Nama: Gigih Hermawan
NPM: 22312138

```
import { cookies } from "next/headers";

import { NextResponse } from "next/server";

export async function middleware(request) {
  const cookieStore = await cookies();
  const email = cookieStore.get("email")?.value;

  if (
    request.nextUrl.pathname.startsWith("/api/admin-dashboard/laptop") ||
    request.nextUrl.pathname.startsWith("/admin-dashboard/laptop")
  ) {
    if (email !== "admin@gmail.com") {
      return NextResponse.redirect(new URL("/", request.url));
    }
  }

  if (request.nextUrl.pathname === "/admin-dashboard") {
    if (email !== "admin@gmail.com") {
      return NextResponse.redirect(new URL("/", request.url));
    } else {
      return NextResponse.redirect(
        new URL("/admin-dashboard/laptop", request.url)
      );
    }
  }

  if (request.nextUrl.pathname === "/sign-in") {
    if (email === "admin@gmail.com") {
      return NextResponse.redirect(new URL("/", request.url));
    }
  }
}

```

Fitur:
Menambahkan middleware untuk melakukan pemrosesan di antara permintaan request dan response sebelum mencapai rute atau handler.

```
import { MongoClient } from "mongodb";

const uri = process.env.MONGODB_ATLAS_URI;
const client = new MongoClient(uri);

export async function GET(req) {
  try {
    await client.connect();
    const database = client.db("e_katalog_penyediaan_laptop");
    const collection = database.collection("laptop");

    const category = req?.nextUrl?.searchParams.get("category");
    const search = req?.nextUrl?.searchParams.get("search");

    let result;
    if (category === "merk") {
      result = await collection
        .find({ merk: new RegExp(search, "i") })
        .sort({ createdAt: -1 })
        .toArray();
    }
    if (category === "nama") {
      result = await collection
        .find({ nama: new RegExp(search, "i") })
        .sort({ createdAt: -1 })
        .toArray();
    }
    if (category === "kapasitas_penyimpanan") {
      result = await collection
        .find({ kapasitas_penyimpanan: new RegExp(search, "i") })
        .sort({ createdAt: -1 })
        .toArray();
    }
    if (category === "ram") {
      result = await collection
        .find({ ram: new RegExp(search, "i") })
        .sort({ createdAt: -1 })
        .toArray();
    }
    if (category === "processor") {
      result = await collection
        .find({ processor: new RegExp(search, "i") })
        .sort({ createdAt: -1 })
        .toArray();
    }
    if (category === "vga") {
      result = await collection
        .find({ vga: new RegExp(search, "i") })
        .sort({ createdAt: -1 })
        .toArray();
    }
    if (category === "harga") {
      result = await collection
        .find({ harga: new RegExp(search, "i") })
        .sort({ createdAt: -1 })
        .toArray();
    }

    return new Response(
      JSON.stringify({
        messageResponse: "Data berhasil diambil dari Database",
        result,
      }),
      {
        status: 200,
        statusText: "OK",
      }
    );
  } catch (error) {
    return new Response(error, {
      status: 500,
      statusText: "Internal Server Error",
    });
  } finally {
    await client.close();
  }
}

```

Fitur:
Menambahkan api endpoint untuk method GET pada halaman home.

```
import { cookies } from "next/headers";

export async function POST(req) {
  try {
    const cookieStore = await cookies();

    const formData = await req.formData();
    const email = formData.get("email");
    const password = formData.get("password");

    if (email !== "admin@gmail.com" || password !== "laptop1945") {
      return new Response(
        JSON.stringify({ messageResponse: "Invalid email or password." }),
        {
          status: 200,
          statusText: "OK",
        }
      );
    }

    cookieStore.set("email", email);

    return new Response(
      JSON.stringify({ messageResponse: "Signed in successful." }),
      {
        status: 200,
        statusText: "OK",
      }
    );
  } catch (error) {
    return new Response(
      JSON.stringify({
        messageResponse: "An error occurred, please try again later.",
      }),
      {
        status: 500,
        statusText: "Internal Server Error",
      }
    );
  }
}

```

Fitur:
Menambahkan api endpoint untuk method POST pada halaman sign-in.

```
import { cookies } from "next/headers";

export async function POST() {
  try {
    const cookieStore = await cookies();
    const email = cookieStore.get("email")?.value;

    if (email) {
      (await cookies()).delete("email");

      return new Response(
        JSON.stringify({ messageResponse: "Successfully signed out" }),
        {
          status: 200,
          statusText: "OK",
        }
      );
    } else {
      return new Response(
        JSON.stringify({ messageResponse: "No user is signed in" }),
        {
          status: 200,
          statusText: "OK",
        }
      );
    }
  } catch (error) {
    return new Response(
      JSON.stringify({ messageResponse: "Error occurred during sign-out" }),
      {
        status: 500,
        statusText: "Internal Server Error",
      }
    );
  }
}

