: string) {
    try {
        const decoded = jwt.verify(token, JWT_SECRET) as { id: string }
        return decoded
    } catch (error) {
        return null
    }
}
```

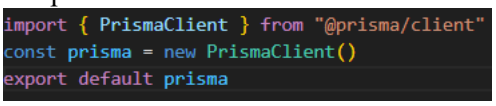
Digunakan untuk memverifikasi token JWT, khususnya token yang kemungkinan digunakan untuk admin.

- lib>middleware.ts

```
export function requireAdmin(req: NextRequest) {
    const auth = req.headers.get("Authorization")
    if (!auth || !auth.startsWith("Bearer ")) return { error: NextResponse.json({ message: "Unauthorized" }, { status: 401 }), admin: null }

    const token = auth.split(' ')[1]
    const admin = verifyAdminToken(token)

    if (!admin) return { error: NextResponse.json({ message: "Invalid token" }, { status: 401 }), admin: null }

    return { error: null, admin }
}
```

Digunakan untuk mengambil header Authorization dari request, memastikan formatnya Bearer <token>: Jika tidak ada atau salah → kirim response 401 Unauthorized. Mengekstrak token dari string Authorization, Verifikasi token menggunakan verifyAdminToken(). Jika token tidak valid → kirim response 401 Invalid token. Jika valid, kembalikan objek { error: null, admin }.

- lib>prisma.ts



```
import { PrismaClient } from "@prisma/client"
const prisma = new PrismaClient()
export default prisma
```

Merupakan cara standar untuk berinteraksi dengan database menggunakan Prisma di aplikasi Node.js atau Next.js. Yang bertujuan untuk menghindari pembuatan banyak instance PrismaClient, yang bisa menyebabkan masalah koneksi, terutama di environment seperti serverless atau saat hot reload (Next.js dev mode).