**Koleksi Tas**

**Dokumentasi Web Programming**

const express = require('express');

const bodyParser = require('body-parser');

const session = require('express-session');

const flash = require("connect-flash");

const passport = require("passport");

Menginstall package yang dibutuhkan

const app = express();

Membuat instance dari express

const db = require('./public/js/db.js');

Agar root JS dapat membaca database yang akan di connect-kan ke website

require('./public/js/passport')(passport);

Untuk handle autentikasi

app.set('view engine', 'ejs');

Setting folder “view” untuk membaca EJS

app.use(bodyParser.urlencoded());

Fungsinya untuk menguraikan isi request dari body buat dimasukkan ke dalam database

app.use(express.static('public'));

Mengatur folder “public” sebagai static file

app.use(

  session({

    secret: "rahasia",

    resave: true,

    saveUninitialized: true,

  })

);

Inisialisasi session untuk keperluan passport.js

app.use(passport.initialize());

app.use(passport.session());

Menyiapkan initialize dan session untuk passport.js

app.use(flash());

Memasangkan middleware flash

app.use((req, res, next) => {

  res.locals.success\_msg = req.flash("success\_msg");

  res.locals.error\_msg = req.flash("error\_msg");

  res.locals.error = req.flash("error");

  next();

});

Untuk menyimpan variabel error dan informasi lainnya

const index = require('./routes/index');

const user = require('./routes/user');

const upload = require('./routes/upload');

const collect = require('./routes/collection');

const taspria = require('./routes/taspria');

const taswanita = require('./routes/taswanita');

Inisialisasi file routes index, user, upload, collect, taspria, taswanita

app.use('/', index);

app.use('/user', user);

app.use('/upload', upload);

app.use('/collection', collect);

app.use('/taspria', taspria);

app.use('/taswanita', taswanita);

Mengikat route yang akan digunakan dengan file route yang telah disiapkan sebelumnya

let PORT = process.env.PORT || 3000;

app.listen(PORT);

console.log('Server runs at port 3000...');

Dapat menggunakan port local 3000, dan port jika sudah melakukan hosting

const mongoose = require('mongoose');

Mengimport modul mongoose

const dbUrl = process.env.MONGODB\_URI || 'mongodb+srv://User:user@cluster0.iyla9.mongodb.net/db\_koleksi\_tas?retryWrites=true&w=majority';

Melakukan connect ke database terhadap connection string dari mongodb compass

const db = mongoose.connection;

Melakukan inisialisasi untuk melakukan connect ke mongoose

<!DOCTYPE html>

<html lang="en">

    <head>

        <meta charset="UTF-8">

        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z" crossorigin="anonymous">

        <link rel="stylesheet" type="text/css" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.14.0/css/all.min.css">

        <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></script>

        <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js" integrity="sha384-9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN" crossorigin="anonymous"></script>

        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js" integrity="sha384-B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV" crossorigin="anonymous"></script>

        <link rel="stylesheet" href="../css/login.css">

        <script src="https://code.jquery.com/jquery-3.5.1.min.js" ></script>

Link bootstrap, css, keperluan JS

        <title>Login Page</title>

    </head>

    <body>

      <section class="forms-section">

        <h1 class="section-title">Gilvy Collection</h1>

Judul sectionnya

          <div class="forms">

              <ul class="nav nav-pills card-header-pills">

                <li class="nav-item is-active">

                  <a class="nav-link" href="/user/login">Login</a>

                </li>

                <li class="nav-item">

                  <a class="nav-link" href="/user/register">Register</a>

                </li>

                <li class="nav-item">

                  <a class="nav-link" href="/upload/admin">Admin</a>

                </li>

              </ul>

            </div>

Buat navbar login, registrasi dan admin

            <form id="loginacc" action="/user/login" method="POST">

            <div class="form-group">

              <label for="inputEmail">Email</label>

              <input type="email" id="email" name="email" class="form-control" aria-describedby="emailHelp"

              placeholder="Enter Email" >

            </div>

            <div class="form-group">

              <label for="inputPassword">Password</label>

              <input type="password" id="password" name="password" class="form-control"

              placeholder="Enter Password">

            </div>

Bikin form buat tempat input email dan password loginnya

            <table>

              <tr>

                <td>

                  <%- include ("../partials/messages"); %>

Buat manggil messagenya ketika disubmit

                  <button type="submit" class="btn btn-secondary bg-dark" style="color: white;">Login</button>

Button submit loginnya

                </td>

              </tr>

            </table>

          </form>

        </div>

      </section>

    </body>

</html>

<!DOCTYPE html>

<html lang="en">

    <head>

        <meta charset="UTF-8">

        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z" crossorigin="anonymous">

        <link rel="stylesheet" type="text/css" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.14.0/css/all.min.css">

