Studi Kasus Node JS Membuat Aplikasi Company Profile

Source Code Project Ini

Source code project ini ada di folder "company_profile".

Website Saya (Jika Masih Ada)

- https://rakifsul.my.id

Cara Mencoba Kode Project Ini

Sekarang, saya akan membahas cara mencoba kode ini di komputer lokal dengan sistem operasi Windows 11.

Untuk mencoba kode ini, copy file .env.example sebagai .env

Sekarang, kita memiliki file .env dalam folder source code yang secara default isinya sama dengan .env.example.

Selanjutnya, Anda bisa mengubah isi dari .env sesuai dengan detail database dan sistem Anda.

Namun, untuk kesamaan, jangan ubah dulu isi dari .env.

Kode ini membutuhkan MySQL server untuk staging dan production.

Jadi, pastikan Anda telah menginstallnya di komputer Anda dan membuat databasenya sesuai konfigurasi yang ada di file .env tadi.

Mencoba Di Environment Development

Sekarang, kita telah memiliki database di MySQL dan file .env yang sudah dikonfigurasi.

Pastikan Anda berada dalam folder source code.

Selanjutnya, jalankan:

```
npm install
```

Selanjutnya, jalankan:

```
npm run db:dev:refresh
```

Catat bahwa dev menggunakan sqlite, jadi tidak perlu install MySQL.

Selanjutnya, jalankan:

```
npm run dev
```

Mencoba Di Environment Production

Karena versi production dari aplikasi ini memerlukan MySQL server, maka install itu terlebih dahulu.

Selanjutnya, buat database bernama "company_profile-prod".

Selanjutnya, ubah .env bagian ini menjadi:

```
# pilih salah satu
#KNEX_ENV=development
#KNEX_ENV=staging
KNEX_ENV=production
```

Pastikan bagian ini sesuai dengan settingan MySQL Anda:

```
# production database
KNEX_PROD_HOST=127.0.0.1
KNEX_PROD_PORT=3306
KNEX_PROD_USER=root
KNEX_PROD_PASSWORD=root
KNEX_PROD_DATABASE=company_profile-prod
```

Sekarang, kita telah memiliki database di MySQL dan file .env yang sudah dikonfigurasi.

Pastikan Anda berada dalam folder source code.

Selanjutnya, jalankan:

```
npm install
```

Selanjutnya, jalankan:

```
npm run db:prod:refresh
```

Selanjutnya, jalankan:

```
npm run dev
```

Mencoba Di Environment Staging

Pastikan .env bagian ini seperti ini:

```
# pilih salah satu
#KNEX_ENV=development
KNEX_ENV=staging
#KNEX_ENV=production
```

Caranya serupa dengan di environment production, tinggal replace prod dengan stg di langkah ini:

```
npm run db:stg:refresh
```

Mengunjungi Aplikasi

Terakhir, buka browser Anda ke alamat yang tertera di BASE_URL yang ada di .env

Secara default adalah:

http://localhost:3000

Di bagian paling bawah halaman tersebut, ada link login dan register untuk admin.

Default admin login:

```
username: admin@example.com
password: admin
```

Mengubah Favicon

Untuk mengubah favicon di halaman depan, replace favicon.png yang ada di folder "/public/img" yang ada di dalam folder source code.

Pendahuluan

Node JS telah menjadi salah satu platform yang populer dalam pengembangan aplikasi web, terutama bagi mereka yang mencari efisiensi dan kinerja tinggi.

Dengan menggunakan JavaScript di sisi server, Node JS menawarkan berbagai keunggulan seperti non-blocking I/O dan arsitektur event-driven yang memungkinkan penanganan banyak koneksi secara simultan tanpa membebani server.

Pada artikel ini, saya akan membahas studi kasus tentang cara membuat aplikasi company profile menggunakan Node JS.

Aplikasi company profile adalah sebuah aplikasi web yang dirancang untuk menampilkan informasi penting tentang sebuah perusahaan.

Informasi ini mencakup sejarah perusahaan, visi dan misi, layanan atau produk yang ditawarkan, portofolio proyek, hingga informasi kontak.

Aplikasi ini bertujuan untuk memberikan gambaran lengkap dan profesional kepada klien atau calon pelanggan tentang identitas dan kapabilitas perusahaan.

Tujuan dari artikel ini adalah untuk memberikan panduan praktis langkah demi langkah bagi pengembang pemula hingga menengah dalam membangun aplikasi company profile yang fungsional dan menarik.

Kita akan membahas mulai dari pembentukan struktur file dan folder project, penulisan source code, pembuatan halaman-halaman web, hingga implementasi fitur-fitur dinamis yang umum digunakan dalam aplikasi company profile.

Dengan mengikuti tutorial ini, diharapkan pembaca akan mendapatkan pemahaman yang lebih baik tentang penggunaan Node JS dalam konteks dunia nyata, serta keterampilan praktis yang dapat langsung diterapkan dalam proyek pengembangan mereka.

Mari kita mulai.

Tujuan

Tujuan dari artikel ini adalah:

- Pembaca mengenal apa itu aplikasi company profile.
- Pembaca mampu membuat aplikasi company profile yang dinamis dengan Node JS dan database dari jenis SQL (SQLite dan MySQL).
- Pembaca mampu menerapkan sistem login dan register pada aplikasi company profile.

Prasyarat

Prasyarat dari artikel ini adalah:

- Menggunakan Windows 11.
- Menggunakan Node JS versi 20.9.0.

- Telah menginstall MySQL Server.
- Telah menginstall MySQL client (bebas, asal Anda bisa menggunakannya).
- Telah menguasai HTML, CSS, dan JavaScript secara lancar.

Langkah-langkah

Walaupun saya telah menyediakan file project yang sudah jadi, ada baiknya jika Anda tahu langkah demi langkah yang harus dilakukan untuk membentuk project tersebut.

Di bagian ini, Anda akan dijelaskan bagaimana membentuk project tersebut.

Di bagian ini, saya tidak akan banyak menjelaskan kode.

Itu karena bagian "Langkah-Langkah" terfokus bagaimana cara membentuk project yang siap pakai.

Adapun pembahasan ada pada bagian "Pembahasan".

Agar Lebih Jelas

Project ini akan dibuat dalam folder bernama "company_profile".

Karena project yang kita bahas memiliki subfolder yang banyak dan masing masing subfolder juga ada isinya, maka sepakati bahwa tiap nama subfolder atau file path-nya relatif terhadap folder project.

Sebagai contoh:

- Jika saya mengatakan "package.json", itu artinya adalah file "company_profile/package.json".
- Jika saya menulis subfolder "controllers", itu artinya adalah subfolder "company_profile/controllers".
- Jika saya menulis file "routes/admin.js", itu artinya adalah file "company_profile/routes/admin.js".
- Selain itu, jika saya menulis folder project, itu artinya adalah folder "company_profile".

Menginisialisasi Project dengan npm

Pertama, buatlah sebuah folder bernama "company_profile".

Kemudian, masuk ke dalam folder tersebut dengan:

```
cd company_profile
```

Selanjutnya, jalankan perintah ini:

```
npm init -y
```

Nanti, Anda akan mendapatkan file "package.json".

Sekarang, replace isi file "package.json" dengan kode ini:

```
{
    "name": "company_profile",
    "version": "2024.05.28",
    "main": "app.js",
    "private": true,
    "scripts": {
        "start": "node app.js",
        "dev": "nodemon -e js,ejs,html -w . -w public -w views -w routes -w models
app.js --debug",
        "db:dev:refresh": "npx knex migrate:rollback --env development --knexfile
./databases/knexfile.js && npx knex migrate:latest --env development --knexfile
./databases/knexfile.js",
        "db:stg:refresh": "npx knex migrate:rollback --env staging --knexfile
./databases/knexfile.js && npx knex migrate:latest --env staging --knexfile
./databases/knexfile.js",
        "db:prod:refresh": "npx knex migrate:rollback --env production --knexfile
./databases/knexfile.js && npx knex migrate:latest --env production --knexfile
./databases/knexfile.js"
    },
    "dependencies": {
        "bcryptjs": "^2.4.3",
        "connect-flash": "^0.1.1",
        "cookie-parser": "~1.4.6",
        "dotenv": "^16.0.3",
        "ejs": "~3.1.9",
        "express": "~4.18.2",
        "express-session": "^1.17.3",
        "http-errors": "~2.0.0",
        "joi": "^17.9.1",
        "knex": "^2.4.2",
        "morgan": "~1.10.0",
        "multer": "^1.4.3",
        "mysql2": "^3.2.1",
        "node-os-utils": "^1.3.7",
        "session-file-store": "^1.5.0",
        "sqlite3": "^5.1.6",
        "uuid": "^9.0.0"
    },
    "devDependencies": {
        "nodemon": "^2.0.22"
    }
}
```

Meng-install Dependencies

Saat ini, Anda telah menginisialisasi project.

Dengan menginisialisasi project, Anda mendapatkan file "package.json"

Di file "package.json" itulah dependencies akan atau telah didaftarkan.

Maka dari itu, jalankan ini dari folder project:

```
npm install
```

Itu untuk meng-install dan mendaftarkan dependencies yang telah terdaftar.

Jika itu sudah dilakukan, maka akan muncul folder baru di dalam folder project bernama folder "node_modules".

Membuat File ".env.example" dan ".env"

Buatlah file ".env.example" dan ".env", kemudian isi dengan kode ini:

```
BASE_URL=http://localhost:3000
SESSION_SECRET=secret1

# pilih salah satu
KNEX_ENV=development
#KNEX_ENV=staging
#KNEX_ENV=production

# development database (jangan gunakan path. nama file saja.)
KNEX_DEV_DATABASE=company_profile-dev.sqlite3

# staging database
KNEX_STG_HOST=127.0.0.1
KNEX_STG_PORT=3306
KNEX_STG_USER=root
KNEX_STG_PASSWORD=root
KNEX_STG_DATABASE=company_profile-stg

# production database
KNEX_PROD_HOST=127.0.0.1
KNEX_PROD_PORT=3306
KNEX_PROD_USER=root
KNEX_PROD_PASSWORD=root
KNEX_PROD_DATABASE=company_profile-prod
```

Membuat File "app.js"

Buatlah file "app.js", kemudian isi dengan kode ini:

```
// script ini adalah script utama dari aplikasi ini.
// tugasnya adalah melakukan konfigurasi umum pada aplikasi ini.

// begin: import modules
const createError = require("http-errors");
const express = require("express");
const path = require("path");
const cookieParser = require("cookie-parser");
```

```
const logger = require("morgan");
const session = require("express-session");
const FileStore = require("session-file-store")(session);
const flash = require("connect-flash");
const multer = require("multer");
const { v4: uuidv4 } = require("uuid");
const url = require("url");
require("dotenv").config();

const indexRouter = require("./routes/index");
const adminRouter = require("./routes/admin");
const authRouter = require("./routes/auth");
// end: import modules

// inisialisasi expressjs
const app = express();

// set view folder ke folder "views"
app.set("views", path.join(__dirname, "views"));

// set view engine yang digunakan adalah EJS
app.set("view engine", "ejs");

// inisialisasi morgan
app.use(logger("dev"));

// parse body json
app.use(express.json({ limit: "100mb" }));

// parse body urlencoded
app.use(
    express.urlencoded({
        limit: "100mb",
        extended: true,
        parameterLimit: 100000,
    })
);

// setup multer agar bisa upload gambar
app.use(
    multer({
        storage: multer.diskStorage({
            destination: (req, file, callback) => {
                callback(null, "./uploads");
            },
            filename: (req, file, callback) => {
                callback(null, uuidv4() + "-" + file.originalname);
            },
        }),
        fileFilter: (req, file, callback) => {
            if (file.mimetype == "image/png" || file.mimetype == "image/jpg" ||
file.mimetype == "image/jpeg" || file.mimetype == "image/gif") {
                callback(null, true);
            } else {
```

```
                callback(null, false);
            }
        },
    }).single("upload")
);

// parse cookie header
app.use(cookieParser());

// agar bisa menggunakan session
app.use(
    session({
        secret: process.env.SESSION_SECRET,
        // store: new FileStore(), // jika di-enable simpan session di file (ada
bug nya)
        resave: false,
        saveUninitialized: false,
    })
);

// inisialisasi connect-flash
app.use(flash());

// setup folder statis untuk menyimpan resources
app.use("/public", express.static(path.join(__dirname, "public")));
app.use("/uploads", express.static(__dirname + "/uploads"));

// route admin ada di path "/admin" nantinya pada URL.
app.use("/admin", adminRouter);

// route auth ada di path "/auth" nantinya pada URL.
app.use("/auth", authRouter);

// route "/" ada di path "/" nantinya pada URL.
app.use("/", indexRouter);

// setup halaman error 404 jika terjadi
app.use(function (req, res, next) {
    next(createError(404));
});

// setup halaman error lainnya jika terjadi
app.use(function (err, req, res, next) {
    res.status(err.status || 500);
    res.render("error", {
        error: {
            message: err.message,
            status: err.status,
            stack: err.stack,
        },
    });
});

// jalankan server pada port yang ada di BASE_URL atau 3000 jika tidak ada
```

```
const port = url.parse(process.env.BASE_URL).port | 3000;
app.listen(port, function () {
    console.log(`server berjalan di port ${port}`);
});
```

Mengkopi Subfolder "public" dari Contoh Project ke Project Ini

Sebenarnya, Anda bisa membuat subfolder ini satu demi satu, tapi agar lebih cepat dikopi saja.

Folder ini hanya untuk kumpulan file-file statis seperti Gambar, CSS dan JavaScript browser.

Jadi, bukalah contoh project "company_profile" yang disertakan kemudian copy folder "public"-nya ke project yang Anda buat ini.

Membuat Subfolder "lib", "sessions", dan "uploads"

Buatlah subfolder "lib", "sessions", dan "uploads".

Kosongkan isi subfolder-subfolder tersebut.

Membuat Subfolder "controllers"

Buatlah folder bernama "controllers", kemudian isi dengan file-file ini:

