const async = require('async'); const _ = require('lodash'); const jwt = require('../../config/jwt'); const db = require('../../config/db'); const mysql = require('mysql2/promise'); const bcrypt = require('bcrypt');

/** mmm

- POST: /signin
- { email, password } **/ module.exports.login = (req, res, next) => { let user_id = req.body.user_id; let password = req.body.password; if (!user_id || !password) { return next({ statusCode: 400, message: res.__('invalidParams') }); }

async.waterfall([(nextStep) => { sql = 'SELECT * FROM user WHERE user_id = ' + mysql.escape(user_id); db.connection.query({ sql }).then(rows => { if (!rows[0][0]) next({ statusCode: 401, message: 'not found user' }); else nextStep(null, rows[0][0]); }).catch(nextStep); }, (user, nextStep) => { if (user.userType == 1) { // 관리자는 비밀번호 암호화가 되어있지 않다. let data = _.pick(user, ['user_id', 'name', 'userType', 'createdAt']); let token = jwt.generate(data); if (password == user.password) nextStep(null, { token, data }); else nextStep({ statusCode: 400, message: 'invalid password' }); } else { Authenticate(password, user.password).then(valid => { if (valid) { let data = _.pick(user, ['user_id', 'name', 'userType', 'createdAt']); let token = jwt.generate(data); nextStep(null, { token, data }); //original : nextStep(null, { token, data }); } else nextStep({ statusCode: 400, message: 'invalid password' }); }).catch(nextStep); } }], (err, result) => { if (err) next(err); else { res.json(result); } }); };

Authenticate = (plainText, db_password) => { return new Promise((resolve, reject) => { console.log("plainText", plainText); console.log("password", db_password); bcrypt.compare(plainText, db_password, (err, res) => { if (err) reject(err); else resolve(res); }); }); };

cryptPassword = (plainText) => { return new Promise((resolve, reject) => { bcrypt.genSalt(10, (err, salt) => { if (err) reject(err); else { bcrypt.hash(plainText, salt, (err, hash) => { if (err) reject(err); else resolve(hash); }); } }); }); };

module.exports.signup = (req, res, next) => { console.log(req.body.languages); let userType = parseInt(req.body.userType[0]); let user_id = req.body.userId; let password = req.body.userPassword; let name = req.body.userName; let phone = req.body.phone; let portfolio = ''; let major = ''; let career = ''; let age = ''; let language = ''; let score = ''; let freelancerId = 0;

if (userType == 2) { portfolio = req.file.originalname; file_location = req.file.path; major = req.body.userMajor; career = parseInt(req.body.career.split(' ')[0]); age = parseInt(req.body.age.split(' ')[0]); language = req.body.languages.split(','); score = req.body.scores.split(','); } let sql = ''; let sql1 = ''; let sql2 = ''; async.waterfall([(nextStep) => { let sqlt1 = 'SELECT user_id FROM Administrator WHERE user_id = ' + mysql.escape(user_id); let sqlt2 = 'SELECT user_id FROM Freelancer WHERE user_id = ' + mysql.escape(user_id); let sqlt3 = 'SELECT user_id FROM Client WHERE user_id = ' + mysql.escape(user_id); sql = '(' + sqlt1 + ') UNION (' + sqlt2 + ') UNION (' + sqlt3 + ')'; db.connection.query({ sql }).then(rows => { if (rows[0][0]) next({ statusCode: 401, message: 'ERROR: already existing user' }); else nextStep(null); }).catch(nextStep); }, (nextStep) => { cryptPassword(password).then(hashedValue => { password = hashedValue; if(userType == 3) { let args1 = { user_id: user_id, password: password, name: name, phone: phone, } sql1 = 'Insert INTO Client SET ' + mysql.escape(args1); } else if(userType == 2) { let args1 = { user_id: user_id, password: password, name: name, phone: phone, major: major, age: age, career: career, } sql1 = 'Insert INTO Freelancer SET ' + mysql.escape(args1); } nextStep(null, sql1); }) .catch(nextStep); }, (sql1, nextStep) => { // user table insertion (id,pw,name,phone) db.connection.query(sql1, function(err, result) { if (err) { console.log(err); next({ statusCode: 401, message: 'ERROR: user creating error!' }); } else nextStep(null,