        <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></script>

        <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js" integrity="sha384-9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN" crossorigin="anonymous"></script>

        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js" integrity="sha384-B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV" crossorigin="anonymous"></script>

        <link rel="stylesheet" href="../css/register.css">

        <script src="https://code.jquery.com/jquery-3.5.1.min.js" ></script>

Link bootstrap, css, keperluan JS

        <title>Register Page</title>

    </head>

    <body>

      <section class="forms-section">

        <h1 class="section-title">Gilvy Collection</h1>

Judul sectionnya

          <div class="forms">

              <ul class="nav nav-pills card-header-pills">

                <li class="nav-item">

                  <a class="nav-link" href="/user/login">Login</a>

                </li>

                <li class="nav-item is-active">

                  <a class="nav-link" href="/user/register">Register</a>

                </li>

                <li class="nav-item">

                  <a class="nav-link" href="/upload/admin">Admin</a>

                </li>

              </ul>

            </div>

Buat navbar login, regis sm admin

          <form id="registeracc" action="/user/register" method="POST">

            <div class="form-group">

              <label for="inputNama">Nama</label>

              <input type="text" id="nama" name="nama" class="form-control"

              placeholder="Enter Name" value="<%= typeof nama != 'undefined' ? nama : '' %>">

            </div>

Buat form registrasi, Label input nama dan tempat inputnya

            <div class="form-group">

              <label for="inputEmail">Email</label>

              <input type="email" id="email" name="email" class="form-control" aria-describedby="emailHelp"

              placeholder="Enter Email" value="<%= typeof email != 'undefined' ? email : '' %>">

            </div>

Label input email dan tempat inputnya

            <div class="form-group">

              <label for="inputPassword1">Password</label>

              <input type="password" id="password1" name="password1" class="form-control"

              placeholder="Create Password" value="<%= typeof password1 != 'undefined' ? password1 : '' %>">

            </div>

Label input password1 dan tempat inputnya

            <div class="form-group">

              <label for="inputPassword2">Confirm Password</label>

              <input type="password" id="password2" name="password2" class="form-control"

              placeholder="Confirm Password" value="<%= typeof password2 != 'undefined' ? password2 : '' %>">

            </div>

            <table>

              <tr>

                <td>

Label input password1 dan tempat inputnya

                  <%- include ("../partials/messages"); %>

Buat manggil messagenya ketika disubmit

                  <button type="submit" class="btn btn-secondary bg-dark" style="color: white;">Register</a></button>

Button submit loginnya

                </td>

              </tr>

            </table>

          </form>

        </div>

      </section>

    </body>

</html

const express = require('express');

Menginstall package yang dibutuhkan

const router = express.Router();

const { ensureAuthenticated } = require("../public/js/auth");

const { route } = require('./upload');

router.get("/", ensureAuthenticated, (req, res) =>

  res.render("pages/home", {

    nama: req.user.nama,

  })

);

Untuk mengalihkan dia pindah ke home

router.get("/about", (req, res) =>

  res.render("pages/about")

);

Untuk mengalihkan dia pindah ke about

router.get("/collection", (req, res) =>

  res.render("pages/collection")

);

Untuk mengalihkan dia pindah ke collection

const mongoose = require("mongoose"),

    userSchema = mongoose.Schema({

        nama: {

            type: String,

            required: true,

        },

        email: {

            type: String,

            required: true,

        },

        password: {

            type: String,

            required: true,

        }

    });

module.exports = mongoose.model("User", userSchema);

Modul akun dengan nama, email, dan passwordnya dalam tipe string

const express = require('express');

const router = express.Router();

const bcrypt = require("bcryptjs");

const passport = require("passport");

Import packages

const User = require('../public/js/account')

router.get("/login", (req, res) => res.render("pages/login"));

router.get("/register", (req, res) => res.render("pages/register"));

router.post("/login", (req, res, next) => {

    passport.authenticate("local", {

      successRedirect: "/",

      failureRedirect: "/user/login",

      failureFlash: true,

    })(req, res, next);

  });

Login handle, untuk mengatur kelanjutan dari loginnya ketika dia berhasil maupun gagal

router.post('/register', async (req, res) => {

    const email = req.body.email;

    const nama = req.body.nama;

    const password1 = req.body.password1;

    const password2 = req.body.password2;

    let errors = []

    if (!nama || !email || !password1 || !password2) {

        errors.push({ msg: "Harap isi semua data yang di minta" });