- admin.js
- auth.js
- index.js

Isi file "controllers/admin.js" dengan kode ini:

```
// script ini akan dipanggil di routes/admin.js

const bcrypt = require("bcryptjs");
const fs = require("fs");
const createError = require("http-errors");
const osu = require("node-os-utils");
const knex = require("../databases/connection");

// render halaman admin index
module.exports.getIndex = async function (req, res, next) {
    try {
        // ambil data text pertama dari tabel texts
        const text = await knex("texts").first();

        // render dan kirimkan data tadi ke front end
        res.render("admin/layout.ejs", {
            child: "admin/index.ejs",
            clientScript: "admin/index.js.ejs",
            data: {
                text: text,
            },
```

```
        });
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// dapatkan data penggunaan CPU
module.exports.getCPUUsage = async function (req, res, next) {
    osu.cpu.usage().then((cpuPercentage) => {
        res.status(200).json({
            name: "CPU Usage",
            value: cpuPercentage,
        });
    });
};

// dapatkan data penggunaan memory
module.exports.getMemoryUsage = function (req, res) {
    osu.mem.used().then((memUsed) => {
        res.status(200).json({
            name: "Memory Usage",
            value: (memUsed.usedMemMb / memUsed.totalMemMb) * 100,
        });
    });
};

// render halaman settings
module.exports.getSettings = async function (req, res) {
    try {
        // ambil data text pertama dari tabel texts
        const text = await knex("texts").first();

        // render dan kirimkan data tadi ke front end
        res.render("admin/layout", {
            child: "admin/settings.ejs",
            clientScript: "admin/settings.js.ejs",
            data: {
                adminEmail: req.session.admin.email,
                text: text,
                errors: req.flash("errors"),
            },
        });
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// handle edit settings
module.exports.postSettingsEdit = async function (req, res, next) {
    try {
        // bongkar request body
        const { email, password } = req.body;
```

```
        if (password && email) {
            // jika password dan email ada

            // update data admin
            const ret = await knex("admins")
                .where({ email: req.session.admin.email })
                .update({
                    email: email,
                    password: bcrypt.hashSync(password, 12),
                });
        } else if (email) {
            // jika hanya email yang ada

            // update data admin
            const ret = await knex("admins").where({ email:
req.session.admin.email }).update({
                email: email,
            });
        }

        // redirect ke settings
        res.redirect("/admin/settings");
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// render halaman messages
module.exports.getMessagesIndex = async function (req, res) {
    try {
        // ambil data text pertama dari tabel texts
        const text = await knex("texts").first();

        // ambil data messages dari tabel messages
        const messages = await knex("messages");

        // render dan kirimkan data tadi ke front end
        res.render("admin/layout", {
            child: "admin/messages.ejs",
            clientScript: "admin/messages.js.ejs",
            data: {
                text: text,
                results: messages,
            },
        });
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// handle delete message
```

```
module.exports.getMessagesDelete = async function (req, res, next) {
    try {
        // delete message yang id-nya adalah req.params.id
        // atau dengan kata lain
        // yang id-nya sama dengan yang ada di URL
        await knex("messages")
            .where({
                _id: req.params.id,
            })
            .del();

        // redirect ke messages
        res.redirect("/admin/messages");
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// render halaman texts
module.exports.getTextsIndex = async function (req, res) {
    try {
        // ambil data text pertama dari tabel texts
        const text = await knex("texts").first();

        // render dan kirimkan data tadi ke front end
        res.render("admin/layout", {
            child: "admin/texts.ejs",
            clientScript: "admin/texts.js.ejs",
            data: {
                adminEmail: req.session.admin.email,
                text: text,
                errors: req.flash("errors"),
            },
        });
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// handle edit data dari tabel texts
module.exports.postTextsEdit = async function (req, res, next) {
    try {
        // ambil data text pertama dari tabel texts
        const target = await knex("texts").first();

        // update data tersebut
        const ret = await knex("texts").where({ _id: target._id }).update({
            siteTitle: req.body.siteTitle,
            siteSEOTitle: req.body.siteSEOTitle,
            siteDescription: req.body.siteDescription,
            siteSEODescription: req.body.siteSEODescription,
            aboutSubHeading: req.body.aboutSubHeading,
```

```
            aboutSubText: req.body.aboutSubText,
            infoSubHeading: req.body.infoSubHeading,
            address: req.body.address,
            phone: req.body.phone,
            email: req.body.email,
        });

        // redirect ke halaman texts
        res.redirect("/admin/texts");
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// render halaman skills
module.exports.getSkillsIndex = async function (req, res, next) {
    try {
        // ambil data text pertama dari tabel texts
        const text = await knex("texts").first();

        // ambil data skills dari tabel skills
        const allSkills = await knex("skills");

        // render dan kirim data tersebut ke front end
        res.render("admin/layout.ejs", {
            child: "admin/skills.ejs",
            clientScript: "admin/skills.js.ejs",
            data: {
                results: allSkills,
                text: text,
            },
        });
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// handle penambahan skill
module.exports.postSkillsAdd = async function (req, res, next) {
    try {
        // bongkar request body
        const { title, level } = req.body;

        // insert skill baru, judul dan levelnya
        const skillId = await knex("skills").insert({
            title: title,
            level: level,
        });

        // redirect ke halaman skills
        res.redirect("/admin/skills");
    } catch (err) {
```

```
        console.log(err);
        next(createError(500));
    }
};

// handle delete skill
module.exports.getSkillsDelete = async function (req, res, next) {
    try {
        // delete skill yang id-nya adalah req.params.id
        // atau dengan kata lain
        // yang id-nya sama dengan yang ada di URL
        await knex("skills")
            .where({
                _id: req.params.id,
            })
            .del();

        // redirect ke halaman skills
        res.redirect("/admin/skills");
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// render halaman services
module.exports.getServicesIndex = async function (req, res, next) {
    try {
        // ambil data text pertama dari tabel texts
        const text = await knex("texts").first();

        // ambil data services dari tabel services
        const allServices = await knex("services");

        // render dan kirim data tersebut ke front end
        res.render("admin/layout.ejs", {
            child: "admin/services.ejs",
            clientScript: "admin/services.js.ejs",
            data: {
                results: allServices,
                text: text,
            },
        });
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// handle tambah service
module.exports.postServicesAdd = async function (req, res, next) {
    try {
        // bongkar request body
        const { title, description, svg } = req.body;
```

```
        // insert service baru, judul, deskripsi, dan svg nya
        const serviceId = await knex("services").insert({
            title: title,
            description: description,
            svg: svg,
        });

        // redirect ke halaman services
        res.redirect("/admin/services");
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// handle delete service
module.exports.getServicesDelete = async function (req, res, next) {
    try {
        // delete service yang id-nya adalah req.params.id
        // atau dengan kata lain
        // yang id-nya sama dengan yang ada di URL
        await knex("services")
            .where({
                _id: req.params.id,
            })
            .del();

        // redirect ke halaman services
        res.redirect("/admin/services");
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// render halaman carousel
module.exports.getCarouselsIndex = async function (req, res, next) {
    try {
        // ambil data text pertama dari tabel texts
        const text = await knex("texts").first();

        // ambil data carousels dari tabel carousels
        const allCarousels = await knex("carousels");

        // render dan kirim data tadi ke front end
        res.render("admin/layout.ejs", {
            child: "admin/carousels.ejs",
            clientScript: "admin/carousels.js.ejs",
            data: {
                results: allCarousels,
                text: text,
            },
        });
```

```
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// handle upload gambar carousel
module.exports.postCarouselsUpload = async function (req, res, next) {
    try {
        if (req.file) {
            // jika request file ada, yang artinya gambar diupload

            // bongkar request body
            const { title, description } = req.body;

            // insert carousel baru, judul, deskripsi, dan path dari file
gambarnya
            const fileId = await knex("carousels").insert({
                title: title,
                description: description,
                path: req.file.path.replace("\\", "/"),
            });
        }

        // redirect ke halaman carousels
        res.redirect("/admin/carousels");
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// handle delete carousel
module.exports.getCarouselsDelete = async function (req, res, next) {
    try {
        // dapatkan dulu datanya
        const willBeDeleted = await knex("carousels").where({
            _id: req.params.id,
        });

        // delete carousel yang id-nya adalah req.params.id
        // atau dengan kata lain
        // yang id-nya sama dengan yang ada di URL
        // ini yang di database
        await knex("carousels")
            .where({
                _id: req.params.id,
            })
            .del();

        // hapus file nya
        // ini yang di folder upload
        fs.unlinkSync("./" + willBeDeleted[0].path);
```

```
        // redirect ke halaman carousels
        res.redirect("/admin/carousels");
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// render halaman portfolios
module.exports.getPortfoliosIndex = async function (req, res, next) {
    try {
        // ambil data text pertama dari tabel texts
        const text = await knex("texts").first();

        // ambil data portfolios dari tabel portfolios
        const allPortfolios = await knex("portfolios");

        // render dan kirim data tadi ke front end
        res.render("admin/layout.ejs", {
            child: "admin/portfolios.ejs",
            clientScript: "admin/portfolios.js.ejs",
            data: {
                results: allPortfolios,
                text: text,
            },
        });
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// handle upload gambar portfolio
module.exports.postPortfoliosUpload = async function (req, res, next) {
    try {
        if (req.file) {
            // jika request file ada, yang artinya gambar diupload

            // bongkar request body
            const { title } = req.body;

            // insert portfolio baru, judul, dan path dari file gambarnya
            const fileId = await knex("portfolios").insert({
                title: title,
                path: req.file.path.replace("\\", "/"),
            });
        }

        // redirect ke halaman portfolios
        res.redirect("/admin/portfolios");
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
```

```
};

// handle delete portfolio
module.exports.getPortfoliosDelete = async function (req, res, next) {
    try {
        // dapatkan dulu datanya
        const willBeDeleted = await knex("portfolios").where({
            _id: req.params.id,
        });

        // delete portfolio yang id-nya adalah req.params.id
        // atau dengan kata lain
        // yang id-nya sama dengan yang ada di URL
        // ini yang di database
        await knex("portfolios")
            .where({
                _id: req.params.id,
            })
            .del();

        // hapus file nya
        // ini yang di folder upload
        fs.unlinkSync("./" + willBeDeleted[0].path);

        // redirect ke halaman portfolios
        res.redirect("/admin/portfolios");
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};
```

Isi file "controllers/auth.js" dengan kode ini:

```
// script ini akan dipanggil di routes/auth.js

// begin: import modules
const createError = require("http-errors");
const bcrypt = require("bcryptjs");
const Joi = require("joi");
const knex = require("../databases/connection");
// end: import modules

// render halaman login
module.exports.getLogin = async function (req, res, next) {
    try {
        // ambil data tabel texts
        const text = await knex("texts").first();

        // render dan kirim data tadi ke front end
        res.render("auth/layout.ejs", {
            child: "auth/login.ejs",
```

```
            clientScript: "auth/login.js.ejs",
            data: {
                text: text,
                errors: req.flash("errors"),
            },
        });
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// handle form login
module.exports.postLogin = async function (req, res, next) {
    // validasi request body
    const schema = Joi.object({
        email: Joi.string().required(),
        password: Joi.string().required(),
    });

    const validObj = schema.validate(req.body);
    if (validObj.error) {
        // jika ada error dalam request body

        // buat pesan error flash
        req.flash("errors", validObj.error.details);
        res.status(422).redirect("/auth/login");
        res.end();
        return;
    }

    // bongkar request body
    const { email, password } = req.body;

    try {
        // ambil admin dari tabel admins yang emailnya adalah email dari request
body tadi
        const admin = await knex("admins").where({ email: email }).first();

        if (admin) {
            // jika admin tadi ada

            if (email == admin.email) {
                // jika email dari request body sama dengan yang di tabel

                // bandingkan passwordnya
                if (bcrypt.compareSync(password, admin.password) == true) {
                    // jika password sama

                    // masukkan data-data ini ke session
                    req.session.isLoggedIn = true;
                    req.session.admin = admin;
                    req.session.save();
```

```
                    // redirect ke halaman admin
                    res.redirect("/admin");
                } else {
                    // jika password beda

                    // redirect ke halaman login
                    res.redirect("/auth/login");
                }
            } else {
                // jika email dari request body beda dengan yang di tabel

                // redirect ke halaman login
                res.redirect("/auth/login");
            }
        } else {
            // jika admin tidak ada

            // redirect ke halaman login
            res.redirect("/auth/login");
        }
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// render halaman register
module.exports.getRegister = async function (req, res, next) {
    try {
        // ambil data tabel texts
        const text = await knex("texts").first();

        // render dan kirim data tadi ke front end
        res.render("auth/layout.ejs", {
            child: "auth/register.ejs",
            clientScript: "auth/register.js.ejs",
            data: {
                text: text,
                errors: req.flash("errors"),
            },
        });
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// handle form register
module.exports.postRegister = async function (req, res, next) {
    // validasi request body
    const schema = Joi.object({
        email: Joi.string().required(),
        name: Joi.string().required(),
        password: Joi.string().required(),
```

```
        password_repeat: Joi.ref("password"),
    }).with("password", "password_repeat");

    const validObj = schema.validate(req.body);
    if (validObj.error) {
        // jika ada error dalam request body

        // buat pesan error flash
        console.log(validObj.error.details);
        req.flash("errors", validObj.error.details);
        res.status(422).redirect("/auth/register");
        res.end();
        return;
    }

    // bongkar request body
    const { email, name, password } = req.body;
    try {
        // masukkan data admin baru ke tabel admins
        const admin = await knex("admins").insert({
            email: email,
            name: name,
            password: bcrypt.hashSync(password, 12),
        });

        // redirect ke halaman login
        res.redirect("/auth/login");
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// handle logout
module.exports.getLogout = function (req, res, next) {
    req.session.destroy((err) => {
        console.log(err);
        res.clearCookie("connect.sid");
        res.redirect("/auth/login");
    });
};
```

Isi file "controllers/index.js" dengan kode ini:

```
// script ini akan dipanggil di routes/index.js

// begin: import modules
const createError = require("http-errors");
const knex = require("../databases/connection");
// end: import modules

// merender homepage
```

```
module.exports.getIndex = async function (req, res, next) {
    try {
        // ambil isi masing-masing tabel dan simpan ke variabel
        const text = await knex("texts").first();
        const skills = await knex("skills");
        const services = await knex("services");
        const carousels = await knex("carousels");
        const portfolios = await knex("portfolios");

        // render dan kirimkan datanya yang berupa isi tabel
        res.render("index.ejs", {
            text: text,
            skills: skills,
            services: services,
            carousels: carousels,
            portfolios: portfolios,
        });
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// meng-handle form kontak post
module.exports.postIndexSendMessage = async function (req, res, next) {
    try {
        // bongkar request body
        const { name, email, message } = req.body;

        // insert data pesan form kontak
        const messageId = await knex("messages").insert({
            name: name,
            email: email,
            message: message,
        });

        // refresh halaman
        res.redirect("/");
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};
```

Membuat Subfolder "databases"

Buatlah subfolder bernama "databases", kemudian buat lagi subfolder dari subfolder "databases" bernama "migrations".

Di subfolder "databases/migrations", buat file bernama "20230316035229_init.js". Kemudian isi dengan kode ini:

```
const bcrypt = require("bcryptjs");

/**
 * @param { import("knex").Knex } knex
 * @returns { Promise<void> }
 */
exports.up = function (knex) {
    return knex.schema
        .createTable("admins", function (table) {
            table.increments("_id");
            table.string("name", 1000).notNullable();
            table.string("email", 1000).notNullable();
            table.string("password", 1000).notNullable();
            table.timestamps(true, true, true);
        })
        .createTable("messages", function (table) {
            table.increments("_id");
            table.string("name", 1000).notNullable();
            table.string("email", 1000).notNullable();
            table.string("message", 1000).notNullable();
            table.timestamps(true, true, true);
        })
        .createTable("texts", function (table) {
            table.increments("_id");
            table.string("siteTitle", 1000).notNullable();
            table.string("siteSEOTitle", 1000).notNullable();
            table.string("siteDescription", 1000);
            table.string("siteSEODescription", 1000);
            table.string("aboutSubHeading", 1000);
            table.string("aboutSubText", 5000);
            table.string("infoSubHeading", 1000);
            table.string("address", 1000);
            table.string("phone", 1000);
            table.string("email", 1000);
            table.timestamps(true, true, true);
        })
        .createTable("skills", function (table) {
            table.increments("_id");
            table.string("title", 1000).notNullable();
            table.integer("level").notNullable();
            table.timestamps(true, true, true);
        })
        .createTable("services", function (table) {
            table.increments("_id");
            table.string("title", 1000).notNullable();
            table.string("description", 1000).notNullable();
            table.string("svg", 10000).notNullable();
            table.timestamps(true, true, true);
        })
        .createTable("portfolios", function (table) {
            table.increments("_id");
            table.string("title", 1000).notNullable();
            table.string("path", 1000).notNullable();
```

```
            table.timestamps(true, true, true);
        })
        .createTable("carousels", function (table) {
            table.increments("_id");
            table.string("title", 1000).notNullable();
            table.string("description", 1000);
            table.string("path", 1000).notNullable();
            table.timestamps(true, true, true);
        })
        .then(() => {
            return knex("admins").insert([
                {
                    name: "admin",
                    email: "admin@example.com",
                    password: bcrypt.hashSync("admin", 12),
                },
            ]);
        })
        .then(() => {
            return knex("texts").insert([
                {
                    siteTitle: "Company Name",
                    siteSEOTitle: "Company Name - A leading software development
agency...",
                    siteDescription: "Company Name is a leading software
development agency...",
                    siteSEODescription: "Company Name is a leading software
development agency...",
                    aboutSubHeading: "Why?",
                    aboutSubText: "Lorem ipsum dolor sit amet consectetur
adipisicing elit. Odit magnam ad doloremque illum iure repellat vero neque dolore
consectetur, quidem corporis tenetur perspiciatis mollitia quos nemo architecto!
Veniam facilis nesciunt, dignissimos in officia tempora repellat autem ipsum,
nostrum accusamus aliquid reiciendis inventore quos aperiam voluptates debitis
magni sit laborum obcaecati possimus necessitatibus quasi adipisci ducimus?
Adipisci enim accusantium, dignissimos quae asperiores eaque tempore
reprehenderit? Quaerat blanditiis sint rerum accusamus aspernatur provident sunt
libero quibusdam totam. Magnam doloribus nisi, quas eligendi quasi modi, velit
alias nulla quae animi excepturi officiis, ipsam voluptatum? Maxime reprehenderit
tenetur facilis, unde obcaecati ratione id magni!",
                    infoSubHeading: "Our Office",
                    address: "Jl. Some Street No. 15, Some City, Some Province,
Some Country",
                    phone: "(021) yyy-xxxx",
                    email: "admin@example.com",
                },
            ]);
        });
};

/**
 * @param { import("knex").Knex } knex
 * @returns { Promise<void> }
 */
```

```
exports.down = function (knex) {
    return
knex.schema.dropTable("skills").dropTable("services").dropTable("portfolios").drop
Table("carousels").dropTable("texts").dropTable("messages").dropTable("admins");
};
```

Di subfolder "databases", buat file-file ini:

- connection.js
- knexfile.js

Isi file "databases/connection.js" dengan kode ini:

```
const knex = require("knex");
const knexfile = require("./knexfile");
const path = require("path");

let knexEnv;
if (process.env.KNEX_ENV === "development") {
    knexEnv = knexfile.development;
} else if (process.env.KNEX_ENV === "staging") {
    knexEnv = knexfile.staging;
} else if (process.env.KNEX_ENV === "production") {
    knexEnv = knexfile.production;
} else {
    throw Error("invalid environment.");
}

console.log("BUGFIX WORKAROUND !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");
knexfile.development.connection.filename = path.join(__dirname,
knexfile.development.connection.filename);
console.log(knexfile.development.connection.filename);

const db = knex(knexEnv);

module.exports = db;
```