result.insertId); }) .catch(nextStep); }, (id, nextStep) => { freelancerId = id; if (userType == 2) { let l = language.length; let sqlarray = []; let tmparray = []; for (var i = 0; i < l; i++) { tmparray = [language[i], score[i], id]; sqlarray.push(tmparray); } let multiple_sql = 'Insert INTO Free_lang (`type` , `level` , `free_id`) Values ' + mysql.escape(sqlarray); db.connection.query(multiple_sql, function(err, result) { if (err) { console.log(err); next({ statusCode: 401, message: 'ERROR: freelancer language spec insert error!' }); } else nextStep(null, id); }) .catch(nextStep); } else { nextStep(null, id); } }, (id, nextStep) => { if (userType == 2) { let jsonData = { title: portfolio, free_id: id, file_location: file_location, } let blob_sql = 'Insert INTO Ext_portfolio SET ' + mysql.escape(jsonData); db.connection.query(blob_sql, function(err, result) { if (err) { console.log(err); next({ statusCode: 401, message: 'ERROR: freelancer ext_portfolio insert error!' }); } else nextStep(null, id); }) .catch(nextStep); } else { nextStep(null, id); } }, (id, nextStep) => { if (userType == 2) { let jsonData = { name: user_id + " alone team", min_career: career, mgr_id: id, } let tsql = 'Insert INTO Team SET ' + mysql.escape(jsonData); db.connection.query(tsql, function(err, result) { if (err) { console.log(err); next({ statusCode: 401, message: 'ERROR: freelancer alone team insert error!' }); } else nextStep(null, result.insertId); }) .catch(nextStep); } else { nextStep(null, id); } }, (id, nextStep) => { if (userType == 2) { l = language.length; sqlarray = []; tmparray = []; for (var i = 0; i < l; i++) { tmparray = [language[i], score[i], id]; sqlarray.push(tmparray); } let multiple_sql2 = 'Insert INTO Team_lang (`type` , `max_level` , `team_id`) Values ' + mysql.escape(sqlarray); db.connection.query(multiple_sql2, function(err, result) { if (err) { console.log(err); next({ statusCode: 401, message: 'ERROR: freelancer alone team lang insert error!' }); } else nextStep(null, id); }) .catch(nextStep); } else { nextStep(null, id); } }, (id, nextStep) => { if (userType == 2) { let jsonData = { team_id: id, member_id: freelancerId, } let tmsql = 'Insert INTO Team_member SET ' + mysql.escape(jsonData); db.connection.query(tmsql, function(err, result) { if (err) { console.log(err); next({ statusCode: 401, message: 'ERROR: freelancer alone team member insert error!' }); } else nextStep(null, result); }) .catch(nextStep); } else { nextStep(null); } },], (err, result) => { if (err) next(err); else res.json(result); }); };

module.exports.getFreelancerInfo = (req, res, next) => { let id = req.body.id; async.waterfall([(nextStep) => { let sql = "select f.id,f.career,f.major,f.age,l.type,l.level, e.title, e.file_location from Freelancer as f join Free_lang as l on f.id = l.free_id join Ext_portfolio as e on f.id = e.free_id where f.id = " + mysql.escape(id); db.connection.query(sql, function(err, result) { if (err) next({ statusCode: 401, message: 'no users' }); else nextStep(null, result); }) .catch(nextStep); },], (err, result) => { if (err) next(err); else res.send(result); }); };

module.exports.getMyTeam = (req, res, next) => { let id = req.body.id; let team_array = []; let member_array = []; let ids = []; async.waterfall([(nextStep) => { let sql = "select t.id,t.name,t.min_career,t.mgr_id,tl.type,tl.max_level from Freelancer as f join Team_member as tm on tm.member_id = f.id join Team as t on t.id = tm.team_id join Team_lang as tl on t.id = tl.team_id where f.id = " + mysql.escape(id); db.connection.query(sql, function(err, result) { if (err) next({ statusCode: 401, message: 'no users' }); else { team_array = result; for(var i=0;i<result.length;i++) { if(ids.indexOf(result[i].id) == -1) ids.push(result[i].id); } nextStep(null); } }) .catch(nextStep); }, (nextStep) => { if(team_array.length == 0) nextStep(null, []); else { let whereSql = ''; for(var i=0;i<ids.length;i++) { if(i == ids.length-1) whereSql += 't.id = ' + mysql.escape(ids[i]); else whereSql += 't.id = ' + mysql.escape(ids[i]) + ' or '; } let sql2 = "select t.id as team_id,f.id as member_id,f.user_id,f.name,f.age,f.career,f.grade_sum,f.request_count from Team as t join Team_member as tm on tm.team_id = t.id join Freelancer as f on tm.member_id = f.id where " + whereSql; db.connection.query(sql2, function(err, result) { if (err) next({ statusCode: 401, message: 'no users' }); else { member_array = result; nextStep(null, [team_array,member_array]); } }) .catch(nextStep); }

```
  },
```

