Programme Name: BCS



Course Code: (ISYS3100)

Course Name: ENTERPRISE SYSTEM

Assignment / Lab Sheet / Project / Case Study No: 1

Date of Submission: Jan 2 2023

**Submitted By: Submitted To:**

Student Name: Laxman Sharma Faculty Name: Bibek Gautam

IUKL ID: 042201900017 Department: Program Office

Semester: 4th Semester

Intake: March 2022

**VIews/App.js**

const express = require('express');

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

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

const bcrypt = require('bcrypt');

const User = require('./models/user');

const UserData = require('./models/userData');

const app = express();

const port = 3000;

app.set('view engine', 'ejs');

app.use(bodyParser.urlencoded({ extended: true }));

app.use(

session({

secret: 'secret-key',

resave: false,

saveUninitialized: true,

})

);

const isAuthenticated = (req, res, next) => {

if (req.session && req.session.user) {

return next();

}

res.redirect('/login');

};

// Routes

app.get('/', isAuthenticated, async (req, res) => {

try {

const usersData = await UserData.findAll();

res.render('index', { user: req.session.user, users: usersData });

} catch (error) {

console.error('Error fetching user data:', error);

res.status(500).send('Internal Server Error');

}

});

app.get('/signup', (req, res) => {

res.render('signup');

});

app.post('/signup', async (req, res) => {

const { username, password } = req.body;

const hashedPassword = await bcrypt.hash(password, 10);

try {

await User.create({

username,

password: hashedPassword,

});

res.redirect('/login');

} catch (error) {

console.error('Error creating user:', error);

res.redirect('/signup');

}

});

app.get('/login', (req, res) => {

res.render('login');

});

app.post('/login', async (req, res) => {

const { username, password } = req.body;

try {

const user = await User.findOne({

where: {

username: username,

},

});

if (user && (await bcrypt.compare(password, user.password))) {

req.session.user = username;

res.redirect('/');

} else {

res.redirect('/login');

}

} catch (error) {

console.error('Error finding user:', error);

res.status(500).send('Internal Server Error');

}

});

app.get('/logout', (req, res) => {

req.session.destroy((err) => {

res.redirect('/login');

});

});

app.get('/create', isAuthenticated, (req, res) => {

res.render('create');

});

app.post('/create', isAuthenticated, async (req, res) => {

const { name, address, email, phone, profession } = req.body;

try {

// Create a new user data entry in the database

await UserData.create({

name,

address,

email,

phone,

profession,

});

res.redirect('/');

} catch (error) {

console.error('Error creating user data entry:', error);

res.status(500).send('Internal Server Error');

}

});

app.get('/user/:id', isAuthenticated, async (req, res) => {

const userId = parseInt(req.params.id);

try {

const user = await UserData.findByPk(userId);

if (user) {

res.render('user', { user });

} else {

res.status(404).send('User not found');

}

} catch (error) {

console.error('Error finding user data entry:', error);

res.status(500).send('Internal Server Error');

}

});

app.get('/update/:id', isAuthenticated, async (req, res) => {

const userId = parseInt(req.params.id);

try {

const user = await UserData.findByPk(userId);

if (user) {

res.render('update', { user });

} else {

res.status(404).send('User not found');

}

} catch (error) {

console.error('Error finding user data entry:', error);

res.status(500).send('Internal Server Error');

}

});

app.post('/update/:id', isAuthenticated, async (req, res) => {

const userId = parseInt(req.params.id);

const { name, address, email, phone, profession } = req.body;

try {

const user = await UserData.findByPk(userId);

if (user) {

await user.update({

name,

address,

email,

phone,

profession,

});

res.redirect('/');

} else {

res.status(404).send('User not found');

}

} catch (error) {

console.error('Error updating user data entry:', error);

res.status(500).send('Internal Server Error');