Isi file "databases/knexfile.js" dengan kode ini:

```
// Update with your config settings.
require("dotenv").config({
    path: "../.env",
});

/**
 * @type { Object.<string, import("knex").Knex.Config> }
 */
module.exports = {
    development: {
        client: "sqlite3",
```

```
        connection: {
            filename: process.env.KNEX_DEV_DATABASE,
        },
        migrations: {
            tableName: "knex_migrations",
        },
    },

    staging: {
        client: "mysql2",
        connection: {
            host: process.env.KNEX_STG_HOST,
            port: process.env.KNEX_STG_PORT,
            user: process.env.KNEX_STG_USER,
            password: process.env.KNEX_STG_PASSWORD,
            database: process.env.KNEX_STG_DATABASE,
        },
        migrations: {
            tableName: "knex_migrations",
        },
    },

    production: {
        client: "mysql2",
        connection: {
            host: process.env.KNEX_PROD_HOST,
            port: process.env.KNEX_PROD_PORT,
            user: process.env.KNEX_PROD_USER,
            password: process.env.KNEX_PROD_PASSWORD,
            database: process.env.KNEX_PROD_DATABASE,
        },
        migrations: {
            tableName: "knex_migrations",
        },
    },
};
```

Membuat Subfolder "middlewares"

Buatlah subfolder bernama "middlewares", kemudian isi dengan file:

- sessionchecker.js

Isi file "middlewares/sessionchecker.js" dengan kode ini:

```
// script ini tugasnya adalah menjadi middleware

module.exports.notLoggedIn = (req, res, next) => {
    if (!req.session.isLoggedIn) {
        // jika varieble isLoggedIn dalam session false, null, atau undefined

        // maka redirect ke /auth/login
```

```
        return res.redirect("/auth/login");
    }
    next();
};

module.exports.loggedIn = (req, res, next) => {
    if (req.session.isLoggedIn) {
        // jika varieble isLoggedIn dalam session true

        // maka redirect ke /admin
        return res.redirect("/admin");
    }
    next();
};
```

Membuat Subfolder "routes"

Buatlah subfolder "routes", kemudian isi dengan file-file ini:

- admin.js
- auth.js
- index.js

Isi file "routes/admin.js" dengan kode ini:

```
// script ini tugasnya adalah menghubungkan controllers/admin.js dengan route.

const express = require("express");
const adminController = require("../controllers/admin");

// session checker digunakan untuk authorization
const sessionChecker = require("../middlewares/sessionchecker");

// buat objek router agar bisa memakai get, post, dan lain-lain.
const router = express.Router();

router.get("/", sessionChecker.notLoggedIn, adminController.getIndex);

router.get("/cpu-usage", sessionChecker.notLoggedIn, adminController.getCPUUsage);

router.get("/memory-usage", adminController.getMemoryUsage);

router.get("/settings", sessionChecker.notLoggedIn, adminController.getSettings);

router.post("/settings/edit", sessionChecker.notLoggedIn,
adminController.postSettingsEdit);

router.get("/messages", sessionChecker.notLoggedIn,
adminController.getMessagesIndex);

router.get("/messages/delete/:id", sessionChecker.notLoggedIn,
```

```
adminController.getMessagesDelete);

router.get("/texts", sessionChecker.notLoggedIn, adminController.getTextsIndex);

router.post("/texts/edit", sessionChecker.notLoggedIn,
adminController.postTextsEdit);

router.get("/skills", sessionChecker.notLoggedIn, adminController.getSkillsIndex);

router.post("/skills/add", sessionChecker.notLoggedIn,
adminController.postSkillsAdd);

router.get("/skills/delete/:id", sessionChecker.notLoggedIn,
adminController.getSkillsDelete);

router.get("/services", sessionChecker.notLoggedIn,
adminController.getServicesIndex);

router.post("/services/add", sessionChecker.notLoggedIn,
adminController.postServicesAdd);

router.get("/services/delete/:id", sessionChecker.notLoggedIn,
adminController.getServicesDelete);

router.get("/carousels", sessionChecker.notLoggedIn,
adminController.getCarouselsIndex);

router.post("/carousels/upload", sessionChecker.notLoggedIn,
adminController.postCarouselsUpload);

router.get("/carousels/delete/:id", sessionChecker.notLoggedIn,
adminController.getCarouselsDelete);

router.get("/portfolios", sessionChecker.notLoggedIn,
adminController.getPortfoliosIndex);

router.post("/portfolios/upload", sessionChecker.notLoggedIn,
adminController.postPortfoliosUpload);

router.get("/portfolios/delete/:id", sessionChecker.notLoggedIn,
adminController.getPortfoliosDelete);

module.exports = router;
```

Isi file "routes/auth.js" dengan kode ini:

```
// script ini tugasnya adalah menghubungkan controllers/auth.js dengan route.

const express = require("express");
const authController = require("../controllers/auth");

// session checker digunakan untuk authorization
```

```
const sessionChecker = require("../middlewares/sessionchecker");

// buat objek router agar bisa memakai get, post, dan lain-lain.
const router = express.Router();

router.get("/login", sessionChecker.loggedIn, authController.getLogin);

router.post("/login", sessionChecker.loggedIn, authController.postLogin);

router.get("/register", authController.getRegister);

router.post("/register", authController.postRegister);

router.get("/logout", authController.getLogout);

module.exports = router;
```

Isi file "routes/index.js" dengan kode ini:

```
// script ini tugasnya adalah menghubungkan controllers/index.js dengan route.

const express = require("express");
const indexController = require("../controllers/index");

// buat objek router agar bisa memakai get, post, dan lain-lain.
const router = express.Router();

router.get("/", indexController.getIndex);

router.post("/send-message", indexController.postIndexSendMessage);

module.exports = router;
```

Membuat Subfolder "views"

Buatlah subfolder bernama "views", kemudian isi dengan file-file ini:

- error.ejs
- helper.ejs

Isi file "views/error.ejs" dengan kode ini:

```
<h1>
    <%= error.message %>
</h1>
<h2>
    <%= error.status %>
</h2>
<pre><%= error.stack %></pre>
```

Isi file "views/helper.ejs" dengan kode ini:

```
<%
generateExcerpt = function(wholeText) {
    var strLen = wholeText.length;
    if (strLen < 100) {
        return wholeText.substring(0, strLen).replace(/<[^(^(>)]*>?/gm, '');
    }
    return wholeText.substring(0, 100).replace(/<[^(^(>)]*>?/gm, '');
}

isOptionSelected = function (val, valDB) {
    return val == valDB ? "selected" : "";
}

pgnPrevious = function (currentPage) {
    let tmp = currentPage - 1;
    return tmp < 0 ? { index: 0, disabled: true } : { index: tmp, disabled: false
};
}

pgnNext = function (currentPage, totalPage) {
    let tmp = currentPage + 1;
    return tmp >= totalPage ? { index: currentPage, disabled: true } : { index:
tmp, disabled: false };
}
%>
```

Membuat Subfolder "views/admin"

Buatlah subfolder bernama "views/admin", kemudian isi dengan file-file ini:

- carousels.ejs
- carousels.js.ejs
- index.ejs
- index.js.ejs
- layout.ejs
- messages.ejs
- messages.js.ejs
- portfolios.ejs
- portfolios.js.ejs
- services.ejs
- services.js.ejs
- settings.ejs
- settings.js.ejs
- skills.ejs
- skills.js.ejs
- texts.ejs

- texts.js.ejs

Isi file "views/admin/carousels.ejs" dengan kode ini:

```
<div class="row mt-2">
  <div class="col-lg-12 col-md-12 col-sm-12 col-xs-12">
    <h3>Carousels</h3>
  </div>
</div>

<div class="row">
  <div class="col-12">
    <div class="card">
      <div class="card-header">
        <button id="btn-upload-file" class="btn btn-dark">Upload Image</button>
      </div>
      <hr>
      <div class="row">
        <% data.results.forEach((item, index)=> { %>
        <div class="col-md-4 mb-3">
          <div class="card">
            <img src="/<%= item.path %>" class="card-img-top img-fluid" alt="...">
            <hr>
            <div class="card-body">
              <h5 class="card-text">
                <%= item.title %>
              </h5>
              <p class="card-text">
                <%= item.description %>
              </p>
            </div>

            <div class="card-footer">
              <a href="/admin/carousels/delete/<%= item._id %>"><span class="badge
rounded-pill text-bg-danger">Delete</span></a>
            </div>
          </div>
        </div>
        <% }); %>
      </div>
    </div>
  </div>
</div>

<div class="modal fade" id="modal-add" tabindex="-1" aria-
labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">Upload Image</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-
label="Close"></button>
      </div>
```

```
      <form action="/admin/carousels/upload" method="post"
enctype="multipart/form-data">
        <div class="modal-body">
          <div class="form-group">
            <label for="title-add">Title:</label>
            <input type="text" class="form-control" id="title-add" name="title">
          </div>
          <div class="form-group">
            <label for="description-add">Description:</label>
            <input type="text" class="form-control" id="description-add"
name="description">
          </div>
          <div class="form-group">
            <input id="fl-file" type="file" name="upload" />
          </div>
        </div>
        <div class="modal-footer">
          <button type="button" class="btn btn-dark" data-bs-
dismiss="modal">Close</button>
          <button type="submit" class="btn btn-dark">Save</button>
        </div>
      </form>
    </div>
  </div>
</div>
```

Isi file "views/admin/carousels.js.ejs" dengan kode ini:

```
<script>
  $("#btn-upload-file").click(function() {
    let addModal = new bootstrap.Modal(document.getElementById("modal-add"), {
      backdrop: 'static',
      keyboard: false
    });
    addModal.show();
  });
</script>
```

Isi file "views/admin/index.ejs" dengan kode ini:

```
<div class="row mt-2">
  <div class="col-lg-12 col-md-12 col-sm-12 col-xs-12">
    <h3>Resource Usage</h3>
  </div>
</div>

<div class="row mt-2">
  <div class="col-md-6">
    <div class="card">
```

```
      <div class="card-header">
        <b>Current CPU Usage</b>
      </div>
      <div class="card-body">
        <canvas id="cpuUsageChartPie" width="100" height="50"></canvas>
      </div>
    </div>
  </div>
  <div class="col-md-6">
    <div class="card">
      <div class="card-header">
        <b>Current Memory Usage</b>
      </div>
      <div class="card-body">
        <canvas id="memoryUsageChartPie" width="100" height="50"></canvas>
      </div>
    </div>
  </div>
</div>
```

Isi file "views/admin/index.js.ejs" dengan kode ini:

```
<script>
    var configCPUUsagePie = {
        type: 'pie',
        data: {
            labels: [
                "CPU Used",
                "CPU Free"
            ],
            datasets: [{
                data: [100, 50],
                backgroundColor: [
                    "#FF6384",
                    "#36A2EB"
                ],
                hoverBackgroundColor: [
                    "#FF6384",
                    "#36A2EB"
                ]
            }]
        }
    };

    var cpuUsageChartPie = new Chart(
        document.getElementById("cpuUsageChartPie").getContext("2d"),
        configCPUUsagePie
    );

    setInterval(function () {
        $.get("/admin/cpu-usage", function (data) {
            configCPUUsagePie.data.datasets[0].data[0] = data.value;
```

```
            configCPUUsagePie.data.datasets[0].data[1] = (100 - data.value);
            cpuUsageChartPie.update();
        });
    }, 1000);

    var configMemoryUsagePie = {
        type: 'pie',
        data: {
            labels: [
                "Memory Used",
                "Memory Free"
            ],
            datasets: [{
                data: [100, 50],
                backgroundColor: [
                    "#FF6384",
                    "#36A2EB"
                ],
                hoverBackgroundColor: [
                    "#FF6384",
                    "#36A2EB"
                ]
            }]
        }
    };

    var memoryUsageChartPie = new Chart(
        document.getElementById("memoryUsageChartPie").getContext("2d"),
        configMemoryUsagePie
    );

    setInterval(function () {
        let self = this;
        $.get("/admin/memory-usage", function (data) {
            configMemoryUsagePie.data.datasets[0].data[0] = data.value;
            configMemoryUsagePie.data.datasets[0].data[1] = 100 - data.value;
            memoryUsageChartPie.update();
        });

    }, 1000);
</script>
```

Isi file "views/admin/layout.ejs" dengan kode ini:

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">
```

```
  <meta name="description" content="">
  <meta name="author" content="">

  <link rel="shortcut icon" type="image/png" href="/public/favicon.png" />

  <title>
    <%= data.text.siteTitle %>
  </title>

  <link rel="stylesheet" href="/public/vendor/bootstrap/css/bootstrap.min.css" />
  <link rel="stylesheet" href="/public/vendor/icons/font/bootstrap-icons.css" />
  <link rel='stylesheet' href="/public/css/bootstrap-mod.css" />
</head>

<body>
  <div id="sidebar" class="sidebar bg-dark">
    <div class="sidebar-brand">
      <img class="img-fluid sidebar-brand-img mx-auto d-block"
src="/public/img/icon.png" alt="icon" width="200" height="200">
      <h3 class="text-white text-center">Company Profile</h3>
    </div>
    <hr class="bg-light">
    <ul class="sidebar-ul">
      <li class="sidebar-li">
        <a class="btn btn-outline-light w-100 btn-mobile" href="/admin"><i
class="bi bi-speedometer float-start"></i><span>
            Dashboard</span></a>
      </li>
      <li class="sidebar-li">
        <a class="btn btn-outline-light w-100 btn-mobile" href="/admin/messages">
<i class="bi bi-folder float-start"></i>
          <span>
            Messages</span></a>
      </li>
      <li class="sidebar-li">
        <a class="btn btn-outline-light w-100 btn-mobile" href="/admin/texts"><i
class="bi bi-folder float-start"></i>
          <span>
            Texts</span></a>
      </li>
      <li class="sidebar-li">
        <a class="btn btn-outline-light w-100 btn-mobile" href="/admin/skills"><i
class="bi bi-folder float-start"></i>
          <span>
            Skills</span></a>
      </li>
      <li class="sidebar-li">
        <a class="btn btn-outline-light w-100 btn-mobile" href="/admin/services">
<i class="bi bi-folder float-start"></i>
          <span>
            Services</span></a>
      </li>
      <li class="sidebar-li">
        <a class="btn btn-outline-light w-100 btn-mobile" href="/admin/carousels">
```

```
<i class="bi bi-upload float-start"></i>
          <span>
            Carousels</span></a>
      </li>
      <li class="sidebar-li">
        <a class="btn btn-outline-light w-100 btn-mobile"
href="/admin/portfolios"><i class="bi bi-upload float-start"></i>
          <span>
            Portfolios</span></a>
      </li>
      <li class="sidebar-li">
        <a class="btn btn-outline-light w-100 btn-mobile" href="/admin/settings">
<i class="bi bi-gear-fill float-start"></i>
          <span>
            Settings</span></a>
      </li>
    </ul>
    <hr class="bg-light">
    <!-- <ul class="sidebar-ul">
            <li class="sidebar-li">
                <a class="btn btn-outline-light w-100 btn-mobile"
href="javascript:void(0);"><i class="bi bi-plug-fill float-left"></i>
                    <span>Plugins</span></a>
            </li>
        </ul> -->
  </div>

  <div class="main">
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
      <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-
bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-
expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>

      <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav ms-auto">
        </ul>
        <ul class="navbar-nav">
          <li class="nav-item active">
            <a class="nav-link" href="/admin">Home <span class="sr-only">(current)
</span></a>
          </li>
          <li class="nav-item active">
            <a class="nav-link" href="/" target="_blank">Preview</a>
          </li>
          <li class="nav-item dropdown active">
            <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-bs-toggle="dropdown" aria-expanded="false">
              Profile
            </a>
            <div class="dropdown-menu dropdown-menu-end" aria-
labelledby="navbarDropdown">
              <a class="dropdown-item" href="/admin/settings">Edit Profile</a>
```

```
              <div class="dropdown-divider"></div>
              <a class="dropdown-item" href="/auth/logout"><i class="bi bi-box-
arrow-right float-left"></i>Logout</a>
            </div>
          </li>
        </ul>
      </div>
    </nav>

    <div class="container-fluid">
      <% include ("../helper") %>

      <%- include("../" + child, {data: data}); %>
    </div>
  </div>

  <script src="/public/vendor/jquery/jquery.min.js"></script>
  <script src="/public/vendor/jquery-sortable/jquery-sortable.js"></script>
  <script src="/public/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>
  <script src="/public/vendor/chart.js/Chart.bundle.min.js"></script>
  <script src="/public/js/admin.js"></script>
  <%- include("../" + clientScript, {data: data}); %>
</body>

</html>
```

Isi file "views/admin/messages.ejs" dengan kode ini:

```
<div class="row mt-2">
  <div class="col-lg-12 col-md-12 col-sm-12 col-xs-12">
    <h3>Messages</h3>
  </div>
</div>

<div class="row">
  <div class="col-12">
    <div class="card">
      <div class="card-header">
        <% if (data.results <= 0) { %>
        <h4>No Messages</h4>
        <% } else { %>
        <h4>You've Got Messages</h4>
        <% } %>
      </div>
      <hr>
      <div class="row">
        <% data.results.forEach((item, index)=> { %>
        <div class="col-md-4 mb-3">
          <div class="card mx-3">
            <div class="card-body">
              <h5 class="card-text">
                <%= item.name %>
```

```
              </h5>

              <h5 class="card-text">
                <%= item.email %>
              </h5>

              <p><%= item.message %></p>
            </div>

            <div class="card-footer">
              <a href="/admin/messages/delete/<%= item._id %>"><span class="badge
rounded-pill text-bg-danger">Delete</span></a>
            </div>
          </div>
        </div>
        <% }); %>
      </div>
    </div>
  </div>
</div>
```

Isi file "views/admin/messages.js.ejs" dengan kode ini:

```
<!-- kosong -->
```

Isi file "views/admin/portfolios.ejs" dengan kode ini:

```
<div class="row mt-2">
  <div class="col-lg-12 col-md-12 col-sm-12 col-xs-12">
    <h3>Portfolios</h3>
  </div>
</div>

<div class="row">
  <div class="col-12">
    <div class="card">
      <div class="card-header">
        <button id="btn-upload-file" class="btn btn-dark">Upload Image</button>
      </div>
      <hr>
      <div class="row">
        <% data.results.forEach((item, index)=> { %>
        <div class="col-md-4 mb-3">
          <div class="card">
            <img src="/<%= item.path %>" class="card-img-top img-fluid" alt="...">
            <hr>
            <div class="card-body">
              <h5 class="card-text">
                <%= item.title %>
```

```
              </h5>
            </div>

            <div class="card-footer">
              <a href="/admin/portfolios/delete/<%= item._id %>"><span
class="badge rounded-pill text-bg-danger">Delete</span></a>
            </div>
          </div>
        </div>
        <% }); %>
      </div>
    </div>
  </div>
</div>

<div class="modal fade" id="modal-add" tabindex="-1" aria-
labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">Upload Image</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-
label="Close"></button>
      </div>
      <form action="/admin/portfolios/upload" method="post"
enctype="multipart/form-data">
        <div class="modal-body">
          <div class="form-group">
            <label for="title-add">Title:</label>
            <input type="text" class="form-control" id="title-add" name="title">
          </div>
          <div class="form-group">
            <input id="fl-file" type="file" name="upload" />
          </div>
        </div>
        <div class="modal-footer">
          <button type="button" class="btn btn-dark" data-bs-
dismiss="modal">Close</button>
          <button type="submit" class="btn btn-dark">Save</button>
        </div>
      </form>
    </div>
  </div>
</div>
```

Isi file "views/admin/portfolios.js.ejs" dengan kode ini:

```
<script>
  $("#btn-upload-file").click(function() {
    let addModal = new bootstrap.Modal(document.getElementById("modal-add"), {
      backdrop: 'static',
      keyboard: false
```

```
    });
    addModal.show();
  });
</script>
```

Isi file "views/admin/services.ejs" dengan kode ini:

```
<div class="row mt-2">
  <div class="col-lg-12 col-md-12 col-sm-12 col-xs-12">
    <h3>Services</h3>
  </div>
</div>

<div class="row">
  <div class="col-12">
    <div class="card">
      <div class="card-header">
        <button id="btn-upload-file" class="btn btn-dark">Add Service</button>
      </div>
      <hr>
      <div class="row">
        <% data.results.forEach((item, index)=> { %>
        <div class="col-md-4 mb-3">
          <div class="card">
            <div class="text-center">
              <%- item.svg %>
            </div>
            <hr>
            <div class="card-body">
              <h5 class="card-text">
                <%= item.title %>
              </h5>

              <p><%= item.description %></p>
            </div>

            <div class="card-footer">
              <a href="/admin/services/delete/<%= item._id %>"><span class="badge
rounded-pill text-bg-danger">Delete</span></a>
            </div>
          </div>
        </div>
        <% }); %>
      </div>
    </div>
  </div>
</div>

<div class="modal fade" id="modal-add" tabindex="-1" aria-
labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
```

```
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">Add New Service</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-
label="Close"></button>
      </div>
      <form action="/admin/services/add" method="post">
        <div class="modal-body">
          <div class="form-group">
            <label for="title-add">Title:</label>
            <input type="text" class="form-control" id="title-add" name="title">
          </div>
          <div class="form-group">
            <label for="description-add">Description:</label>
            <input type="text" class="form-control" id="description-add"
name="description">
          </div>
          <div class="form-group">
            <label for="svg-add">Copy Paste SVG Here (Must be Valid):</label>
            <textarea class="form-control" id="svg-add" name="svg"></textarea>
          </div>
        </div>
        <div class="modal-footer">
          <button type="button" class="btn btn-dark" data-bs-
dismiss="modal">Close</button>
          <button type="submit" class="btn btn-dark">Save</button>
        </div>
      </form>
    </div>
  </div>
</div>
```

Isi file "views/admin/services.js.ejs" dengan kode ini:

```
<script>
  $("#btn-upload-file").click(function() {
    let addModal = new bootstrap.Modal(document.getElementById("modal-add"), {
      backdrop: 'static',
      keyboard: false
    });
    addModal.show();
  });
</script>
```

Isi file "views/admin/settings.ejs" dengan kode ini:

```
<div class="row mt-2">
  <div class="col-lg-12 col-md-12 col-sm-12 col-xs-12">
    <h3>Settings</h3>
  </div>
```

```
</div>

<div class="row">
  <div class="col-lg-12 col-md-12 col-sm-12 col-xs-12">
    <div class="card">
      <div class="card-header">
        <h3 class="">Account Settings</h3>
      </div>
      <% data.errors.forEach(function(result){ %>
      <div class="alert alert-danger" role="alert">
        <%= result.message %>
      </div>
      <% }); %>
      <div class="card-body">
        <form action="/admin/settings/edit" method="POST">
          <div class="form-group">
            <label for="email">Email:</label>
            <input id="email" name="email" value="<%= data.adminEmail %>"
type="email" class="form-control" placeholder="Enter Email">
          </div>
          <div class="form-group">
            <label for="password">Password:</label>
            <div class="input-group mb-3">
              <input id="password" name="password" value="" type="password"
class="form-control" placeholder="Enter Password">
              <div class="input-group-append">
                <button onclick="toggleShowHidePassword();" class="btn btn-
outline-secondary" type="button" id="btn-show-hide"><i class="bi bi-eye-slash"
aria-hidden="true"></i></button>
              </div>
            </div>
          </div>
          <div class="form-group">
            <button type="submit" class="btn btn-dark">Update</button>
          </div>
        </form>
      </div>
    </div>
  </div>
</div>
```

Isi file "views/admin/settings.js.ejs" dengan kode ini:

```
<script>
    function toggleShowHidePassword() {
        let passwordElement = $("#password");
        if (passwordElement.attr("type") === "password") {
            passwordElement.attr("type", "text");
            $('#btn-show-hide i').removeClass("bi-eye-slash");
            $('#btn-show-hide i').addClass("bi-eye");
        } else {
            passwordElement.attr("type", "password");
```

```
            $('#btn-show-hide i').addClass("bi-eye-slash");
            $('#btn-show-hide i').removeClass("bi-eye");
        }
    };
</script>
```

Isi file "views/admin/skills.ejs" dengan kode ini:

```
<div class="row mt-2">
  <div class="col-lg-12 col-md-12 col-sm-12 col-xs-12">
    <h3>Skills</h3>
  </div>
</div>

<div class="row">
  <div class="col-12">
    <div class="card">
      <div class="card-header">
        <button id="btn-upload-file" class="btn btn-dark">Add Skill</button>
      </div>
      <hr>
      <div class="row">
        <div class="col-lg-12 col-md-12 col-sm-12 col-xs-12">
          <div class="table-responsive mt-2">
            <table class="table table-hover">
              <tr>
                <th>IDX</th>
                <th>Title</th>
                <th>Level</th>
                <th>Actions</th>
              </tr>
              <% data.results.forEach((item, index)=> { %>
              <tr>
                <td>
                  <%= ++index %>
                </td>
                <td>
                  <%= item.title %>
                </td>
                <td>
                  <%= item.level %>
                </td>
                <td>
                  <a href="/admin/Skills/delete/<%= item._id %>" class="badge
text-bg-danger">Delete</a>
                </td>
              </tr>
              <% }); %>
            </table>
          </div>
        </div>
      </div>
```

```
    </div>
  </div>
</div>

<div class="modal fade" id="modal-add" tabindex="-1" aria-
labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">Add New Service</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-
label="Close"></button>
      </div>
      <form action="/admin/skills/add" method="post">
        <div class="modal-body">
          <div class="form-group">
            <label for="title-add">Title</label>
            <input type="text" class="form-control" id="title-add" name="title">
          </div>
          <div class="form-group">
            <label for="level-add">Level (0 to 100)</label>
            <input type="text" class="form-control" id="level-add" name="level">
          </div>
        </div>
        <div class="modal-footer">
          <button type="button" class="btn btn-dark" data-bs-
dismiss="modal">Close</button>
          <button type="submit" class="btn btn-dark">Save</button>
        </div>
      </form>
    </div>
  </div>
</div>
```

Isi file "views/admin/skills.js.ejs" dengan kode ini:

```
<script>
  $("#btn-upload-file").click(function() {
    let addModal = new bootstrap.Modal(document.getElementById("modal-add"), {
      backdrop: 'static',
      keyboard: false
    });
    addModal.show();
  });
</script>
```

Isi file "views/admin/texts.ejs" dengan kode ini:

```
<div class="row mt-2">
  <div class="col-lg-12 col-md-12 col-sm-12 col-xs-12">
    <h3>Texts</h3>
  </div>
</div>

<div class="row">
  <div class="col-lg-12 col-md-12 col-sm-12 col-xs-12">
    <div class="card">
      <div class="card-body">
        <form action="/admin/texts/edit" method="POST">
          <div class="form-group">
            <label for="tx-site-title">Site Title:</label>
            <input id="tx-site-title" name="siteTitle" type="text" value="<%=
data.text.siteTitle %>" class="form-control">
          </div>

          <div class="form-group">
            <label for="tx-site-seo-title">Site SEO Title:</label>
            <input id="tx-site-seo-title" name="siteSEOTitle" type="text" value="
<%= data.text.siteSEOTitle %>" class="form-control">
          </div>

          <div class="form-group">
            <label for="tx-site-description">Site Description:</label>
            <input id="tx-site-description" name="siteDescription" type="text"
value="<%= data.text.siteDescription %>" class="form-control">
          </div>

          <div class="form-group">
            <label for="tx-site-seo-description">Site SEO Description:</label>
            <input id="tx-site-seo-description" name="siteSEODescription"
type="text" value="<%= data.text.siteSEODescription %>" class="form-control">
          </div>

          <div class="form-group">
            <label for="tx-site-about-subheading">About Subheading:</label>
            <input id="tx-site-about-subheading" name="aboutSubHeading"
type="text" value="<%= data.text.aboutSubHeading %>" class="form-control">
          </div>

          <div class="form-group">
            <label for="tx-site-about-subtext">About Subtext:</label>
            <input id="tx-site-about-subtext" name="aboutSubText" type="text"
value="<%= data.text.aboutSubText %>" class="form-control">
          </div>

          <div class="form-group">
            <label for="tx-site-info-subheading">Info Subheading:</label>
            <input id="tx-site-info-subheading" name="infoSubHeading" type="text"
value="<%= data.text.infoSubHeading %>" class="form-control">
          </div>
```

```
          <div class="form-group">
            <label for="tx-site-address">Address:</label>
            <input id="tx-site-address" name="address" type="text" value="<%=
data.text.address %>" class="form-control">
          </div>

          <div class="form-group">
            <label for="tx-site-phone">Phone:</label>
            <input id="tx-site-phone" name="phone" type="text" value="<%=
data.text.phone %>" class="form-control">
          </div>

          <div class="form-group">
            <label for="tx-site-email">Email:</label>
            <input id="tx-site-email" name="email" type="text" value="<%=
data.text.email %>" class="form-control">
          </div>

          <div class="space-15"></div>

          <div class="form-group">
            <button type="submit" class="btn btn-dark">Update</button>
          </div>
        </form>
      </div>
    </div>
  </div>
</div>
```

Isi file "views/admin/texts.js.ejs" dengan kode ini:

```
<!-- kosong -->
```

Membuat Subfolder "views/auth"

Buatlah subfolder bernama "views/auth", kemudian isi dengan file-file ini:

- layout.ejs
- login.ejs
- login.js.ejs
- register.ejs
- register.js.ejs

Isi file "views/auth/layout.ejs" dengan kode ini:

```
<!DOCTYPE html>
<html>

<head>
```

```
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">
  <meta name="description" content="">
  <meta name="author" content="">

  <link rel="shortcut icon" type="image/png" href="/public/favicon.png" />

  <title>
    <%= data.text.siteTitle %>
  </title>

  <link rel="stylesheet" href="/public/vendor/bootstrap/css/bootstrap.min.css" />
  <link rel="stylesheet" href="/public/vendor/icons/font/bootstrap-icons.css" />
</head>

<body>
  <div id="app" class="container">
    <% include ("../helper") %>
    <%- include("../" + child, {data: data}); %>
  </div>
  <script src="/public/vendor/jquery/jquery.min.js"></script>
  <script src="/public/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>
  <script src="/public/vendor/chart.js/Chart.bundle.min.js"></script>
  <script src="/public/js/auth.js"></script>
  <%- include("../" + clientScript, {data: data}); %>
</body>

</html>
```

Isi file "views/auth/login.ejs" dengan kode ini:

```
<div class="row">
  <div class="col-md">
    <div class="modal fade" id="loginModal" tabindex="-1" role="dialog" aria-
labelledby="loginModalTitle" aria-hidden="true">
      <div class="modal-dialog modal-dialog-centered" role="document">
        <div class="modal-content">
          <div class="modal-header">
            <h5 class="modal-title" id="loginModalTitle">
              <%= data.text.siteTitle %> - Log In
            </h5>
          </div>
          <% data.errors.forEach(function(result){ %>
          <div class="alert alert-danger" role="alert">
            <%= result.message %>
          </div>
          <% }); %>
          <form action="/auth/login" method="POST">
            <div class="modal-body">
              <div class="form-group">
```

```
                <label for="exampleInputEmail1">Email address</label>
                <input type="email" name="email" class="form-control"
id="exampleInputEmail1" aria-describedby="emailHelp" placeholder="Enter email">
              </div>
              <div class="form-group">
                <label for="exampleInputPassword1">Password</label>
                <input type="password" name="password" class="form-control"
id="exampleInputPassword1" placeholder="Password">
              </div>
            </div>
            <div class="modal-footer">
              <a class="btn btn-dark" href="/">Back to Home</a>
              <button type="submit" class="btn btn-dark">Log In</button>
            </div>
          </form>
        </div>
      </div>
    </div>
  </div>
</div>
```

Isi file "views/auth/login.js.ejs" dengan kode ini:

```
<script>
    showAuthModal("loginModal");
</script>
```

Isi file "views/auth/register.ejs" dengan kode ini:

```
<div class="row">
  <div class="col-md">
    <div class="modal fade" id="registerModal" tabindex="-1" role="dialog" aria-
labelledby="registerModalTitle" aria-hidden="true">
      <div class="modal-dialog modal-dialog-centered" role="document">
        <div class="modal-content">
          <div class="modal-header">
            <h5 class="modal-title" id="registerModalTitle">
              <%= data.text.siteTitle %> - Register
            </h5>
          </div>
          <% data.errors.forEach(function(result){ %>
          <div class="alert alert-danger" role="alert">
            <%= result.message %>
          </div>
          <% }); %>
          <form action="/auth/register" method="POST">
            <div class="modal-body">
              <div class="form-group">
                <label for="tx-email">Email address</label>
```

```
                <input type="email" name="email" class="form-control" id="tx-
email" aria-describedby="emailHelp" placeholder="Enter email">
              </div>
              <div class="form-group">
                <label for="tx-username">Username</label>
                <input type="text" name="name" class="form-control" id="tx-
username" aria-describedby="emailHelp" placeholder="Enter Username">
              </div>
              <div class="form-group">
                <label for="tx-password">Password</label>
                <input type="password" name="password" class="form-control"
id="tx-password" placeholder="Password">
              </div>
              <div class="form-group">
                <label for="tx-password-repeat" class="form-label">Repeat
                  Password</label>
                <input type="password" name="password_repeat" class="form-control"
id="tx-password-repeat" placeholder="Masukkan password lagi">
              </div>
            </div>
            <div class="modal-footer">
              <a class="btn btn-dark" href="/">Back to Home</a>
              <button type="submit" class="btn btn-dark">Register</button>
            </div>
          </form>
        </div>
      </div>
    </div>
  </div>
</div>
```

Isi file "views/auth/register.js.ejs" dengan kode ini:

```
<script>
    showAuthModal("registerModal");
</script>
```

Membuat Subfolder "views/index"

Buatlah subfolder bernama "views/index", kemudian isi dengan file-file ini:

- index.ejs

Isi file "views/index/index.ejs" dengan kode ini:

```
<!DOCTYPE html>
<html>

<head>
```

```
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no">
  <meta name="description" content="<%= text.siteSEODescription %>">
  <meta name="author" content="">

  <link rel="shortcut icon" type="image/png" href="/public/img/favicon.png" />

  <title><%= text.siteSEOTitle %></title>

  <link rel="stylesheet" href="/public/vendor/bootstrap/css/bootstrap.min.css" />
  <link rel="stylesheet" href="/public/vendor/icons/font/bootstrap-icons.css" />
  <link rel='stylesheet' href="/public/css/style.css" />
</head>

