**Encrypting Password**

npm install --save bcrypt

const bcrypt = require('bcrypt');

User.findOne({email: email})

.then(user => {

    if(user) {

        return res.redirect('/signup');

    }

    return bcrypt.hash(password, 12)

    .then(hashedPassword => {

        const newUser = new User({

            email: email,

            password: hashedPassword

        })

        return newUser.save();

    })

    .then(result => {

        return res.redirect('/login');

    });

})

.catch(err => console.log('Error signing up user', err));

**Authenticating Users**

User.findOne({email: email})

.then(user => {

    if(!user) {

        return res.redirect('/login');

    }

    bcrypt.compare(password, user.password)

    .then(matched => {

        if(matched) {

            req.session.isAuthenticated = true;

            req.session.user = user;

            return res.redirect('/products');

        }

        return res.redirect('/login');

    })

})

.catch(err => console.log('Error siging in user', err));

**Route Protection**

// auth-guard.js

exports.loginRequired = (req, res, next) => {

    if(!req.session.isAuthenticated) {

        return res.redirect('/login');

    }

    next();

}

Now this loginRequired can be used as first middlewares before calling the handles in the routes as

const express = require('express');

const authController = require('../controllers/admin');

const authGuard = require('../middlewares/auth-guard');

const router = express.Router();

router.get('/products', authGuard.loginRequired, authController.getProducts);

**Using CSRF Token**

**npm install --save csurf**

// app.js

const csrf = require('csurf');

const csrfProtection = csrf();

app.use(csrfProtection); // after we initialize the session

Now we want to have the csrf token on every view(page).

There is a special feature of express **res.locals** which provides us some local variable to be passed to the views. Locals because they are only passed in the views which are currently being rendered.

exports.resLocals = (req, res, next) => {

    res.locals.isAuthenticated = req.session.isAuthenticated;

    res.locals.csrfToken = req.csrfToken();

    next();

}

Add this middleware before all the routes as

app.use(locals.resLocals);

app.use(authRoutes);

app.use('/admin', adminRoutes);

...

**Providing User Feedback**

**npm install --save connect-flash**

// app.js

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

app.use(flash()); // after initializing session

// It can now be used in the handler as in auth controller

req.flash('error', 'Email exists already. Please use a different one.');