    }

Untuk mengeluarkan message jika terdapat tempat input yang masih kosong

    if (password1 !== password2) {

        errors.push({ msg: "Password tidak sama" });

    }

Untuk keluarin message jika passwordnya tidak sama

    if (errors.length > 0) {

        res.render("pages/register", {

          errors,

          nama,

          email,

          password1,

          password2,

        });

    }

    else {

        User.findOne({ email: email }).then((user) => {

          if (user) {

            errors.push({ msg: "Email sudah terdaftar" });

            res.render("pages/register", {

              errors,

              nama,

              email,

              password1,

              password2,

            });

          }

Validasi emailnya apakah sudah terdaftar atau belum, klo sudah terdaftar nnti keluar messagenya

          else {

            bcrypt.genSalt(10, (err, salt) =>

              bcrypt.hash(password1, salt, (err, hash) => {

                if (err) throw err;

                const newUser = new User({

                  nama,

                  email,

                  password : hash,

                });

Set password jadi hashuntuk mengenkripsi password

                newUser

                  .save()

                  .then((user) => {

                    req.flash(

                      "success\_msg",

                      "Anda berhasil registrasi, Silahkan Login"

                    );

                    res.redirect("/user/login");

                  })

                  .catch((err) => console.log(err));

              })

            );

          }

        });

  }

});

Untuk menyimpan user dan mengerluarkan message

module.exports = {

    ensureAuthenticated: function (req, res, next) {

      if (req.isAuthenticated()) {

        return next();

      }

      res.render('pages/login', {layout: false});

    },

};

Untuk mengecek apakah autentifikasinya berhasil atau tidak, jika berhasil nanti dia bakal lanjut

const LocalStrategy = require("passport-local").Strategy;

const db = require('../js/db')

const bcrypt = require("bcryptjs");

const User = require('../js/account')

Inisialisasi untuk membaca dari JS, mengimport package yang diperlukan

module.exports = function (passport) {

  passport.use(

    new LocalStrategy({ usernameField: "email" }, (email, password, done) => {

      //user cocok

      User.findOne({ email: email })

        .then((user) => {

          if (!user) {

            return done(null, false, { message: "Email tidak terdaftar" });

          }

Untuk mengecek email dalam database apakah terdaftar atau tidak

          //cek password

          bcrypt.compare(password, user.password, (err, isMatch) => {

            if (err) throw err;

            if (isMatch) {

              return done(null, user);

            } else {

              return done(null, false, { message: "Password salah" });

            }

          });

        })

        .catch((err) => console.log(err));

    })

  );

Untuk mengecek password di dalam database benar atau salah dengan yang diinput

  passport.serializeUser(function (user, done) {

    done(null, user.id);

  });

Untuk mengubah agar dapat disimpan dalam file

  passport.deserializeUser(function (id, done) {

    User.findById(id, function (err, user) {

      done(err, user);

    });

  });

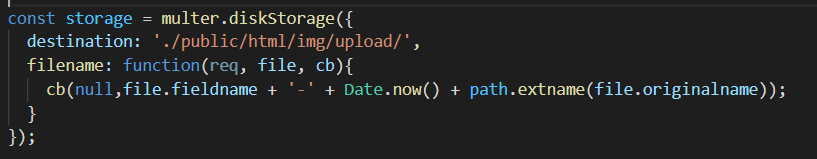
};

Kebalikan dari serialize

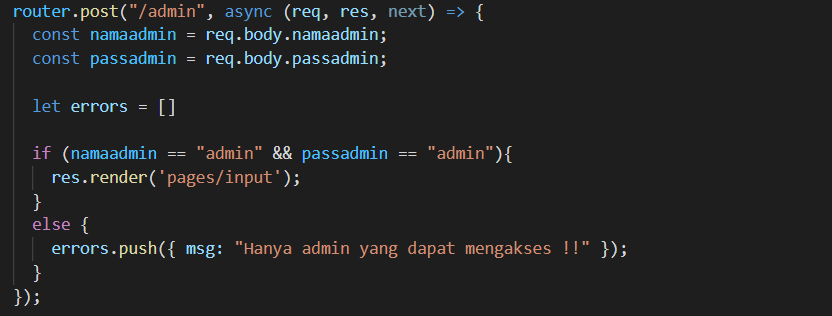
const multer = require('multer');

const path = require('path');