<body>
  <div id="app" class="container-fluid px-0">
    <nav class="navbar navbar-expand-lg fixed-top bg-dark" data-bs-theme="dark">
      <div class="container">
        <a class="navbar-brand" href="#"><%= text.siteTitle %></a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"
aria-expanded="false" aria-label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarSupportedContent">
          <ul class="navbar-nav ms-auto mb-2 mb-lg-0">
            <li class="nav-item">
              <a class="nav-link active" href="#home">Home</a>
            </li>
            <li class="nav-item">
              <a class="nav-link active" href="#about">About</a>
            </li>
            <li class="nav-item">
              <a class="nav-link active" href="#services">Services</a>
            </li>
            <li class="nav-item">
              <a class="nav-link active" href="#portfolios">Portfolios</a>
            </li>
            <li class="nav-item">
              <a class="nav-link active" href="#contact">Contact</a>
            </li>
          </ul>
        </div>
      </div>
    </nav>


    <section id="home" class="home scrollspy">
      <div id="main-carousel" class="carousel slide" data-bs-ride="false">
        <% if (carousels.length <= 0){ %>

        <div class="carousel-indicators">
```

```
          <button type="button" data-bs-target="#main-carousel" data-bs-slide-
to="0" class="active" aria-current="true" aria-label="Slide 1"></button>
          <button type="button" data-bs-target="#main-carousel" data-bs-slide-
to="1" aria-label="Slide 2"></button>
          <button type="button" data-bs-target="#main-carousel" data-bs-slide-
to="2" aria-label="Slide 3"></button>
        </div>
        <div class="carousel-inner">
          <div class="carousel-item active">
            <img src="/public/img/slider/1.jpg" class="d-block" alt="...">
            <div class="carousel-caption d-none d-md-block bg-dark">
              <h5>Slide Label 1</h5>
              <p>Slide description 1.</p>
            </div>
          </div>
          <div class="carousel-item">
            <img src="/public/img/slider/2.jpg" class="d-block" alt="...">
            <div class="carousel-caption d-none d-md-block bg-dark">
              <h5>Slide Label 2</h5>
              <p>Slide description 2.</p>
            </div>
          </div>
          <div class="carousel-item">
            <img src="/public/img/slider/3.jpg" class="d-block" alt="...">
            <div class="carousel-caption d-none d-md-block bg-dark">
              <h5>Slide Label 3</h5>
              <p>Slide description 3.</p>
            </div>
          </div>
        </div>
        <button class="carousel-control-prev" type="button" data-bs-target="#main-
carousel" data-bs-slide="prev">
          <span class="carousel-control-prev-icon" aria-hidden="true"></span>
          <span class="visually-hidden">Previous</span>
        </button>
        <button class="carousel-control-next" type="button" data-bs-target="#main-
carousel" data-bs-slide="next">
          <span class="carousel-control-next-icon" aria-hidden="true"></span>
          <span class="visually-hidden">Next</span>
        </button>

        <% } else { %>

        <div class="carousel-indicators">
          <% carousels.forEach((item, index)=> { %>
          <button type="button" data-bs-target="#main-carousel" data-bs-slide-to="
<%= index %>" <% if( index === 0) { %> class="active" <% } %> aria-current="true"
aria-label="<%= item.title %>">
          </button>
          <% }); %>
        </div>

        <div class="carousel-inner">
          <% carousels.forEach((item, index)=> { %>
```

```
          <div <% if( index === 0) { %> class="carousel-item active" <% } else {
%> class="carousel-item" <% } %>>
            <img src="/<%= item.path %>" class="d-block" alt="...">
            <div class="carousel-caption d-none d-md-block bg-dark">
              <h5><%= item.title %></h5>
              <p><%= item.description %></p>
            </div>
          </div>
          <% }); %>
        </div>

        <button class="carousel-control-prev" type="button" data-bs-target="#main-
carousel" data-bs-slide="prev">
          <span class="carousel-control-prev-icon" aria-hidden="true"></span>
          <span class="visually-hidden">Previous</span>
        </button>
        <button class="carousel-control-next" type="button" data-bs-target="#main-
carousel" data-bs-slide="next">
          <span class="carousel-control-next-icon" aria-hidden="true"></span>
          <span class="visually-hidden">Next</span>
        </button>
        <% } %>
      </div>
    </section>

    <section id="about" class="about scrollspy text-bg-dark px-0 pt-5 pb-5">
      <div class="container">
        <h1 class="center">About Us</h1>
        <hr>
        <div class="row">
          <div class="col-md-6 col-sm-12">
            <h3 class="pb-2"><%= text.aboutSubHeading %></h3>
            <blockquote><%= text.aboutSubText %></blockquote>
          </div>
          <div class="col-md-6 col-sm-12">
            <h3 class="pb-2">Skills</h3>

            <% skills.forEach((item, index)=> { %>
            <h5><%= item.title %></h5>
            <div class="progress" role="progressbar" aria-label="Default striped
example" aria-valuenow="<%= item.level %>" aria-valuemin="0" aria-valuemax="100">
              <div class="progress-bar progress-bar-striped bg-dark" style="width:
<%= item.level %>%"></div>
            </div>
            <hr>
            <% }); %>
          </div>
        </div>
      </div>
    </section>

    <section id="services" class="services orange lighten-3 scrollspy pt-5 pb-5">
      <div class="container">
        <h1 class="center">Services</h1>
```

```
        <hr>
        <div class="row d-flex justify-content-center">
          <% services.forEach((item, index)=> { %>
          <div class="col-md-4 col-sm-12">
            <div class="card-panel center">
              <div class="text-center">
                <%- item.svg %>
              </div>
              <hr>
              <h5 class="text-center"><%= item.title %></h5>
              <p><%= item.description %></p>
            </div>
          </div>
          <% }); %>
        </div>
      </div>
    </section>

    <section id="portfolios" class="portfolios scrollspy text-bg-dark pt-5 pb-5">
      <div class="container">
        <h1 class="center">Portfolios</h1>
        <hr>
        <div class="row d-flex justify-content-center">
          <% portfolios.forEach((item, index)=> { %>
          <div class="col-md-4 col-sm-12 mb-2 d-flex justify-content-center text-
center">
            <div class="card bg-dark" style="width: 18rem; border: none;">
              <img src="/<%= item.path %>" class="card-img-top" alt="...">
              <div class="card-body">
                <h5 class="card-title text-center bg-light pt-2 pb-2"><%=
item.title %></h5>
              </div>
            </div>
          </div>
          <% }); %>
        </div>
      </div>
    </section>

    <section id="contact" class="contact orange lighten-3 scrollspy pt-5 pb-5">
      <div class="container">
        <h1 class="center">Contact Us</h1>
        <hr>
        <div class="row d-flex">
          <div class="col-md-4 col-sm-12 flex-grow-1 mb-2">
            <div class="card p-2" style="height: 100%;">
              <div class="text-center">
                <svg xmlns="http://www.w3.org/2000/svg" width="110" height="110"
fill="currentColor" class="bi bi-info-square-fill" viewBox="0 0 16 16">
                  <path d="M0 2a2 2 0 0 1 2-2h12a2 2 0 0 1 2 2v12a2 2 0 0 1-2
2H2a2 2 0 0 1-2-2V2zm8.93 4.588-
2.29.287-.082.38.45.083c.294.07.352.176.288.469l-.738 3.468c-.194.897.105
1.319.808 1.319.545 0 1.178-.252
1.465-.598l.088-.416c-.2.176-.492.246-.686.246-.275 0-.375-.193-.304-.533L8.93
```

```
6.588zM8 5.5a1 1 0 1 0 0-2 1 1 0 0 0 0 2z" />
                </svg>
              </div>
              <div style="margin-top: 60px"></div>
              <h5 class="text-center">
                <%= text.infoSubHeading %>
              </h5>
              <hr>
              <ul class="collection">
                <li class="collection-item">
                  <%= text.address %>
                </li>
                <li class="collection-item">
                  <%= text.phone %>
                </li>
                <li class="collection-item">
                  <%= text.email %>
                </li>
              </ul>
            </div>
          </div>

          <div class="col-md-8 col-sm-12 flex-grow-1 mb-2">
            <div class="card p-2" style="height: 100%;">
              <h5 class="text-center">
                Send Us a Message
              </h5>
              <hr>
              <form action="/send-message" method="post">
                <div class="row">
                  <div class="input-field col-md-12 col-sm-12">
                    <label for="name" class="form-label">Name</label>
                    <input id="name" name="name" type="text" class="form-control">
                  </div>
                </div>
                <div class="row">
                  <div class="input-field col-md-12 col-sm-12">
                    <label for="email" class="form-label">Email</label>
                    <input id="email" name="email" type="email" class="form-
control">
                  </div>
                </div>
                <div class="row mb-2">
                  <div class="input-field col-md-12 col-sm-12">
                    <label for="message" class="form-label">Message</label>
                    <textarea id="message" name="message" class="form-control"
rows="3"></textarea>
                  </div>
                </div>
                <div class="row">
                  <div class="input-field col-md-12 col-sm-12">
                    <button type="submit" class="btn btn-outline-dark w-100">
                      Send
                    </button>
```

```
                  </div>
                </div>
              </form>
            </div>
          </div>
        </div>
      </div>
    </section>

    <footer id="footer" class="footer text-bg-dark text-center">
      <p class="center white-text">Copyright &copy <span id="current-
year">2025</span> <%= text.siteTitle %></p>
      <p><a href="/auth/login">Login</a> | <a href="/auth/register">Register</a>
</p>
    </footer>
  </div>
  <script src="/public/vendor/jquery/jquery.min.js"></script>
  <script src="/public/vendor/bootstrap/js/bootstrap.bundle.min.js"></script>
  <script>
    $(document).ready(function() {
      console.log(location.pathname)
      $(`a[href*='${location.pathname}']`).addClass("active");

      $("#current-year").text(new Date().getFullYear().toString())
    });
  </script>
</body>

</html>
```

Pembuatan Struktur Project Selesai

Sekarang, Anda telah membangun struktur file dan folder project.

Langkah selanjutnya Adalah mencoba menjalankan aplikasi ini.

Sebenarnya di bagian ini, langkahnya sama dengan bagian "Cara Mencoba Kode Ini".

Jadi, silakan baca bagian tersebut dan praktekkan.

Pembahasan

File "package.json"

File "package.json" dibuat secara otomatis saat kita menjalankan perintah npm init.

Penambahan parameter -y pada perintah npm init -y menandakan bahwa kita menerima semua jawaban dari pertanyaan npm init sebagai "yes" atau "iya".

Karena kita mendapat file "package.json" default dari perintah tersebut, maka pada bagian "Langkah-Langkah", kita me-replace "package.json" default yang di-generate dengan versi kita sendiri.

Yaitu yang ini:

```
{
    "name": "company_profile",
    "version": "2024.05.28",
    "main": "app.js",
    "private": true,
    "scripts": {
        "start": "node app.js",
        "dev": "nodemon -e js,ejs,html -w . -w public -w views -w routes -w models
app.js --debug",
        "db:dev:refresh": "npx knex migrate:rollback --env development --knexfile
./databases/knexfile.js && npx knex migrate:latest --env development --knexfile
./databases/knexfile.js",
        "db:stg:refresh": "npx knex migrate:rollback --env staging --knexfile
./databases/knexfile.js && npx knex migrate:latest --env staging --knexfile
./databases/knexfile.js",
        "db:prod:refresh": "npx knex migrate:rollback --env production --knexfile
./databases/knexfile.js && npx knex migrate:latest --env production --knexfile
./databases/knexfile.js"
    },
    "dependencies": {
        "bcryptjs": "^2.4.3",
        "connect-flash": "^0.1.1",
        "cookie-parser": "~1.4.6",
        "dotenv": "^16.0.3",
        "ejs": "~3.1.9",
        "express": "~4.18.2",
        "express-session": "^1.17.3",
        "http-errors": "~2.0.0",
        "joi": "^17.9.1",
        "knex": "^2.4.2",
        "morgan": "~1.10.0",
        "multer": "^1.4.3",
        "mysql2": "^3.2.1",
        "node-os-utils": "^1.3.7",
        "session-file-store": "^1.5.0",
        "sqlite3": "^5.1.6",
        "uuid": "^9.0.0"
    },
    "devDependencies": {
        "nodemon": "^2.0.22"
    }
}
```

Berikut ini adalah deskripsi dari property-property file "package.json" di atas:

- "name" adalah nama project kita.
- "version" adalah versi project kita.
- "main" adalah script yang akan dijalankan Electron saat pertama kali atau bootstrapper-nya.
- "private" yang berisi "true" artinya project ini tidak akan dipublikasi di npm.
- "scripts" adalah daftar npm script (misal npm run dev) yang tersedia.
- "dependencies" adalah daftar package npm yang disertakan di production.

- "devDependencies" adalah daftar dependencies yang kita install dengan perintah npm install [nama_package] --save-dev.

Script "start"

Script ini akan menjalankan perintah ini di command line:

```
node app.js
```

Artinya, file "app.js" akan dijalankan sebagai input dari Node JS.

Script "dev"

Script ini akan menjalankan perintah ini di command line:

```
nodemon -e js,ejs,html -w . -w public -w views -w routes -w models app.js --debug
```

Artinya, file "app.js" akan dijalankan dengan Nodemon.

Dengan nodemon maka project ini akan di-restart secara otomatis jika file-file atau folder-folder yang memenuhi prasyarat ini:

```
-e js,ejs,html -w . -w public -w views -w routes -w models app.js --debug
```

Telah berubah.

Adapun arti dari prasyarat tadi adalah:

- file ber-extension js, ejs, dan html
- folder "public", "views", "routes", "models".

Script "db:dev:refresh"

Script ini akan menjalankan perintah ini di command line:

```
npx knex migrate:rollback --env development --knexfile ./databases/knexfile.js &&
npx knex migrate:latest --env development --knexfile ./databases/knexfile.js
```

Artinya, database yang digunakan dalam environment development di project ini akan di-rollback (dihapus) kemudian dibuat ulang.

Dengan kata lain databasenya akan di-refresh.

Perintah "npx" membutuhkan koneksi internet.

Script "db:stg:refresh"

Sama dengan "db:dev:refresh", tapi untuk environment staging.

Script "db:prod:refresh"

Sama dengan "db:dev:refresh", tapi untuk environment production.

File ".env.example" dan ".env"

Sebenarnya, di antara kedua file ini ada kesamaan.

Yakni isinya yang berupa konfigurasi aplikasi.

Bedanya, file yang dibaca adalah ".env".

Adapun, ".env.example" hanya template atau contoh untuk ".env".

Dalam kaitannya dengan repository git, ".env" seharusnya tidak di-push karena mengandung informasi sensitif.

Yang di-push ke repository hanya contohnya saja, yakni ".env.example" sebagai panduan untuk developer.

Sekarang, saya bahas baris-baris dari ".env.example".

```
BASE_URL=http://localhost:3000
```

Kode di atas menandakan bahwa base URL default dari aplikasi ini adalah berdomain "localhost" di port 3000.

Nantinya di "app.js" port tersebut akan dibaca.

```
SESSION_SECRET=secret1
```

Kode di atas akan dibaca di "app.js" saat menginisialisasi modul session sebagai setting-an secret-nya.

```
# pilih salah satu
KNEX_ENV=development
#KNEX_ENV=staging
#KNEX_ENV=production
```

Kode di atas ada 3 baris, tapi hanya 1 saja yang seharusnya diaktifkan. Yang lain dinonaktifkan dengan tanda pagar "#".

Tujuan dari kode di atas adalah untuk menentukan environment mana yang akan digunakan developer.

Jika development, maka tanda pagar dihapus dari "KNEX_ENV=development", yang lainnya tetap diberi tanda pagar.

Jika staging, maka tanda pagar dihapus dari "KNEX_ENV=staging", yang lainnya tetap diberi tanda pagar.

Jika production, maka tanda pagar dihapus dari "KNEX_ENV=production", yang lainnya tetap diberi tanda pagar.

```
# development database (jangan gunakan path. nama file saja.)
KNEX_DEV_DATABASE=company_profile-dev.sqlite3
```

Kode di atas adalah konfigurasi database untuk environment development, perhatikan ada "_DEV" nya.

Di environment tersebut, kita menggunakan SQLite dan bukan MySQL.

```
# staging database
KNEX_STG_HOST=127.0.0.1
KNEX_STG_PORT=3306
KNEX_STG_USER=root
KNEX_STG_PASSWORD=root
KNEX_STG_DATABASE=company_profile-stg
```

Kode di atas adalah konfigurasi database untuk environment staging, perhatikan ada "_STG" nya.

Di environment tersebut, kita menggunakan MySQL.

Host-nya adalah "127.0.0.1".

Port-nya adalah "3306".

Username-nya adalah "root".

Password-nya adalah "root".

Nama database-nya adalah "company_profile-stg".

Penjelasan serupa juga berlaku pada bagian ini:

```
# production database
KNEX_PROD_HOST=127.0.0.1
KNEX_PROD_PORT=3306
KNEX_PROD_USER=root
KNEX_PROD_PASSWORD=root
KNEX_PROD_DATABASE=company_profile-prod
```

File "app.js"

File "app.js" adalah file bootstrapper aplikasi ini.

Di script "package.json" file "app.js" dijadikan input untuk Node JS.

Dalam file ini routes-routes dimuat sehingga penanganan request web dari pengunjung akan dilakukan di sana.

Di file "app.js" express dan middleware-nya juga akan di-setup.

Logger, json parser, body parser, session, cookie, flash, upload (multer) akan di-setup di file "app.js".

Di "app.js" view engine yang menggunakan EJS juga di-setup.

Di file ini juga server akan melakukan listen ke port yang tertulis pada "BASE_URL" di file ".env".

Sekarang, saya akan membahas kodenya.