}

});

app.get('/delete/:id', isAuthenticated, async (req, res) => {

const userId = parseInt(req.params.id);

try {

const user = await UserData.findByPk(userId);

if (user) {

await user.destroy();

res.redirect('/');

} else {

res.status(404).send('User not found');

}

} catch (error) {

console.error('Error deleting user data entry:', error);

res.status(500).send('Internal Server Error');

}

});

app.listen(port, () => {

console.log(`Server is running on http://localhost:${port}`);

});

**db.js**const { Sequelize } = require('sequelize');

const sequelize = new Sequelize('crud', 'postgres', 'root', {

dialect: 'postgres',

host: 'localhost',

port: 5432,

});

module.exports = sequelize;

**models/index.js**'use strict';

const fs = require('fs');

const path = require('path');

const Sequelize = require('sequelize');

const process = require('process');

const basename = path.basename(\_\_filename);

const env = process.env.NODE\_ENV || 'development';

const config = require(\_\_dirname + '/../config/config.json')[env];

const db = {};

let sequelize;

if (config.use\_env\_variable) {

sequelize = new Sequelize(process.env[config.use\_env\_variable], config);

} else {

sequelize = new Sequelize(config.database, config.username, config.password, config);

}

fs

.readdirSync(\_\_dirname)

.filter(file => {

return (

file.indexOf('.') !== 0 &&

file !== basename &&

file.slice(-3) === '.js' &&

file.indexOf('.test.js') === -1

);

})

.forEach(file => {

const model = require(path.join(\_\_dirname, file))(sequelize, Sequelize.DataTypes);

db[model.name] = model;

});

Object.keys(db).forEach(modelName => {

if (db[modelName].associate) {

db[modelName].associate(db);

}

});

db.sequelize = sequelize;

db.Sequelize = Sequelize;

module.exports = db;

**users.js**

const { DataTypes } = require("sequelize")

const sequelize = require('../db')

const User = sequelize.define('User', {

username: {

type: DataTypes.STRING,

allowNull: false,

unique: true,

},

password: {

type: DataTypes.STRING,

allowNull: false,

},

},{

tableName: 'Users'

});

module.exports = User

**userData.js**

const { DataTypes } = require("sequelize")

const sequelize = require('../db')

const User = sequelize.define('User', {

username: {

type: DataTypes.STRING,

allowNull: false,

unique: true,

},

password: {

type: DataTypes.STRING,

allowNull: false,

},

},{

tableName: 'Users'

});

module.exports = User

**Migration**'use strict';

module.exports = {

up: async (queryInterface, Sequelize) => {

await queryInterface.createTable('UserData', {

id: {

type: Sequelize.INTEGER,

primaryKey: true,

autoIncrement: true,

allowNull: false,

},

name: {

type: Sequelize.STRING,

allowNull: true,

},

address: {

type: Sequelize.STRING,

allowNull: true,

},

email: {

type: Sequelize.STRING,

allowNull: true,

},

phone: {

type: Sequelize.STRING,

allowNull: true,

},

profession: {

type: Sequelize.STRING,

allowNull: true,

},

createdAt: {

type: Sequelize.DATE,

allowNull: false,

},

updatedAt: {

type: Sequelize.DATE,

allowNull: false,

},

});

},

down: async (queryInterface, Sequelize) => {

await queryInterface.dropTable('UserData');

},

};

'use strict';

module.exports = {

up: async (queryInterface, Sequelize) => {

await queryInterface.createTable('Users', {

id: {

type: Sequelize.INTEGER,

primaryKey: true,

autoIncrement: true,

allowNull: false,

},

username: {

type: Sequelize.STRING,

allowNull: false,

unique: true,

},

password: {

type: Sequelize.STRING,

allowNull: false,

},

createdAt: {

type: Sequelize.DATE,

allowNull: false,

},

updatedAt: {

type: Sequelize.DATE,

allowNull: false,

},

});

},

down: async (queryInterface, Sequelize) => {

await queryInterface.dropTable('Users');

},

};