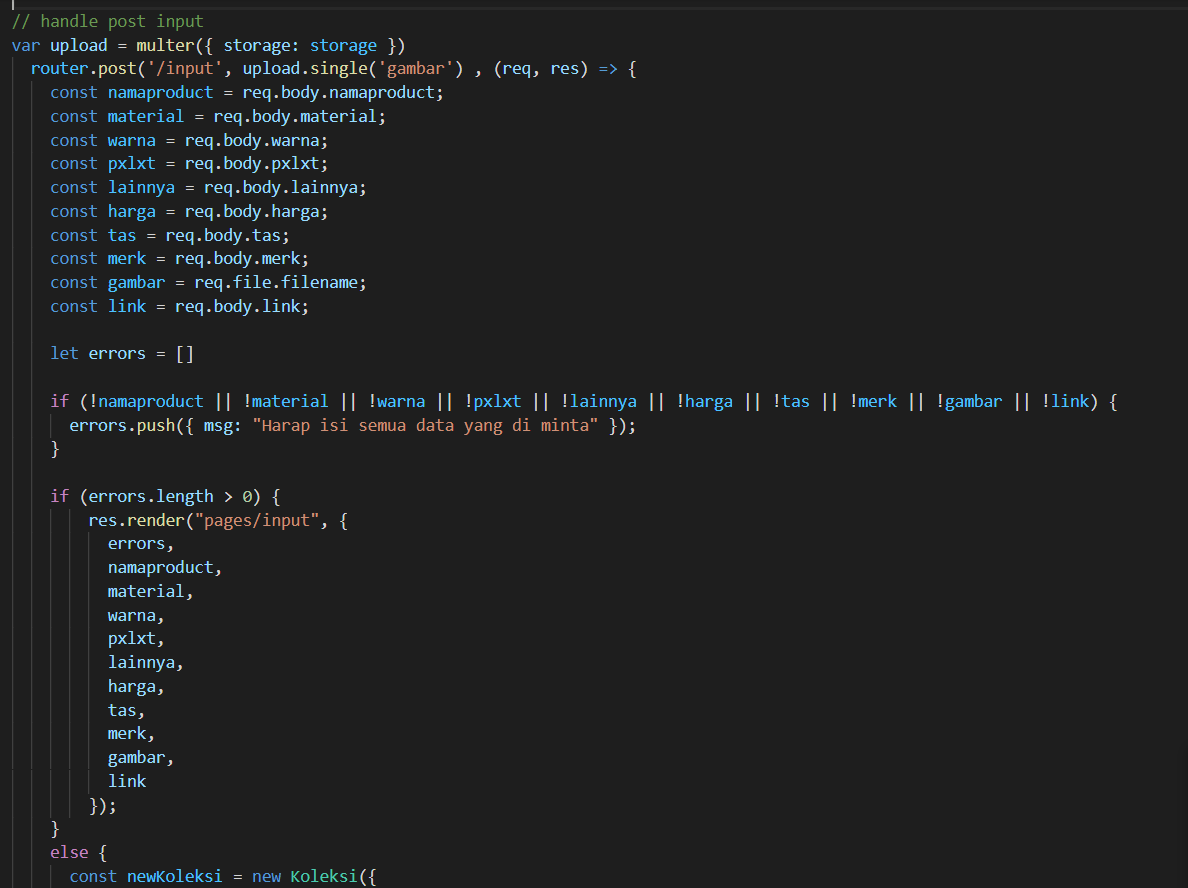
Menginstall package path danmulter



Membuat inisialisasi untuk storage multerdan path dan file namenya



Melakukan method post untuk pengecekan input sesuai atau tidak





Untuk menyimpan document baru ke dalam database

router.get("/input", (req, res) => res.render("pages/input"));

router.get("/admin", (req, res) => res.render("pages/admin"));

Melakukan method get untuk menampilkan view dari file input pages dan admin pages

const mongoose = require("mongoose"),

    koleksiSchema = mongoose.Schema({

        namaproduct: {

            type: String,

            required: true,

        },

        material: {

            type: String,

            required: true,

        },

        warna: {

            type: Object,

            required: true,

        },

        pxlxt: {

            type: String,

            required: true,

        },

        lainnya: {

            type: String,

            required: true,

        },

        harga: {

            type: String,

            required: true,

        },

        gambar: {

            type: String,

            required: true,

        },

        tas: {

            type: String,

            required: true,

        },

        merk: {

            type: String,

            required: true,

        },

        link:{

            type: String,

            required: true,

        }

    });

module.exports = mongoose.model("Koleksi", koleksiSchema);

Schema untuk koleksi



Menginisialisasi koleksi untuk membaca isi dari input.js

const express = require('express');

const router = express.Router();

Menginstall package yang dibutuhkan

const dbKoleksi = require('../public/js/input');

Inisialisasi dbKoleksi untuk membaca isi dari input.js

router.get("/taspria", (req, res) => {

    tas = dbKoleksi.find({merk: 'hpuppies', tas: 'pria'}).exec((error, data) => {

    if (data) {

        res.render("../public/html/tas pria/hush-puppies", {tas: data});

        console.log(JSON.stringify(data));