], (err, result) => { if (err) next(err); else res.send(result); }); };

module.exports.getAllTeam = (req, res, next) => { let team_array = []; let member_array = []; let ids = []; async.waterfall([(nextStep) => { let sql = "select t.id,t.name,t.min_career,t.mgr_id,tl.type,tl.max_level from Freelancer as f join Team_member as tm on tm.member_id = f.id join Team as t on t.id = tm.team_id join Team_lang as tl on t.id = tl.team_id"; db.connection.query(sql, function(err, result) { if (err) next({ statusCode: 401, message: 'no users' }); else { team_array = result; for(var i=0;i<result.length;i++) { if(ids.indexOf(result[i].id) == -1) ids.push(result[i].id); } nextStep(null); } }) .catch(nextStep); }, (nextStep) => { if(team_array.length == 0) nextStep(null, []); else { let whereSql = ''; for(var i=0;i<ids.length;i++) { if(i == ids.length-1) whereSql += 't.id = ' + mysql.escape(ids[i]); else whereSql += 't.id = ' + mysql.escape(ids[i]) + ' or '; } let sql2 = "select t.id as team_id,f.id as member_id,f.user_id,f.name,f.age,f.career,f.grade_sum,f.request_count from Team as t join Team_member as tm on tm.team_id = t.id join Freelancer as f on tm.member_id = f.id where " + whereSql; db.connection.query(sql2, function(err, result) { if (err) next({ statusCode: 401, message: 'no users' }); else { member_array = result; nextStep(null, [team_array,member_array]); } }) .catch(nextStep); }

```
  },
```

], (err, result) => { if (err) next(err); else res.send(result); }); };

module.exports.getTeamMessage = (req, res, next) => { let tid = req.body.tid; console.log(tid); async.waterfall([(nextStep) => { let sql = 'select r.title, m.* from Message as m join Request as r on r.id = m.request_id where m.team_id = ' + mysql.escape(tid); db.connection.query(sql, function(err, result) { if (err) next({ statusCode: 401, message: 'no messages!' }); else { nextStep(null,result); } }) .catch(nextStep); },], (err, result) => { if (err) next(err); else res.send(result); }); };

module.exports.addTeam = (req, res, next) => { let mgr_id = req.body.mgr_id; let name = req.body.name; let photo = req.body.photo; let uid = req.body.uid; let mgr_uid = req.body.mgr_uid;

let whereSql = ''; let = []; let career = 100; let lang = []; let lv = []; let ids = [];