```
// begin: import modules
const createError = require("http-errors");
const express = require("express");
const path = require("path");
const cookieParser = require("cookie-parser");
const logger = require("morgan");
const session = require("express-session");
const FileStore = require("session-file-store")(session);
const flash = require("connect-flash");
const multer = require("multer");
const { v4: uuidv4 } = require("uuid");
const url = require("url");
require("dotenv").config();

const indexRouter = require("./routes/index");
const adminRouter = require("./routes/admin");
const authRouter = require("./routes/auth");
// end: import modules
```

Kode di atas akan mengimpor semua modul yang dibutuhkan "app.js".

Perhatikan bahwa "indexRouter", "adminRouter", dan "authRouter" diambil dari folder project:

```
const indexRouter = require("./routes/index");
const adminRouter = require("./routes/admin");
const authRouter = require("./routes/auth");
```

Lainnya dari folder "node_modules":

```
const createError = require("http-errors");
const express = require("express");
const path = require("path");
const cookieParser = require("cookie-parser");
const logger = require("morgan");
const session = require("express-session");
const FileStore = require("session-file-store")(session);
```

```
const flash = require("connect-flash");
const multer = require("multer");
const { v4: uuidv4 } = require("uuid");
const url = require("url");
```

Adapun kode ini:

```
require("dotenv").config();
```

Akan menginisialisasi modul "dotenv" agar bisa membaca ".env".

Selanjutnya...

```
// inisialisasi expressjs
const app = express();
```

Kode di atas akan menginisialisasi express, sehingga objek express akan dimasukkan ke variabel "app".

```
// set view folder ke folder "views"
app.set("views", path.join(__dirname, "views"));

// set view engine yang digunakan adalah EJS
app.set("view engine", "ejs");
```

Kode di atas akan men-set view folder di folder "views".

Selanjutnya, ditentukan bahwa view engine-nya adalah EJS.

EJS itu sendiri adalah view engine atau template engine yang memiliki sintaks mirip HTML.

EJS itu sendiri bisa diprogram dengan bahasa yang menyerupai JavaScript.

Selanjutnya kode ini:

```
// inisialisasi morgan
app.use(logger("dev"));

// parse body json
app.use(express.json({ limit: "100mb" }));

// parse body urlencoded
app.use(
    express.urlencoded({
        limit: "100mb",
        extended: true,
        parameterLimit: 100000,
```

```
    })
);

// setup multer agar bisa upload gambar
app.use(
    multer({
        storage: multer.diskStorage({
            destination: (req, file, callback) => {
                callback(null, "./uploads");
            },
            filename: (req, file, callback) => {
                callback(null, uuidv4() + "-" + file.originalname);
            },
        }),
        fileFilter: (req, file, callback) => {
            if (file.mimetype == "image/png" || file.mimetype == "image/jpg" ||
file.mimetype == "image/jpeg" || file.mimetype == "image/gif") {
                callback(null, true);
            } else {
                callback(null, false);
            }
        },
    }).single("upload")
);

// parse cookie header
app.use(cookieParser());

// agar bisa menggunakan session
app.use(
    session({
        secret: process.env.SESSION_SECRET,
        // store: new FileStore(), // jika di-enable simpan session di file (ada
bug nya)
        resave: false,
        saveUninitialized: false,
    })
);

// inisialisasi connect-flash
app.use(flash());
```

Kode di atas akan mendaftarkan beragam middleware secara global.

Logger, json parser, body parser, session, cookie, flash, upload (multer) adalah contohnya.

Middleware-middleware itu memiliki kegunaan yang berbeda-beda.

Dengan multer misalnya, aplikasi ini jadi dapat menangani file upload.

Untuk yang lainnya Anda bisa cek di dokumentasinya masing-masing.

Selanjutnya kode ini:

```
// setup folder statis untuk menyimpan resources
app.use("/public", express.static(path.join(__dirname, "public")));
app.use("/uploads", express.static(__dirname + "/uploads"));
```

Kode di atas akan menentukan folder file statis bernama "public" dan menempatkannya pada path URL bernama "/public".

Jadi, "BASE_URL/public" (misal: https://localhost:3000/public) akan mengakses folder tersebut.

Isi foldernya itu sendiri adalah file gambar, file JavaScript, dan file CSS.

Isi subfolder vendor-nya itu juga serupa, hanya saja dari pihak ketiga, seperti:

- bootstrap
- chart.js
- jquery
- jquery-sortable

Selanjutnya kode ini:

```
// route admin ada di path "/admin" nantinya pada URL.
app.use("/admin", adminRouter);

// route auth ada di path "/auth" nantinya pada URL.
app.use("/auth", authRouter);

// route "/" ada di path "/" nantinya pada URL.
app.use("/", indexRouter);
```

Kode di atas akan mendaftarkan routes-routes sebagai middleware global.

Route untuk admin akan diakses melalui "BASE_URL/admin".

Route untuk auth akan diakses melalui "BASE_URL/auth".

Route untuk index akan diakses melalui "BASE_URL/".

Selanjutnya kode ini:

```
// setup halaman error 404 jika terjadi
app.use(function (req, res, next) {
    next(createError(404));
});

// setup halaman error lainnya jika terjadi
app.use(function (err, req, res, next) {
    res.status(err.status || 500);
    res.render("error", {
        error: {
```

```
            message: err.message,
            status: err.status,
            stack: err.stack,
        },
    });
});
```

Kode di atas adalah middleware untuk menangani error.

Response nya adalah http error code.

Dan yang terakhir:

```
// jalankan server pada port yang ada di BASE_URL atau 3000 jika tidak ada
const port = url.parse(process.env.BASE_URL).port | 3000;
app.listen(port, function () {
    console.log(`server berjalan di port ${port}`);
});
```

kode di atas akan menjalankan server express di port yang didapat dari BASE_URL atau jika tidak ada, yang digunakan adalah port 3000.

Subfolder "public"

Saya telah menyinggung subfolder ini pada bagian sebelumnya. Jadi kira-kira seperti itulah.

Subfolder "lib", "sessions", dan "uploads"

Subfolder-subfolder ini isinya kosong, tapi bukan berarti tidak ada gunanya.

Subfolder lib bisa diisi dengan kumpulan fungsi Node JS custom buatan Anda jika mau.

Subfolder "sessions" berisi file-file sessions jika kita menggunakan jenis session berupa file yang bisa dilakukan di "app.js":

```
// agar bisa menggunakan session
app.use(
    session({
        secret: process.env.SESSION_SECRET,
        // store: new FileStore(), // jika di-enable simpan session di file (ada
bug nya)
        // ...
```

Subfolder "uplaods" berisi file-file gambar yang telah ter-upload.

Subfolder "databases"

Subfolder ini terdiri dari 2 file dan satu subfolder.

Dua file tersebut adalah "databases/knexfile.js" dan "databases/connection.js".

Satu subfolder tersebut adalah "migrations" yang berisi 1 file migrasi berakhiran "_init".

Di subfolder ini juga file database SQLite ditempatkan jika kita menggunakan environment development.

File "databases/migrations/20230316035229_init.js"

Saat perintah migrasi yang ada di script "package.json" dijalankan, misalnya "db:dev:refresh", maka file migrasi berakhiran "_init" tadi akan dibaca dan diterapkan pada database target. Saat itu tabel-tabel dibuat dan diisi sesuai instruksi di file migrasi berakhiran "_init" ini:

```
const bcrypt = require("bcryptjs");

/**
 * @param { import("knex").Knex } knex
 * @returns { Promise<void> }
 */
exports.up = function (knex) {
    return knex.schema
        .createTable("admins", function (table) {
            table.increments("_id");
            table.string("name", 1000).notNullable();
            table.string("email", 1000).notNullable();
            table.string("password", 1000).notNullable();
            table.timestamps(true, true, true);
        })
        .createTable("messages", function (table) {
            table.increments("_id");
            table.string("name", 1000).notNullable();
            table.string("email", 1000).notNullable();
            table.string("message", 1000).notNullable();
            table.timestamps(true, true, true);
        })
        .createTable("texts", function (table) {
            table.increments("_id");
            table.string("siteTitle", 1000).notNullable();
            table.string("siteSEOTitle", 1000).notNullable();
            table.string("siteDescription", 1000);
            table.string("siteSEODescription", 1000);
            table.string("aboutSubHeading", 1000);
            table.string("aboutSubText", 5000);
            table.string("infoSubHeading", 1000);
            table.string("address", 1000);
            table.string("phone", 1000);
            table.string("email", 1000);
            table.timestamps(true, true, true);
        })
        .createTable("skills", function (table) {
            table.increments("_id");
            table.string("title", 1000).notNullable();
            table.integer("level").notNullable();
```

```
            table.timestamps(true, true, true);
        })
        .createTable("services", function (table) {
            table.increments("_id");
            table.string("title", 1000).notNullable();
            table.string("description", 1000).notNullable();
            table.string("svg", 10000).notNullable();
            table.timestamps(true, true, true);
        })
        .createTable("portfolios", function (table) {
            table.increments("_id");
            table.string("title", 1000).notNullable();
            table.string("path", 1000).notNullable();
            table.timestamps(true, true, true);
        })
        .createTable("carousels", function (table) {
            table.increments("_id");
            table.string("title", 1000).notNullable();
            table.string("description", 1000);
            table.string("path", 1000).notNullable();
            table.timestamps(true, true, true);
        })
        .then(() => {
            return knex("admins").insert([
                {
                    name: "admin",
                    email: "admin@example.com",
                    password: bcrypt.hashSync("admin", 12),
                },
            ]);
        })
        .then(() => {
            return knex("texts").insert([
                {
                    siteTitle: "Company Name",
                    siteSEOTitle: "Company Name - A leading software development
agency...",
                    siteDescription: "Company Name is a leading software
development agency...",
                    siteSEODescription: "Company Name is a leading software
development agency...",
                    aboutSubHeading: "Why?",
                    aboutSubText: "Lorem ipsum dolor sit amet consectetur
adipisicing elit. Odit magnam ad doloremque illum iure repellat vero neque dolore
consectetur, quidem corporis tenetur perspiciatis mollitia quos nemo architecto!
Veniam facilis nesciunt, dignissimos in officia tempora repellat autem ipsum,
nostrum accusamus aliquid reiciendis inventore quos aperiam voluptates debitis
magni sit laborum obcaecati possimus necessitatibus quasi adipisci ducimus?
Adipisci enim accusantium, dignissimos quae asperiores eaque tempore
reprehenderit? Quaerat blanditiis sint rerum accusamus aspernatur provident sunt
libero quibusdam totam. Magnam doloribus nisi, quas eligendi quasi modi, velit
alias nulla quae animi excepturi officiis, ipsam voluptatum? Maxime reprehenderit
tenetur facilis, unde obcaecati ratione id magni!",
                    infoSubHeading: "Our Office",
```

```
                    address: "Jl. Some Street No. 15, Some City, Some Province,
Some Country",
                    phone: "(021) yyy-xxxx",
                    email: "admin@example.com",
                },
            ]);
        });
};

/**
 * @param { import("knex").Knex } knex
 * @returns { Promise<void> }
 */
exports.down = function (knex) {
    return
knex.schema.dropTable("skills").dropTable("services").dropTable("portfolios").drop
Table("carousels").dropTable("texts").dropTable("messages").dropTable("admins");
};
```

Pertama, kode ini akan mengimpor "bcryptjs" karena akan digunakan saat menggenerate hash dari password admin:

```
const bcrypt = require("bcryptjs");
```

Selanjutnya kode ini akan terpanggil saat perintah migrasi pada command line dijalankan:

```
exports.up = function (knex) {
    return knex.schema
        .createTable("admins", function (table) {
            table.increments("_id");
            table.string("name", 1000).notNullable();
            table.string("email", 1000).notNullable();
            table.string("password", 1000).notNullable();
            table.timestamps(true, true, true);
        })
        // ...
```

Sedangkan, kode ini akan terpanggil saat perintah rollback pada command line dijalankan:

```
exports.down = function (knex) {
    return
knex.schema.dropTable("skills").dropTable("services").dropTable("portfolios").drop
Table("carousels").dropTable("texts").dropTable("messages").dropTable("admins");
};
```

Secara umum berarti, saat migrate, buat tabel dan isi sebagiannya dan saat rollback drop tabel-tabel tersebut.

Fungsi pembentukan schema-nya saya kira cukup terjelaskan sendiri.

"createTable" jelas artinya untuk membuat table.

"table.increments("_id")" jelas artinya buat kolom id yang auto increment.

Jika Anda paham SQL saya kira itu tidak terlalu sulit dipahami.

Mungkin ini saja yang saya bahas:

```
.then(() => {
            return knex("admins").insert([
                {
                    name: "admin",
                    email: "admin@example.com",
                    password: bcrypt.hashSync("admin", 12),
                },
            ]);
        })
```

Kode di atas akan meng-insert entry admin bernama "admin" dengan email "admin@example.com" dan ber-password "admin". Kemudian password tadi disimpan ke database dalam bentuk hash.

Perintah insert tadi akan dijalankan setelah tabel-tabel dibuat di saat migrasi.

File "databases/knexfile.js"

File ini dimulai dari kode:

```
// Update with your config settings.
require("dotenv").config({
    path: "../.env",
});
```

Kode di atas akan membaca file ".env" karena kita akan menggunakan sebagian barisnya untuk mengonfigurasi knex.

Selanjutnya kode ini:

```
development: {
        client: "sqlite3",
        connection: {
            filename: process.env.KNEX_DEV_DATABASE,
        },
        migrations: {
            tableName: "knex_migrations",
```

```
        },
    },
```

Kode di atas adalah definisi konfigurasi knex untuk environment development.

Tampak bahwa database yang digunakan adalah SQLite3:

```
client: "sqlite3",
```

nama file database-nya adalah yang ada pada baris "KNEX_DEV_DATABASE" di ".env":

```
KNEX_DEV_DATABASE=company_profile-dev.sqlite3
```

Pembacaannya yakni dengan cara mengakses variabel:

```
process.env.KNEX_DEV_DATABASE
```

Penjelasan yang sama berlaku untuk kode sisanya.

Bedanya di sana menggunakan MySQL:

```
staging: {
        client: "mysql2",
```

```
production: {
        client: "mysql2",
```

Terakhir, property-property objek tadi seperti: development, staging, dan production akan diekspor sehingga nantinya akan bisa digunakan dari script "databases/connection.js".

File "databases/connection.js"

File ini tujuannya adalah membuat koneksi database yang telah dipilih dan mengembalikan objek knex siap pakai.

Kode ini akan mengimpor modul yang dibutuhkan:

```
const knex = require("knex");
const knexfile = require("./knexfile");
const path = require("path");
```

Selanjutnya kode ini:

```
let knexEnv;
if (process.env.KNEX_ENV === "development") {
    knexEnv = knexfile.development;
} else if (process.env.KNEX_ENV === "staging") {
    knexEnv = knexfile.staging;
} else if (process.env.KNEX_ENV === "production") {
    knexEnv = knexfile.production;
} else {
    throw Error("invalid environment.");
}
```

Kode di atas akan membaca konfigurasi ".env" tentang apa environment yang dipilih.

Jika "development", maka baca variabel yang diexport oleh "databases/knexfile.js", yakni knexfile.development.

Ingat kembali file "database/knexfile.js" bagian ini:

```
module.exports = {
    development
```

Di sana ada "module.exports"-nya bukan?

Sekarang kode ini:

```
console.log("BUGFIX WORKAROUND !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");
knexfile.development.connection.filename = path.join(__dirname,
knexfile.development.connection.filename);
console.log(knexfile.development.connection.filename);
```

Kode di atas akan memodifikasi objek property dari knexfile agar filename menggunakan absolute path.

Saya melakukan ini karena saat mencoba dengan SQLite tanpa itu gagal terus.

Kode di bawah ini akan membuat objek knex siap pakai dan mengekspornya agar nantinya bisa digunakan di script lain:

```
const db = knex(knexEnv);

module.exports = db;
```

Subfolder "routes"

Subfolder "routes" berisi kode routing web request dari pengunjung ke controller dan sebagiannya setelah melewati middleware sessionchecker.

Di subfolder tersebut ada dua jenis request: get dan post.

Adapun routes dipisahkan menjadi tiga script: "admin.js", "auth.js", dan "index.js".

Berikut ini penjelasannya.

File "routes/index.js"

Routes "index" merupakan routes yang diakses melalui path "/" pada URL.

Jika Anda lupa lihat kembali pembahasan "app.js".

