npm install --save express-session bunyan

Buat file src > models > user.model.js

```
export default class User {
    constructor(userId, userName, userPassword, user
        this.userId = userId;
        this.userName = userName;
        this.userPassword = userPassword;
        this.userFullName = userFullName;
    }
}
```

Buat file src > models > userInfo.model.js

```
export default class UserInfo {
    constructor(id, address, phoneNo) {
        this.id = id;
        this.address = address;
        this.phoneNo = phoneNo;
    }
}
```

FullName) {

Buat file src > entities > user.schema.js

```
import User from "../models/User.model";
import {EntitySchema} from "typeorm";

const UserSchema = new EntitySchema({
    name: 'User',
    target: User,
    tableName: 'master_user',
    columns: {
        userId: {
            primary: true,
            type: 'int',
            generated: true,
            name: 'user_id'
        },
        userName: {
            name: 'user_name',
            type: 'varchar'
        },
        userPassword: {
            name: 'user_password',
            type: 'varchar'
        },
        userFullName: {
            name: 'user_full_name',
            type: 'varchar'
        }
    },
    relations: {
        userInfo: {
            target: 'UserInfo',
            type: 'one-to-one',
            cascade: true,
            // eager:true,
            joinColumn: {name: 'user_info_id'}
        }
    }
});
export default UserSchema
```

Buat file src > entities > userInfo.schema.js

```
import UserInfo from "../models/UserInfo.model";
import {EntitySchema} from "typeorm";

const UserInfoSchema = new EntitySchema({
    name: 'UserInfo',
    target: UserInfo,
    tableName: 'master_user_info',
    columns: {
        id: {
            primary: true,
            type: 'int',
            generated: true
        },
        address: {
            type: 'varchar'
        },
        phoneNo: {
            name: 'phone_no',
            type: 'varchar'
        }
    }
});
export default UserInfoSchema
```

Buat file src > repository > user.repository.js

```
import {getRepository} from "typeorm";
import UserSchema from "../entities/user.schema";

export default class UserRepository {
    userRepository() {
        return getRepository(UserSchema);
    }

    async findOne(id) {
        const user = await
            this.userRepository()
                .createQueryBuilder("user")
                .innerJoinAndSelect("user.userInfo
                .where("user.userId = :id", {id: i
                .getOne();
        return user;
    }

    async findByUserNameAndPassword(userName, pass
        return await this.userRepository().find({
            where: {userName: userName, userPasswo
            relations: ["userInfo"]
        })
    }
```

```
async findAll(skip = 0, take = 10) {
        const user = await
            this.userRepository()
                .createQueryBuilder("user")
                .innerJoinAndSelect("user.userInfo", "
                .skip(skip)
                .take(take)
                .getMany();
        return user;
    }

    async save(data) {
        const user = this.userRepository().create(data
        return await
            this.userRepository().save(user);
    }
}
```

```
userInfo")




);
```

Buat file src > services > user.service.js

```
import UserRepository from "../repository/user.repository";

class UserService {

    findUser(id) {
        return new UserRepository().findOne(id);
    }

    findAllUser(skip = 0, take = 10) {
        return new UserRepository().findAll(skip, take);
        return user;
    }

    createUser(data) {
        return new UserRepository().save(data);
    }
}

export default UserService;
```

Buat file src > logger.js

```
import bunyan from "bunyan";

export const log = bunyan.createLogger({
    name: 'hello-world-db',
    streams: [
        {
            level: 'info',
            stream: process.stdout
        },
        {
            level: 'error',
            path: '/Users/edwardsuwirya/Downloads/
        }
    ],
    level: 'info'
});
```

Update file src > middlewares > app-middleware.js

```
import express from 'express';
import session from 'express-session';

export default express.Router()
    .use(session({
        secret: 'hello-secret',
        resave: false,
        rolling: true,
        saveUninitialized: true,
        cookie: {maxAge: 15000}
    }))
    .use(express.json());
```

Buat file src > routes > user.router.js

```
import {Router} from "express";
import UserService from "../services/user.service";

const userService = new UserService();
const UserRouter = Router()
    .get('/', async (req, res) => {
        const users = await userService.findAllUser(req.query.skip, req.query
        res.json(users);
    })
    .get('/:id', async (req, res) => {
        let users = null;
        if (req.params.id) {
            users = await userService.findUser(req.params.id);
        }
        res.json(users);
    }).post('/', async (req, res) => {
        let user = {...req.body};
        const users = await userService.createUser(user);

        res.json(users);
    });

export default UserRouter;
```

```
.take);
```

Buat file src > routes > auth.router.js

```
import {Router} from "express";
import AuthService from "../services/auth.service";
import {log} from '../logger';

const AuthRouter = Router()
    .post('/', async (req, res) => {
        let user = {...req.body};
        const userInfo = await new AuthService().doAuth(user);
        if (userInfo.length > 0) {
            req.session.name = JSON.stringify(userInfo);
            log.info({transType: 'USERAUTH-SUCCESS'}, userInfo);
            res.json({message: 'Successfully Authenticated'});
        } else {
            log.error({transType: 'USERAUTH-FAILED'}, user);
            res.sendStatus(401);
        }

    });

export default AuthRouter;
```

Update file src > routes > index.js

```
import express from 'express';
import ProductRouter from "./product.router";
import CategoryRouter from "./category.router";
import UserRouter from "./user.router";
import AuthRouter from "./auth.router";
import {log} from '../logger';

export default express.Router()
    .use('/auth', AuthRouter)
    .use(function (req, res, next) {
        if (req.session.name) {
            next()
        } else {
            res.sendStatus(401);
        }
    })
    .use(function (req, res, next) {
        log.info({transType: 'USERACCESS-SUCCESS'}, req);
        next();
    })
    .use('/product', ProductRouter)
    .use('/category', CategoryRouter)
    .use('/user', UserRouter)

    .use(function (req, res, next) {
        res.body = res.body + "modified";

        next();
    })
    .use((req, res, next) => {
        log.error({transType: 'USERACCESS-FAILED'}, req);
        res.status(404).json({message: 'Not Found.'});
    });
```