for(var i=0;i<uid.length;i++) { if(i == uid.length-1) whereSql += 'f.user_id = ' + mysql.escape(uid[i]); else whereSql += 'f.user_id = ' + mysql.escape(uid[i]) + ' or '; } whereSql += ' or f.user_id = ' + mysql.escape(mgr_uid); async.waterfall([(nextStep) => { let sql = 'select f.id, f.career,l.type,l.level from Freelancer as f join Free_lang as l on f.id = l.free_id where ' + whereSql; console.log(sql); db.connection.query(sql, function(err, result) { if (err) next({ statusCode: 401, message: 'no users' }); else { console.log(result); for(var i=0;i<result.length;i++) { if(ids.indexOf(result[i].id) == -1) ids.push(result[i].id); if(result[i].career < career) career = result[i].career; if(lang.indexOf(result[i].type) == -1) { lang.push(result[i].type); lv.push(result[i].level); } else { let index = lang.indexOf(result[i].type); if(lv[index] < result[i].level) lv[index] = result[i].level; } } console.log("career : " + career); console.log("lang : " + lang); console.log("lv : " + lv); nextStep(null); } }) .catch(nextStep); }, (nextStep) => { let jsonData = { name: name, min_career: career, photo: photo, mgr_id: mgr_id } let sql2 = 'Insert INTO Team SET ' + mysql.escape(jsonData); db.connection.query(sql2, function(err, result) { if (err) next({ statusCode: 401, message: 'team adding error!' }); else { nextStep(null,result.insertId); } }) .catch(nextStep); }, (id,nextStep) => { let tmparray = []; let sqlarray = []; for(var i=0;i<ids.length;i++) { tmparray = [id,ids[i]]; sqlarray.push(tmparray); } let sql3 = 'Insert INTO Team_member (`team_id` , `member_id`) Values ' + mysql.escape(sqlarray); console.log(sql3); db.connection.query(sql3, function(err, result) { if (err) next({ statusCode: 401, message: 'team member adding error!' }); else { nextStep(null,id); } }) .catch(nextStep); }, (id,nextStep) => { tmparray = []; sqlarray = []; for(var i=0;i<lang.length;i++) { tmparray = [id,lang[i],lv[i]]; sqlarray.push(tmparray); } let sql4 = 'Insert INTO Team_lang (`team_id` , `type` , `max_level`) Values ' + mysql.escape(sqlarray); console.log(sql4); db.connection.query(sql4, function(err, result) { if (err) next({ statusCode: 401, message: 'team lang adding

error!' }); else { nextStep(null,result); } }) .catch(nextStep); },], (err, result) => { if (err) next(err); else res.send(result); }); }

module.exports.getall = (req, res, next) => { let resultArr = []; async.waterfall([(nextStep) => { let sql = 'Select * from Administrator;' db.connection.query(sql, function(err, result) { if (err) next({ statusCode: 401, message: 'no users' }); else { resultArr.push(result); nextStep(null); } }) .catch(nextStep); }, (nextStep) => { let sql2 = 'Select * from Freelancer;' db.connection.query(sql2, function(err, result) { if (err) next({ statusCode: 401, message: 'no users' }); else { resultArr.push(result); nextStep(null); } }) .catch(nextStep); }, (nextStep) => { let sql3 = 'Select * from Client;' db.connection.query(sql3, function(err, result) { if (err) next({ statusCode: 401, message: 'no users' }); else { resultArr.push(result); nextStep(null, resultArr); } }) .catch(nextStep); },], (err, result) => { if (err) next(err); else res.send(result); }); };

module.exports.deleteUser = (req, res, next) => { let id = req.body.id; let type = req.body.type; console.log(id + "," + type); let sql = '' if(type == 2) sql = 'Delete from Freelancer where id = ' + mysql.escape(id); else sql = 'Delete from Client where id = ' + mysql.escape(id); console.log(sql); async.waterfall([(nextStep) => { db.connection.query(sql, function(err, result) { if (err) next({ statusCode: 401, message: 'no users' }); else nextStep(null, result); }) .catch(nextStep); },], (err, result) => { if (err) next(err); else res.send(result); }); };

module.exports.addRequest = (req, res, next) => { async.waterfall([(nextStep) => { let args1 = { client_user_id: req.body.client_user_id, title: req.body.title, money: req.body.money, start_date: req.body.start_date, end_date: req.body.end_date, people_min: req.body.people_min, people_max: req.body.people_max, career: req.body.career, spec: req.body.spec, } sql1 = 'Insert INTO request SET ' + mysql.escape(args1); db.connection.query(sql1, function(err, result) { if (err) next({ statusCode: 401, message: 'ERROR: adding request error!' }); nextStep(null, result.insertId); }) .catch(nextStep); }, (id, nextStep) => { let l = req.body.language.length; let sqlarray = []; let tmparray = []; for (var i = 0; i < l; i++) { tmparray = [req.body.language[i], req.body.score[i], id, req.body.client_user_id]; console.log(tmparray); sqlarray.push(tmparray); } console.log(sqlarray); let multiple_sql = 'Insert INTO require_languages (`type`, `level`, `request_requestId`, `request_client_user_id`) Values ' + mysql.escape(sqlarray); db.connection.query(sql1, function(err, result) { if (err) next({ statusCode: 401, message: 'ERROR: require languages inserting error!' }); nextStep(null, result.insertId); }) .catch(nextStep); },], (err, result) => { if (err) next(err); else res.send(result); }); };