```
// script ini tugasnya adalah menghubungkan controllers/index.js dengan route.

const express = require("express");
const indexController = require("../controllers/index");

// buat objek router agar bisa memakai get, post, dan lain-lain.
const router = express.Router();

router.get("/", indexController.getIndex);

router.post("/send-message", indexController.postIndexSendMessage);

module.exports = router;
```

Pada kode "routes/index.js" di atas, tampak tidak ada middleware yang digunakan.

Itu karena pada dasarnya route ini hanya mengarahkan request "/" atau homepage dan "/send-message" untuk mengirim contact us.

File "routes/auth.js"

Routes "auth" merupakan routes yang diakses melalui path "/auth" pada URL.

Routes "auth" merupakan penghubung dari routes "index" ke routes "admin".

Itu karena pada routes ini, controller "auth" yang menangani login dan register akan dihubungkan.

Berikut ini kodenya:

```
// script ini tugasnya adalah menghubungkan controllers/auth.js dengan route.

const express = require("express");
const authController = require("../controllers/auth");
```

```
// session checker digunakan untuk authorization
const sessionChecker = require("../middlewares/sessionchecker");

// buat objek router agar bisa memakai get, post, dan lain-lain.
const router = express.Router();

router.get("/login", sessionChecker.loggedIn, authController.getLogin);

router.post("/login", sessionChecker.loggedIn, authController.postLogin);

router.get("/register", authController.getRegister);

router.post("/register", authController.postRegister);

router.get("/logout", authController.getLogout);

module.exports = router;
```

Pada kode di atas, tampak bahwa middleware sessionchecker digunakan di beberapa baris. Oleh karena itulah middleware tersebut diimpor.

Berikut ini contoh baris yang menggunakannya:

```
router.get("/login", sessionChecker.loggedIn, authController.getLogin);

router.post("/login", sessionChecker.loggedIn, authController.postLogin);
```

Pada kode di atas, tampak juga fungsi yang digunakan adalah sessionChecker.loggedIn yang akan dijelaskan pada pembahasan bagian middlewares.

File "routes/admin.js"

Routes "admin" merupakan routes yang diakses melalui path "/admin" pada URL.

Routes "admin" merupakan routes yang mengarahkan request seputar admin dashboard.

Middleware sessionchecker banyak digunakan di sini.

Berikut ini adalah potongan kodenya:

```
// script ini tugasnya adalah menghubungkan controllers/admin.js dengan route.

const express = require("express");
const adminController = require("../controllers/admin");

// session checker digunakan untuk authorization
const sessionChecker = require("../middlewares/sessionchecker");
```

```
// buat objek router agar bisa memakai get, post, dan lain-lain.
const router = express.Router();

router.get("/", sessionChecker.notLoggedIn, adminController.getIndex);

//...
```

Pada kode di atas, tampak juga fungsi yang digunakan adalah sessionChecker.notLoggedIn yang akan dijelaskan pada pembahasan bagian middlewares.

Jika kita bandingkan dengan routes "auth" tadi:

- auth menggunakan sessionChecker.loggedIn
- admin menggunakan sessionChecker.notLoggedIn

Terasa masuk akal, bukan?

Jika masuk ke bagian admin, maka yang belum login harus difilter.

Selain itu, jika masuk bagian login, maka yang sudah login harus difilter.

Subfolder "middlewares"

Subfolder ini berisi script middleware yang digunakan pada script-script di subfolder "routes".

Saat ini hanya satu script saja yang kita gunakan: "sessionchecker.js".

Berikut ini adalah kodenya:

```
// script ini tugasnya adalah menjadi middleware

module.exports.notLoggedIn = (req, res, next) => {
    if (!req.session.isLoggedIn) {
        // jika varieble isLoggedIn dalam session false, null, atau undefined

        // maka redirect ke /auth/login
        return res.redirect("/auth/login");
    }
    next();
};

module.exports.loggedIn = (req, res, next) => {
    if (req.session.isLoggedIn) {
        // jika varieble isLoggedIn dalam session true

        // maka redirect ke /admin
        return res.redirect("/admin");
    }
    next();
};
```

Ada dua fungsi yang diekspor: notLoggedIn dan loggedIn.

Seperti yang telah saya bahas sebelumnya, auth menggunakan loggedIn dan admin menggunakan notLoggedIn.

Kode ini:

```
module.exports.notLoggedIn = (req, res, next) => {
    if (!req.session.isLoggedIn) {
        // jika varieble isLoggedIn dalam session false, null, atau undefined

        // maka redirect ke /auth/login
        return res.redirect("/auth/login");
    }
    next();
};
```

Kode di atas memiliki fungsi notLoggedIn yang parameternya adalah req, res, dan next.

- req maksudnya adalah request
- res maksudnya adalah response
- next jika dipanggil sebagai fungsi maka akan melanjutkan ke middleware selanjutnya.

Di sini:

```
if (!req.session.isLoggedIn) {
        // jika varieble isLoggedIn dalam session false, null, atau undefined
```

Kita mempertanyakan apakah pengunjung memiliki session login.

Jika tidak, maka redirect ke halaman login:

```
        // maka redirect ke /auth/login
        return res.redirect("/auth/login");
```

Sekarang, kode ini:

```
module.exports.loggedIn = (req, res, next) => {
    if (req.session.isLoggedIn) {
        // jika varieble isLoggedIn dalam session true

        // maka redirect ke /admin
        return res.redirect("/admin");
    }
```

```
    next();
};
```

Kode di atas memiliki fungsi notLoggedIn yang parameternya adalah req, res, dan next.

- req maksudnya adalah request
- res maksudnya adalah response
- next jika dipanggil sebagai fungsi maka akan melanjutkan ke middleware selanjutnya.

Di sini:

```
  if (req.session.isLoggedIn) {
      // jika varieble isLoggedIn dalam session true
```

Kita mempertanyakan apakah pengunjung memiliki session login.

Jika ya, maka redirect ke halaman admin:

```
      // maka redirect ke /admin
      return res.redirect("/admin");
```

Subfolder "controllers"

Subfolder ini berisi controller yang tidak lain adalah fungsi yang menangani route.

Di controller, query-query database dilakukan dan hasilnya akan menjadi input bagi view melalui proses rendering.

Controller memiliki 3 file script berikut ini:

- admin.js
- auth.js
- index.js

File "controllers/index.js"

File ini diimpor di "routes/index.js" dan fungsi dijadikan input bagi event routing-nya seperti get dan post.

Script ini dimulai dengan impor modul:

```
// begin: import modules
const createError = require("http-errors");
const knex = require("../databases/connection");
// end: import modules
```

Method ini akan me-render home page:

```
// merender homepage
module.exports.getIndex = async function (req, res, next) {
    try {
        // ambil isi masing-masing tabel dan simpan ke variabel
        const text = await knex("texts").first();
        const skills = await knex("skills");
        const services = await knex("services");
        const carousels = await knex("carousels");
        const portfolios = await knex("portfolios");

        // render dan kirimkan datanya yang berupa isi tabel
        res.render("index.ejs", {
            text: text,
            skills: skills,
            services: services,
            carousels: carousels,
            portfolios: portfolios,
        });
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};
```

Kode ini meminta data dari database:

```
        // ambil isi masing-masing tabel dan simpan ke variabel
        const text = await knex("texts").first();
        const skills = await knex("skills");
        const services = await knex("services");
        const carousels = await knex("carousels");
        const portfolios = await knex("portfolios");
```

Untuk nantinya akan dijadikan sebagai input bagi view saat di-render:

```
        // render dan kirimkan datanya yang berupa isi tabel
        res.render("index.ejs", {
            text: text,
            skills: skills,
            services: services,
            carousels: carousels,
            portfolios: portfolios,
        });
```

Adapun view yang di-render adalah "views/index/index.ejs".

File "controllers/auth.js"

File ini diimpor di "routes/auth.js" untuk nantinya digunakan sebagai definisi callback dari fungsi routing.

Script ini dimulai dengan mengimpor modul yang dibutuhkan:

```
// begin: import modules
const createError = require("http-errors");
const bcrypt = require("bcryptjs");
const Joi = require("joi");
const knex = require("../databases/connection");
// end: import modules
```

Modul "http-errors" digunakan untuk membuat halaman error tergantung error code-nya.

Modul "bcryptjs" digunakan untuk membuat hash dari input password.

Modul "joi" digunakan untuk validasi input di sisi server.

Modul "../databases/connection" adalah objek knex siap pakai yang nantinya digunakan untuk membuat query ke database.

Fungsi ini akan mengambil data dari database kemudian menjadikannya input untuk proses render halaman login:

```
// render halaman login
module.exports.getLogin = async function (req, res, next) {
    try {
        // ambil data tabel texts
        const text = await knex("texts").first();

        // render dan kirim data tadi ke front end
        res.render("auth/layout.ejs", {
            child: "auth/login.ejs",
            clientScript: "auth/login.js.ejs",
            data: {
                text: text,
                errors: req.flash("errors"),
            },
        });
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};
```

Data yang di-query di sini akan berisi informasi seputar SEO:

```
        // ambil data tabel texts
        const text = await knex("texts").first();
```

Seperti yang telah disinggung pada pembahasan sebelumnya tentang script migrasi:

```
// file: "databases/migrations/20230316035229_init.js"
              return knex("texts").insert([
                {
                    siteTitle: "Company Name",
                    siteSEOTitle: "Company Name - A leading software development
agency...",
                    siteDescription: "Company Name is a leading software
development agency...",
                    siteSEODescription: "Company Name is a leading software
development agency...",
                    aboutSubHeading: "Why?",
                    aboutSubText: "Lorem ipsum dolor sit amet consectetur
adipisicing elit. Odit magnam ad doloremque illum iure repellat vero neque dolore
consectetur, quidem corporis tenetur perspiciatis mollitia quos nemo architecto!
Veniam facilis nesciunt, dignissimos in officia tempora repellat autem ipsum,
nostrum accusamus aliquid reiciendis inventore quos aperiam voluptates debitis
magni sit laborum obcaecati possimus necessitatibus quasi adipisci ducimus?
Adipisci enim accusantium, dignissimos quae asperiores eaque tempore
reprehenderit? Quaerat blanditiis sint rerum accusamus aspernatur provident sunt
libero quibusdam totam. Magnam doloribus nisi, quas eligendi quasi modi, velit
alias nulla quae animi excepturi officiis, ipsam voluptatum? Maxime reprehenderit
tenetur facilis, unde obcaecati ratione id magni!",
                    infoSubHeading: "Our Office",
                    address: "Jl. Some Street No. 15, Some City, Some Province,
Some Country",
                    phone: "(021) yyy-xxxx",
                    email: "admin@example.com",
                },
            ]);
```

Selanjutnya, proses render dilakukan:

```
        // render dan kirim data tadi ke front end
        res.render("auth/layout.ejs", {
            child: "auth/login.ejs",
            clientScript: "auth/login.js.ejs",
            data: {
                text: text,
                errors: req.flash("errors"),
            },
        });
```

Perlu Anda perhatikan bagian ini:

```
            child: "auth/login.ejs",
            clientScript: "auth/login.js.ejs",
```

Ada 2 input file view yang berekstensi "ejs".

File "auth/login.ejs" adalah bagian HTML-nya.

File "auth/login.js.ejs" adalah bagian script JavaScript yang ada di bagian footer-nya.

Selain itu, bagian ini:

```
            data: {
                text: text,
                errors: req.flash("errors"),
            },
```

Adalah transfer data text (yang berisi info seputar SEO) dan pesan error jika ada.

Sekarang, saya akan membahas fungsi postLogin-nya:

```
// handle form login
module.exports.postLogin = async function (req, res, next) {
    // validasi request body
    const schema = Joi.object({
        email: Joi.string().required(),
        password: Joi.string().required(),
    });

    const validObj = schema.validate(req.body);
    if (validObj.error) {
        // jika ada error dalam request body

        // buat pesan error flash
        req.flash("errors", validObj.error.details);
        res.status(422).redirect("/auth/login");
        res.end();
        return;
    }

    // bongkar request body
    const { email, password } = req.body;

    try {
        // ambil admin dari tabel admins yang emailnya adalah email dari request
body tadi
        const admin = await knex("admins").where({ email: email }).first();

        if (admin) {
            // jika admin tadi ada
```

```
            if (email == admin.email) {
                // jika email dari request body sama dengan yang di tabel

                // bandingkan passwordnya
                if (bcrypt.compareSync(password, admin.password) == true) {
                    // jika password sama

                    // masukkan data-data ini ke session
                    req.session.isLoggedIn = true;
                    req.session.admin = admin;
                    req.session.save();

                    // redirect ke halaman admin
                    res.redirect("/admin");
                } else {
                    // jika password beda

                    // redirect ke halaman login
                    res.redirect("/auth/login");
                }
            } else {
                // jika email dari request body beda dengan yang di tabel

                // redirect ke halaman login
                res.redirect("/auth/login");
            }
        } else {
            // jika admin tidak ada

            // redirect ke halaman login
            res.redirect("/auth/login");
        }
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};
```

Fungsi di atas adalah untuk menangani request post yang dikirim dari view melalui form tag karena pada "views/auth/login.ejs", ada kode ini:

```
<form action="/auth/login" method="POST">
```

Komentar kode di atas cukup menjelaskan secara rinci menurut saya.

Itu adalah algoritma yang cukup umum untuk keperluan login.

Serupa dengan kode tadi, method getRegister untuk merender form, dan postRegister untuk menangani request post dari form tersebut.

Sekarang, saya akan membahas tentang "controllers/admin.js".

Kodenya begini:

```
// script ini akan dipanggil di routes/admin.js

const bcrypt = require("bcryptjs");
const fs = require("fs");
const createError = require("http-errors");
const osu = require("node-os-utils");
const knex = require("../databases/connection");

// render halaman admin index
module.exports.getIndex = async function (req, res, next) {
    try {
        // ambil data text pertama dari tabel texts
        const text = await knex("texts").first();

        // render dan kirimkan data tadi ke front end
        res.render("admin/layout.ejs", {
            child: "admin/index.ejs",
            clientScript: "admin/index.js.ejs",
            data: {
                text: text,
            },
        });
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// dapatkan data penggunaan CPU
module.exports.getCPUUsage = async function (req, res, next) {
    osu.cpu.usage().then((cpuPercentage) => {
        res.status(200).json({
            name: "CPU Usage",
            value: cpuPercentage,
        });
    });
};

// dapatkan data penggunaan memory
module.exports.getMemoryUsage = function (req, res) {
    osu.mem.used().then((memUsed) => {
        res.status(200).json({
            name: "Memory Usage",
            value: (memUsed.usedMemMb / memUsed.totalMemMb) * 100,
        });
    });
};

// render halaman settings
```

```
module.exports.getSettings = async function (req, res) {
    try {
        // ambil data text pertama dari tabel texts
        const text = await knex("texts").first();

        // render dan kirimkan data tadi ke front end
        res.render("admin/layout", {
            child: "admin/settings.ejs",
            clientScript: "admin/settings.js.ejs",
            data: {
                adminEmail: req.session.admin.email,
                text: text,
                errors: req.flash("errors"),
            },
        });
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// handle edit settings
module.exports.postSettingsEdit = async function (req, res, next) {
    try {
        // bongkar request body
        const { email, password } = req.body;

        if (password && email) {
            // jika password dan email ada

            // update data admin
            const ret = await knex("admins")
                .where({ email: req.session.admin.email })
                .update({
                    email: email,
                    password: bcrypt.hashSync(password, 12),
                });
        } else if (email) {
            // jika hanya email yang ada

            // update data admin
            const ret = await knex("admins").where({ email:
req.session.admin.email }).update({
                email: email,
            });
        }

        // redirect ke settings
        res.redirect("/admin/settings");
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};
```

```
// render halaman messages
module.exports.getMessagesIndex = async function (req, res) {
    try {
        // ambil data text pertama dari tabel texts
        const text = await knex("texts").first();

        // ambil data messages dari tabel messages
        const messages = await knex("messages");

        // render dan kirimkan data tadi ke front end
        res.render("admin/layout", {
            child: "admin/messages.ejs",
            clientScript: "admin/messages.js.ejs",
            data: {
                text: text,
                results: messages,
            },
        });
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// handle delete message
module.exports.getMessagesDelete = async function (req, res, next) {
    try {
        // delete message yang id-nya adalah req.params.id
        // atau dengan kata lain
        // yang id-nya sama dengan yang ada di URL
        await knex("messages")
            .where({
                _id: req.params.id,
            })
            .del();

        // redirect ke messages
        res.redirect("/admin/messages");
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// render halaman texts
module.exports.getTextsIndex = async function (req, res) {
    try {
        // ambil data text pertama dari tabel texts
        const text = await knex("texts").first();

        // render dan kirimkan data tadi ke front end
        res.render("admin/layout", {
            child: "admin/texts.ejs",
```

```
            clientScript: "admin/texts.js.ejs",
            data: {
                adminEmail: req.session.admin.email,
                text: text,
                errors: req.flash("errors"),
            },
        });
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// handle edit data dari tabel texts
module.exports.postTextsEdit = async function (req, res, next) {
    try {
        // ambil data text pertama dari tabel texts
        const target = await knex("texts").first();

        // update data tersebut
        const ret = await knex("texts").where({ _id: target._id }).update({
            siteTitle: req.body.siteTitle,
            siteSEOTitle: req.body.siteSEOTitle,
            siteDescription: req.body.siteDescription,
            siteSEODescription: req.body.siteSEODescription,
            aboutSubHeading: req.body.aboutSubHeading,
            aboutSubText: req.body.aboutSubText,
            infoSubHeading: req.body.infoSubHeading,
            address: req.body.address,
            phone: req.body.phone,
            email: req.body.email,
        });

        // redirect ke halaman texts
        res.redirect("/admin/texts");
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// render halaman skills
module.exports.getSkillsIndex = async function (req, res, next) {
    try {
        // ambil data text pertama dari tabel texts
        const text = await knex("texts").first();