```

Fitur:
Menambahkan api endpoint untuk method POST pada komponen sign-out.

```
import { MongoClient, ObjectId } from "mongodb";

import { del, put } from "@vercel/blob";

const uri = process.env.MONGODB_ATLAS_URI;
const client = new MongoClient(uri);

export async function GET() {
  try {
    await client.connect();
    const database = client.db("e_katalog_penyediaan_laptop");
    const collection = database.collection("laptop");

    const result = await collection.find({}).sort({ createdAt: -1 }).toArray();

    return new Response(
      JSON.stringify({
        messageResponse: "Data berhasil diambil dari Database",
        result,
      }),
      {
        status: 200,
        statusText: "OK",
      }
    );
  } catch (error) {
    return new Response(error, {
      status: 500,
      statusText: "Internal Server Error",
    });
  } finally {
    await client.close();
  }
}

```

Fitur:
Menambahkan api endpoint untuk method GET pada halaman admin-dashboard.

```
import { MongoClient, ObjectId } from "mongodb";

import { del, put } from "@vercel/blob";

const uri = process.env.MONGODB_ATLAS_URI;
const client = new MongoClient(uri);

export async function POST(req) {
  try {
    await client.connect();
    const database = client.db("e_katalog_penyediaan_laptop");
    const collection = database.collection("laptop");

    const formData = await req.formData();
    const merk = formData.get("merk");
    const nama = formData.get("nama");
    const img_file = formData.get("img_file");

    const img_file_buffer = new Uint8Array(await img_file.arrayBuffer());

    const blob = await put(
      `e-katalog-penyediaan-laptop/${img_file.name}`,
      img_file_buffer,
      {
        access: "public",
      }
    );

    const kapasitas_penyimpanan = formData.get("kapasitas_penyimpanan");
    const ram = formData.get("ram");
    const processor = formData.get("processor");
    const vga = formData.get("vga");
    const harga = formData.get("harga");

    const date = new Intl.DateTimeFormat("id-ID", {
      dateStyle: "short",
      timeStyle: "long",
      timeZone: "Asia/Jakarta",
    }).format(new Date());

    await collection.insertOne({
      merk,
      nama,
      img_file: {
        pathname: blob.pathname,
        url: blob.url,
      },
      kapasitas_penyimpanan,
      ram,
      processor,
      vga,
      harga,
      createdAt: date,
      updatedAt: date,
    });

    return new Response(
      JSON.stringify({
        messageResponse: "Data berhasil disimpan di Database",
      }),
      {
        status: 200,
        statusText: "OK",
      }
    );
  } catch (error) {
    return new Response(error, {
      status: 500,
      statusText: "Internal Server Error",
    });
  } finally {
    await client.close();
  }
}

```

Fitur:

Menambahkan api endpoint untuk method POST pada halaman admin-dashboard.

```
import { MongoClient, ObjectId } from "mongodb";

import { del, put } from "@vercel/blob";

const uri = process.env.MONGODB_ATLAS_URI;
const client = new MongoClient(uri);

export async function PUT(req) {
  try {
    await client.connect();
    const database = client.db("e_katalog_penyediaan_laptop");
    const collection = database.collection("laptop");

    const formData = await req.formData();
    const _id = formData.get("_id");
    const merk = formData.get("merk");
    const nama = formData.get("nama");

    const kapasitas_penyimpanan = formData.get("kapasitas_penyimpanan");
    const ram = formData.get("ram");
    const processor = formData.get("processor");
    const vga = formData.get("vga");
    const harga = formData.get("harga");

    const date = new Intl.DateTimeFormat("id-ID", {
      dateStyle: "short",
      timeStyle: "long",
      timeZone: "Asia/Jakarta",
    }).format(new Date());

    if (formData.has("img_file")) {
      const img_file = formData.get("img_file");

      const img_file_buffer = new Uint8Array(await img_file.arrayBuffer());

      const blob = await put(
        `e-katalog-penyediaan-laptop/${img_file.name}`,
        img_file_buffer,
        {
          access: "public",
        }
      );

      await collection.updateOne(
        { _id: new ObjectId(`${_id}`) },
        {
          $set: {
            merk,
            nama,
            img_file: {
              pathname: blob.pathname,
              url: blob.url,
            },
            kapasitas_penyimpanan,
            ram,
            processor,
            vga,
            harga,
            updatedAt: date,
          },
        }
      );
    } else {
      await collection.updateOne(
        { _id: new ObjectId(`${_id}`) },
        {
          $set: {
            merk,
            nama,
            kapasitas_penyimpanan,
            ram,
            processor,
            vga,
            harga,
            updatedAt: date,
          },
        }
      );
    }

    return new Response(
      JSON.stringify({
        messageResponse: "Data berhasil diubah di Database",
      }),
      {
        status: 200,
        statusText: "OK",
      }
    );
  } catch (error) {
    return new Response(error, {
      status: 500,
      statusText: "Internal Server Error",
    });
  } finally {
    await client.close();
  }
}