    }

    });

});

router.get("/taswanita", (req, res) => {

    tas = dbKoleksi.find({merk: 'hpuppies', tas: 'wanita'}).exec((error, data) => {

    if (data) {

        res.render("../public/html/tas wanita/hush-puppies", {tas: data});

        console.log(JSON.stringify(data));

    }

    });

});

router.get("/hush-puppies", (req, res) => {

    tas = dbKoleksi.find({merk: 'hpuppies', tas: 'pria'}).exec((error, data) => {

    if (data) {

        res.render("../public/html/tas pria/hush-puppies", {tas: data});

        console.log(JSON.stringify(data));

    }

    });

});

router.get("/burberry", (req, res) => {

    tas = dbKoleksi.find({merk: 'burberry', tas: 'pria'}).exec((error, data) => {

    if (data) {

        res.render("../public/html/tas pria/burberry", {tas: data});

        console.log(JSON.stringify(data));

    }

    });

});

router.get("/coach", (req, res) => {

    tas = dbKoleksi.find({merk: 'coach', tas: 'pria'}).exec((error, data) => {

    if (data) {

        res.render("../public/html/tas pria/coach", {tas: data});

        console.log(JSON.stringify(data));

    }

    });

});

router.get("/fossil", (req, res) => {

    tas = dbKoleksi.find({merk: 'fossil', tas: 'pria'}).exec((error, data) => {

    if (data) {

        res.render("../public/html/tas pria/fossil", {tas: data});

        console.log(JSON.stringify(data));

    }

    });

});

router.get("/guess", (req, res) => {

    tas = dbKoleksi.find({merk: 'guess', tas: 'pria'}).exec((error, data) => {

    if (data) {

        res.render("../public/html/tas pria/guess", {tas: data});

        console.log(JSON.stringify(data));

    }

    });

});

router.get("/hermes", (req, res) => {

    tas = dbKoleksi.find({merk: 'hermes', tas: 'pria'}).exec((error, data) => {

    if (data) {

        res.render("../public/html/tas pria/hermes", {tas: data});

        console.log(JSON.stringify(data));

    }

    });

});

router.get("/louisvuitton", (req, res) => {

    tas = dbKoleksi.find({merk: 'lvuitton', tas: 'pria'}).exec((error, data) => {

    if (data) {

        res.render("../public/html/tas pria/louisvuitton", {tas: data});

        console.log(JSON.stringify(data));

    }

    });

});

Untuk tiap merk, dan tas yang berdasarkan kategori “Merk” dan “Tas” akan di tampilkan di tiap ejs sesuai yang di render

module.exports = router;

Mengeksport router

const router = express.Router();

Menginisialisasi router untuk menggunakan router dari express

const dbKoleksi = require('../public/js/input');

Inisialisasi dbKoleksi untuk membaca isi dari input.js

router.get("/hush-puppies", (req, res) => {

    tas = dbKoleksi.find({merk: 'hpuppies', tas: 'wanita'}).exec((error, data) => {

    if (data) {

        res.render("../public/html/tas wanita/hush-puppies", {tas: data});

        console.log(JSON.stringify(data));

    }

    });

});

router.get("/burberry", (req, res) => {

    tas = dbKoleksi.find({merk: 'burberry', tas: 'wanita'}).exec((error, data) => {

    if (data) {

        res.render("../public/html/tas wanita/burberry", {tas: data});

        console.log(JSON.stringify(data));

    }

    });

});

router.get("/gucci", (req, res) => {

    tas = dbKoleksi.find({merk: 'gucci', tas: 'wanita'}).exec((error, data) => {

    if (data) {

        res.render("../public/html/tas wanita/gucci", {tas: data});

        console.log(JSON.stringify(data));

    }

    });

});

router.get("/chanel", (req, res) => {

    tas = dbKoleksi.find({merk: 'chanel', tas: 'wanita'}).exec((error, data) => {

    if (data) {

        res.render("../public/html/tas wanita/chanel", {tas: data});

        console.log(JSON.stringify(data));

    }

    });

});

router.get("/prada", (req, res) => {

    tas = dbKoleksi.find({merk: 'prada', tas: 'wanita'}).exec((error, data) => {

    if (data) {

        res.render("../public/html/tas wanita/prada", {tas: data});

        console.log(JSON.stringify(data));

    }

    });

});

Untuk tiap merk, dan tas yang berdasarkan kategori “Merk” dan “Tas” akan di tampilkan di tiap EJS sesuai yang di render