        // ambil data skills dari tabel skills
        const allSkills = await knex("skills");

        // render dan kirim data tersebut ke front end
        res.render("admin/layout.ejs", {
            child: "admin/skills.ejs",
            clientScript: "admin/skills.js.ejs",
```

```
            data: {
                results: allSkills,
                text: text,
            },
        });
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// handle penambahan skill
module.exports.postSkillsAdd = async function (req, res, next) {
    try {
        // bongkar request body
        const { title, level } = req.body;

        // insert skill baru, judul dan levelnya
        const skillId = await knex("skills").insert({
            title: title,
            level: level,
        });

        // redirect ke halaman skills
        res.redirect("/admin/skills");
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// handle delete skill
module.exports.getSkillsDelete = async function (req, res, next) {
    try {
        // delete skill yang id-nya adalah req.params.id
        // atau dengan kata lain
        // yang id-nya sama dengan yang ada di URL
        await knex("skills")
            .where({
                _id: req.params.id,
            })
            .del();

        // redirect ke halaman skills
        res.redirect("/admin/skills");
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// render halaman services
module.exports.getServicesIndex = async function (req, res, next) {
    try {
```

```
        // ambil data text pertama dari tabel texts
        const text = await knex("texts").first();

        // ambil data services dari tabel services
        const allServices = await knex("services");

        // render dan kirim data tersebut ke front end
        res.render("admin/layout.ejs", {
            child: "admin/services.ejs",
            clientScript: "admin/services.js.ejs",
            data: {
                results: allServices,
                text: text,
            },
        });
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// handle tambah service
module.exports.postServicesAdd = async function (req, res, next) {
    try {
        // bongkar request body
        const { title, description, svg } = req.body;

        // insert service baru, judul, deskripsi, dan svg nya
        const serviceId = await knex("services").insert({
            title: title,
            description: description,
            svg: svg,
        });

        // redirect ke halaman services
        res.redirect("/admin/services");
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// handle delete service
module.exports.getServicesDelete = async function (req, res, next) {
    try {
        // delete service yang id-nya adalah req.params.id
        // atau dengan kata lain
        // yang id-nya sama dengan yang ada di URL
        await knex("services")
            .where({
                _id: req.params.id,
            })
            .del();
```

```
        // redirect ke halaman services
        res.redirect("/admin/services");
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// render halaman carousel
module.exports.getCarouselsIndex = async function (req, res, next) {
    try {
        // ambil data text pertama dari tabel texts
        const text = await knex("texts").first();

        // ambil data carousels dari tabel carousels
        const allCarousels = await knex("carousels");

        // render dan kirim data tadi ke front end
        res.render("admin/layout.ejs", {
            child: "admin/carousels.ejs",
            clientScript: "admin/carousels.js.ejs",
            data: {
                results: allCarousels,
                text: text,
            },
        });
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// handle upload gambar carousel
module.exports.postCarouselsUpload = async function (req, res, next) {
    try {
        if (req.file) {
            // jika request file ada, yang artinya gambar diupload

            // bongkar request body
            const { title, description } = req.body;

            // insert carousel baru, judul, deskripsi, dan path dari file
gambarnya
            const fileId = await knex("carousels").insert({
                title: title,
                description: description,
                path: req.file.path.replace("\\", "/"),
            });
        }

        // redirect ke halaman carousels
        res.redirect("/admin/carousels");
    } catch (err) {
        console.log(err);
```

```
        next(createError(500));
    }
};

// handle delete carousel
module.exports.getCarouselsDelete = async function (req, res, next) {
    try {
        // dapatkan dulu datanya
        const willBeDeleted = await knex("carousels").where({
            _id: req.params.id,
        });

        // delete carousel yang id-nya adalah req.params.id
        // atau dengan kata lain
        // yang id-nya sama dengan yang ada di URL
        // ini yang di database
        await knex("carousels")
            .where({
                _id: req.params.id,
            })
            .del();

        // hapus file nya
        // ini yang di folder upload
        fs.unlinkSync("./" + willBeDeleted[0].path);

        // redirect ke halaman carousels
        res.redirect("/admin/carousels");
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// render halaman portfolios
module.exports.getPortfoliosIndex = async function (req, res, next) {
    try {
        // ambil data text pertama dari tabel texts
        const text = await knex("texts").first();

        // ambil data portfolios dari tabel portfolios
        const allPortfolios = await knex("portfolios");

        // render dan kirim data tadi ke front end
        res.render("admin/layout.ejs", {
            child: "admin/portfolios.ejs",
            clientScript: "admin/portfolios.js.ejs",
            data: {
                results: allPortfolios,
                text: text,
            },
        });
    } catch (err) {
        console.log(err);
```

```
        next(createError(500));
    }
};

// handle upload gambar portfolio
module.exports.postPortfoliosUpload = async function (req, res, next) {
    try {
        if (req.file) {
            // jika request file ada, yang artinya gambar diupload

            // bongkar request body
            const { title } = req.body;

            // insert portfolio baru, judul, dan path dari file gambarnya
            const fileId = await knex("portfolios").insert({
                title: title,
                path: req.file.path.replace("\\", "/"),
            });
        }

        // redirect ke halaman portfolios
        res.redirect("/admin/portfolios");
    } catch (err) {
        console.log(err);
        next(createError(500));
    }
};

// handle delete portfolio
module.exports.getPortfoliosDelete = async function (req, res, next) {
    try {
        // dapatkan dulu datanya
        const willBeDeleted = await knex("portfolios").where({
            _id: req.params.id,
        });

        // delete portfolio yang id-nya adalah req.params.id
        // atau dengan kata lain
        // yang id-nya sama dengan yang ada di URL
        // ini yang di database
        await knex("portfolios")
            .where({
                _id: req.params.id,
            })
            .del();

        // hapus file nya
        // ini yang di folder upload
        fs.unlinkSync("./" + willBeDeleted[0].path);

        // redirect ke halaman portfolios
        res.redirect("/admin/portfolios");
    } catch (err) {
        console.log(err);
```

```
        next(createError(500));
    }
};
```

Panjang sekali, ya...

Saya tidak tertarik untuk menulis ulang penjelasan kode tadi, karena komentarnya juga cukup jelas dan banyak.

Kecuali pada kedua hal terkait modul-modul ini:

```
const fs = require("fs");
```

Dan

```
const osu = require("node-os-utils");
```

Selain kedua hal tersebut, apa yang dilakukan oleh script "controllers/admin.js" adalah hanya create, read, update, dan delete.

Modul "fs" digunakan nantinya untuk menghapus file gambar yang telah di-upload. Bukan dari database, tapi dari filesystem.

Kita menggunakan database untuk menyimpan data tentang file, tapi file-nya itu sendiri disimpan pada filesystem, dalam hal ini folder "uploads".

Jadi kode ini:

```
fs.unlinkSync("./" + willBeDeleted[0].path);
```

Akan menghapus file gambar dari filesystem.

Anda akan mendapati dua implementasi tadi karena dalam company profile ada gambar carousel dan ada gambar portfolio.

Sekarang tentang modul "node-os-utils" yang disimpan dalam variabel "osu":

```
// dapatkan data penggunaan CPU
module.exports.getCPUUsage = async function (req, res, next) {
    osu.cpu.usage().then((cpuPercentage) => {
        res.status(200).json({
            name: "CPU Usage",
            value: cpuPercentage,
        });
    });
```

```
};

// dapatkan data penggunaan memory
module.exports.getMemoryUsage = function (req, res) {
    osu.mem.used().then((memUsed) => {
        res.status(200).json({
            name: "Memory Usage",
            value: (memUsed.usedMemMb / memUsed.totalMemMb) * 100,
        });
    });
};
```

Fungsi-fungsi di atas digunakan karena pada halaman dashboard admin ada dua chart yang menampilkan penggunaan resource dari server.

Yang pertama adalah penggunaan CPU dan yang kedua adalah penggunaan memory.

Kedua fungsi tadi dipanggil dari client melalui ajax dalam interval tertentu.

Jadi, itu dipanggil via JavaScript browser dari sisi client.

Dan ingat, pemanggilan itu hanya dilakukan dari view yang menjalankan JavaScript di halaman dashboard admin saja, yakni "views/admin/index.js.ejs". bukan dari "views/admin/index.ejs".

Penting untuk Diketahui Sebelum Membahas Subfolder "views"

Mungkin ada di antara pembaca yang bingung setelah pembahasan saya sampai di sini.

View merupakan file yang ada di server, tapi kenapa ada pemanggilan ajax di sisi client?

Sebenarnya alasannya sederhana dan ini soal prinsip.

Folder "views" memang ada di server.

Akan tetapi hasil render dengan ketentuan file-file ejs yang ada di "views" nantinya akan diterima client berupa HTML yang sudah diproses.

Jadi, HTML itulah yang memiliki JavaScript dan JavaScript tersebut melakukan fungsi ajax untuk meminta request ke server.

Subfolder "views"

Sekarang mungkin Anda sudah tidak bingung setelah penjelasan di subbab sebelum ini.

Oleh karena itu, saya akan membahas sisanya.

Penting untuk diketahui bahwa saya tidak akan membahas semua file .ejs yang ada.

Itu karena akan sangat panjang.

Jadi saya bahas yang penting-penting saja, terutama untuk file yang berekstensi "*.js.ejs".

File "views/*/layout.ejs"

Sekarang kita mulai dari "layout.ejs" yang ada di dua tempat berbeda:

- admin/layout.ejs
- auth/layout.ejs

Kedua file tadi beda isinya, tapi ada kesamaan:

```
<!-- file "auth/layout.ejs" -->

<% include ("../helper") %>
    <%- include("../" + child, {data: data}); %>

<!-- sesuatu di antaranya -->

    <%- include("../" + clientScript, {data: data}); %>
```

```
<!-- file "admin/layout.ejs" -->

<% include ("../helper") %>

      <%- include("../" + child, {data: data}); %>

<!-- sesuatu di antaranya -->

      <%- include("../" + clientScript, {data: data}); %>
```

Keduanya memiliki kode di atas.

Simbol "<% sesuatu %>" dan "<%- sesuatu %>" itu adalah ciri khas template engine EJS.

Artinya dengan simbol tadi, operasi logika rendering dari EJS yang ada di antara simbol tadi akan dijalankan.

Dengan fungsi include maka "views/helper.ejs" akan disertakan:

```
<%
generateExcerpt = function(wholeText) {
    var strLen = wholeText.length;
    if (strLen < 100) {
        return wholeText.substring(0, strLen).replace(/<[^(^(>)]*>?/gm, '');
    }
    return wholeText.substring(0, 100).replace(/<[^(^(>)]*>?/gm, '');
}

isOptionSelected = function (val, valDB) {
    return val == valDB ? "selected" : "";
}

pgnPrevious = function (currentPage) {
    let tmp = currentPage - 1;
```

```
    return tmp < 0 ? { index: 0, disabled: true } : { index: tmp, disabled: false
};
}

pgnNext = function (currentPage, totalPage) {
    let tmp = currentPage + 1;
    return tmp >= totalPage ? { index: currentPage, disabled: true } : { index:
tmp, disabled: false };
}
%>
```

Kode di atas isinya adalah fungsi untuk membantu pemrosesan view dari template mentah ke HTML jadi.

Kodenya menyerupai JavaScript, tapi mungkin Anda tidak bisa menggunakan fungsi dalam konteks web browser, karena script tersebut dijalankan di sisi server.

Beda lagi dengan JavaScript yang ada di "*.js.ejs" setelah diterima di sisi client.

Selain itu, dengan fungsi include maka ""../" + child, {data: data}" juga akan disertakan:

```
<%- include("../" + child, {data: data}); %>
```

Variabel "child" adalah nama file ".ejs" yang diinputkan pada fungsi render di controller yang telah saya bahas tadi.

Jika lupa, ini contohnya:

```
// file: "controllers/admin.js"

        // render dan kirimkan data tadi ke front end
        res.render("admin/layout.ejs", {
            child: "admin/index.ejs",
            clientScript: "admin/index.js.ejs",
            data: {
                text: text,
            },
        });
```

Yang ini misalnya:

```
child: "admin/index.ejs",
```

Adapun variabel clientScript seperti di sini:

```
<%- include("../" + clientScript, {data: data}); %>
```

Akan dimasukkan file "*.js.ejs":

```
clientScript: "admin/index.js.ejs",
```

Jika Anda perhatikan posisinya, mungkin Anda akan paham bahwa itu ada di bagian footer atau bawah.

Tujuannya agar script JavaScript-nya diproses di akhir.

File "views/admin/index.js.ejs"

Karena di bagian dashboard admin ada 2 pie chart, maka kode ini diterapkan:

```
<script>
    var configCPUUsagePie = {
        type: 'pie',
        data: {
            labels: [
                "CPU Used",
                "CPU Free"
            ],
            datasets: [{
                data: [100, 50],
                backgroundColor: [
                    "#FF6384",
                    "#36A2EB"
                ],
                hoverBackgroundColor: [
                    "#FF6384",
                    "#36A2EB"
                ]
            }]
        }
    };

    var cpuUsageChartPie = new Chart(
        document.getElementById("cpuUsageChartPie").getContext("2d"),
        configCPUUsagePie
    );

    setInterval(function () {
        $.get("/admin/cpu-usage", function (data) {
            configCPUUsagePie.data.datasets[0].data[0] = data.value;
            configCPUUsagePie.data.datasets[0].data[1] = (100 - data.value);
            cpuUsageChartPie.update();
        });
    }, 1000);
```

```
    var configMemoryUsagePie = {
        type: 'pie',
        data: {
            labels: [
                "Memory Used",
                "Memory Free"
            ],
            datasets: [{
                data: [100, 50],
                backgroundColor: [
                    "#FF6384",
                    "#36A2EB"
                ],
                hoverBackgroundColor: [
                    "#FF6384",
                    "#36A2EB"
                ]
            }]
        }
    };

    var memoryUsageChartPie = new Chart(
        document.getElementById("memoryUsageChartPie").getContext("2d"),
        configMemoryUsagePie
    );

    setInterval(function () {
        let self = this;
        $.get("/admin/memory-usage", function (data) {
            configMemoryUsagePie.data.datasets[0].data[0] = data.value;
            configMemoryUsagePie.data.datasets[0].data[1] = 100 - data.value;
            memoryUsageChartPie.update();
        });

    }, 1000);
</script>
```

File "views/admin/carousels.js.ejs", "views/admin/services.js.ejs", "views/admin/skills.js.ejs", dan "views/admin/portfolios.js.ejs"

File ini digunakan untuk meng-handle event click pada tombol upload di admin area.

Tepatnya di bagian carousels, services, skills, dan portfolios.

```
<script>
  $("#btn-upload-file").click(function() {
    let addModal = new bootstrap.Modal(document.getElementById("modal-add"), {
      backdrop: 'static',
      keyboard: false
    });
```

```
    addModal.show();
  });
</script>
```

Dengan demikian modal dialog akan muncul pada saat tombol upload diklik.

File "public/js/auth.js", "views/auth/login.js.ejs", "views/auth/register.js.ejs"

File-file views di folder auth sedikit berbeda.

Itu karena mereka memiliki dependency dengan file statis JavaScript di folder "public".

Tepatnya di "public/js/auth.js":

```
// untuk menampilkan modal login dan register
function showAuthModal(elmID) {
    let authModal = new bootstrap.Modal(document.getElementById(elmID), {
backdrop: "static", keyboard: false });
    authModal.show();
}
```

Nanti di "views/auth/login.js.ejs" dan "views/auth/register.js.ejs" kode di atas akan dipanggil:

```
<!-- file: "views/auth/login.js.ejs" -->
<script>
    showAuthModal("loginModal");
</script>
```

```
<!-- file: "views/auth/register.js.ejs" -->
<script>
    showAuthModal("registerModal");
</script>
```

Lalu Mana File "views/index/index.js.ejs"?

File "views/index/index.js.ejs" tidak ada, karena saya tidak membuatnya.

Alasannya, saya ingin memberikan 2 cara berbeda untuk mengangani file JavaScript dari server:

- dengan cara yang memungkinkan dinamis (pada pembahasan sebelum ini)
- dengan cara statis

Kode yang ada di "index.ejs" ini juga tadinya ada di server (di file "views/index/index.ejs"), tapi tidak bisa diubah oleh server secara wajar:

```
<script>
    $(document).ready(function() {
      console.log(location.pathname)
      $(`a[href*='${location.pathname}']`).addClass("active");

      $("#current-year").text(new Date().getFullYear().toString())
    });
  </script>
```

Karena itu di "controllers/index.js":

```
// render dan kirimkan datanya yang berupa isi tabel
        res.render("index/index.ejs", {
            text: text,
            skills: skills,
            services: services,
            carousels: carousels,
            portfolios: portfolios,
        });
```

Tidak ada variabel "clientScript" seperti yang lainnya.

Penutup

Artikel ini sangat panjang...

Tapi dengan itu, saya berharap Anda bisa lebih paham tentang kode yang saya tulis.

Sekarang, giliran Anda untuk mencoba memahami lebih dalam tentang kode project company profile ini dengan membaca kodenya sambil mencari info lebih lengkap lagi.

Entah itu melalui dokumentasi, search engine, AI chatbot, dan lain-lain.

Sekian...