```

Fitur:

Menambahkan api endpoint untuk method PUT pada halaman admin-dashboard.

```
import { MongoClient, ObjectId } from "mongodb";

import { del, put } from "@vercel/blob";

const uri = process.env.MONGODB_ATLAS_URI;
const client = new MongoClient(uri);

export async function DELETE(req) {
  try {
    await client.connect();
    const database = client.db("e_katalog_penyediaan_laptop");
    const collection = database.collection("laptop");

    const formData = await req.formData();
    const _id = formData.get("_id");

    const findDataById = await collection
      .find({ _id: new ObjectId(`${_id}`) })
      .toArray();

    await del(findDataById[0].img_file.url);

    await collection.deleteOne({ _id: new ObjectId(`${_id}`) });

    return new Response(
      JSON.stringify({
        messageResponse: "Data berhasil dihapus di Database",
      }),
      {
        status: 200,
        statusText: "OK",
      }
    );
  } catch (error) {
    return new Response(error, {
      status: 500,
      statusText: "Internal Server Error",
    });
  } finally {
    await client.close();
  }
}

```

Fitur:
Menambahkan api endpoint untuk method DELETE pada halaman admin-dashboard.

```jsx
const SearchLaptop = ({ router, category, setCategory, search, setSearch }) => {
  return (
    <div className="m-16 mb-8">
      <h1 className="text-3xl font-medium text-center mb-8">Laptop Xmute</h1>

      <form
        className="flex flex-wrap gap-9 place-content-center mx-20"
        onSubmit={(e) => {
          e.preventDefault();

          router.push(`/?category=${category}&search=${search}`);
        }}
      >
        <div className="flex flex-wrap gap-3">
          <select
            className="bg-gray-200 p-2 border rounded border-black"
            value={category}
            onChange={(e) => {
              setCategory(e.target.value);
            }}
          >
            <option value="merk">Merk</option>
            <option value="nama">Nama</option>
            <option value="kapasitas_penyimpanan">Kapasitas Penyimpanan</option>
            <option value="ram">RAM</option>
            <option value="processor">Processor</option>
            <option value="vga">VGA</option>
            <option value="harga">Harga</option>
          </select>

          <input
            className="bg-gray-200 w-80 p-2 border rounded border-black"
            type="text"
            search={search}
            onChange={(e) => setSearch(e.target.value)}
          />
        </div>

        <button
          className="bg-gray-700 py-1 px-3 rounded text-white"
          type="submit"
        >
          Search
        </button>
      </form>
    </div>
  );
};

export default SearchLaptop;

```

Fitur:
Menambahkan komponen cari laptop berdasarkan Kategori Merk, Nama, Kapasitas Penyimpanan, RAM, Processor, VGA, Harga.

```
import { MongoClient } from "mongodb";

const uri = process.env.MONGODB_ATLAS_URI;
const client = new MongoClient(uri);

export async function GET(req) {
  try {
    await client.connect();
    const database = client.db("e_katalog_penyediaan_laptop");
    const collection = database.collection("laptop");

    const category = req?.nextUrl?.searchParams.get("category");
    const search = req?.nextUrl?.searchParams.get("search");

    let result;
    if (category === "merk") {
      result = await collection
        .find({ merk: new RegExp(search, "i") })
        .sort({ createdAt: -1 })
        .toArray();
    }
    if (category === "nama") {
      result = await collection
        .find({ nama: new RegExp(search, "i") })
        .sort({ createdAt: -1 })
        .toArray();
    }
    if (category === "kapasitas_penyimpanan") {
      result = await collection
        .find({ kapasitas_penyimpanan: new RegExp(search, "i") })
        .sort({ createdAt: -1 })
        .toArray();
    }
    if (category === "ram") {
      result = await collection
        .find({ ram: new RegExp(search, "i") })
        .sort({ createdAt: -1 })
        .toArray();
    }
    if (category === "processor") {
      result = await collection
        .find({ processor: new RegExp(search, "i") })
        .sort({ createdAt: -1 })
        .toArray();
    }
    if (category === "vga") {
      result = await collection
        .find({ vga: new RegExp(search, "i") })
        .sort({ createdAt: -1 })
        .toArray();
    }
    if (category === "harga") {
      result = await collection
        .find({ harga: new RegExp(search, "i") })
        .sort({ createdAt: -1 })
        .toArray();
    }

    return new Response(
      JSON.stringify({
        messageResponse: "Data berhasil diambil dari Database",
        result,
      }),
      {
        status: 200,
        statusText: "OK",
      }
    );
  } catch (error) {
    return new Response(error, {
      status: 500,
      statusText: "Internal Server Error",
    });
  } finally {
    await client.close();
  }
}

```

Fitur:
Menambahkan api endpoint untuk method GET pada halaman home untuk pencarian laptop berdasarkan Kategori tertentu.