**/\* User authentication unit test \*/**

var expect = require('chai').expect

var app = require('../app')

var request = require('supertest')

var authenticatedUser = request.agent(app)

before(function (done) {

authenticatedUser

.post('/login')

.send({

username: 'jack',

password: '123456'

})

.end(function (err, res) {

expect(res.statusCode).to.equal(302)

done()

})

})

describe('GET /dashboard', function (done) {

it('should return a 200 response if the user is logged in', function (done) {

authenticatedUser.get('/dashboard')

.expect(200, done)

})

it('should return a 302 response and redirect to /login if user is not logged in', function (done) {

request(app).get('/dashboard')

.expect('Location', '/login')

.expect(302, done);

})

})

Results

> mocha ./test/login

GET /dashboard

√ should return a 200 response if the user is logged in

√ should return a 302 response and redirect to /login if user is not logged in

2 passing (1s)

**/\* Database test script \*/**

var seeder = require('mongoose-seed')

// Connect to MongoDB via Mongoose

mogodb = "mongodb://localhost/dbtest"

seeder.connect(mogodb, {

useNewUrlParser: true

}, function () {

// Load Mongoose models

seeder.loadModels([

'./models/incident\_details.js',

'./models/incident\_reporter.js',

'./models/incidents.js',

'./models/resources.js',

'./models/user.js'

])

// Clear specified collections

seeder.clearModels(['incident\_details', 'incident\_reporter', 'incidents', 'resources', 'user'], function () {

// Callback to populate DB once collections have been cleared

seeder.populateModels(data, function () {

seeder.disconnect();

})

})

})

// Data array containing seed data - documents organized by Model

var data = [{

'model': 'incidents',

'documents': [

{

\_id: "5bda598006b20e02dd8ebe41",

incident\_location: "Choa Chu Kang, Singapore",

incident\_lat: 1.3839803,

incident\_lng: 103.7469611,

incident\_name: "Dengue",

incident\_description: "Fire",

incident\_details: "5bda598006b20e02dd8ebe40",

incident\_reporter: "5bda597f06b20e02dd8ebe3f"

},

{

\_id: "5be05c4c25e0070e22b35638",

incident\_location: "Asia Square Tower 2, Singapore",

incident\_lat: 1.2781459,

incident\_lng: 103.8511983,

incident\_name: "Dengue",

incident\_description: "A",

incident\_details: "5be05c4c25e0070e22b35637",

incident\_reporter: "5be05c4c25e0070e22b35636"

},

{

\_id: "5be030f55790eb591c5c9f90",

incident\_location: "Pipit Road, Singapore",

incident\_lat: 1.3238142,

incident\_lng: 103.8861799,

incident\_name: "Haze",

incident\_description: "Heavy Haze, very smokey",

incident\_details: "5be030f55790eb591c5c9f8f",

incident\_reporter: "5bda597f06b20e02dd8ebe3f"

},

{

\_id: "5be08fe53a639734e30ccea8",

incident\_location: "Choa Chu Kang Avenue 4, Lot One Shoppers' Mall, Singapore",

incident\_lat: 1.3851131,

incident\_lng: 103.744937,

incident\_name: "Fire",

incident\_description: "Fire in mall",

incident\_details: "5be08fe53a639734e30ccea7",

incident\_reporter: "5be08fe53a639734e30ccea6"

},

{

\_id: "5be025e25790eb591c5c9f24",

incident\_location: "Robinson Road, WeWork 71 Robinson, Singapore",

incident\_lat: 1.2785389,

incident\_lng: 103.8488535,

incident\_name: "Dengue",

incident\_description: "3 new dengue cases, dispatch NEA to examine",

incident\_details: "5be025e25790eb591c5c9f23",

incident\_reporter: "5be025e25790eb591c5c9f22"

},

{

\_id: "5be026605790eb591c5c9f2d",

incident\_location: "Tampines Street 22, 280 Indian Muslim Food, Singapore",

incident\_lat: 1.3478031,

incident\_lng: 103.9516125,

incident\_name: "Dengue",

incident\_description: "ad",

incident\_details: "5be06b29f7ef0d1bd9e43d97",

incident\_reporter: "5be06b29f7ef0d1bd9e43d96"

},

{

\_id: "5be026a45790eb591c5c9f30",

incident\_location: "Jurong Town Hall Road, Snow City, Singapore",

incident\_lat: 1.3351431,

incident\_lng: 103.7351617,

incident\_name: "Dengue",

incident\_description: "Alot ad",

incident\_details: "5be026f25790eb591c5c9f35",

incident\_reporter: "5be026f25790eb591c5c9f34"

},

{

\_id: "5be026b65790eb591c5c9f33",

incident\_location: "Singapore",

incident\_lat: 1.352083,

incident\_lng: 103.819836,

incident\_name: "Fire",

incident\_description: "F",

incident\_details: "5be026b65790eb591c5c9f32",

incident\_reporter: "5be026b65790eb591c5c9f31"

}

]

},

{

'model': 'incident\_details',

'documents': [

{

\_id: "5bda598006b20e02dd8ebe40",

incident\_status: "ONGOING",

crisis\_level: 3,

incident\_activities: "None",

date\_created: "2018-11-01T01:40:16.007Z",

date\_resolved: "2018-11-01T01:40:16.007Z"

},

{

\_id: "5be05c4c25e0070e22b35637",

incident\_status: "ONGOING",

crisis\_level: 2,

incident\_activities: "None",

date\_created: "2018-11-01T01:40:16.007Z",

date\_resolved: "2018-11-01T01:40:16.007Z"

},

{

\_id: "5be030f55790eb591c5c9f8f",

incident\_status: "ONGOING",

crisis\_level: 1,

incident\_activities: "None",

date\_created: "2018-11-01T01:40:16.007Z",

date\_resolved: "2018-11-01T01:40:16.007Z"

},

{

\_id: "5be08fe53a639734e30ccea7",

incident\_status: "ONGOING",

crisis\_level: 0,

incident\_activities: "None",

date\_created: "2018-11-01T01:40:16.007Z",

date\_resolved: "2018-11-01T01:40:16.007Z"

},

{

\_id: "5be025e25790eb591c5c9f23",

incident\_status: "ONGOING",

crisis\_level: 0,

incident\_activities: "NEA",

date\_created: "2018-11-01T01:40:16.007Z",

date\_resolved: "2018-11-01T01:40:16.007Z"

},

{

\_id: "5be06b29f7ef0d1bd9e43d97",

incident\_status: "RESOLVED",

crisis\_level: 1,

incident\_activities: "None",

date\_created: "2018-11-01T01:40:16.007Z",

date\_resolved: "2018-11-01T01:40:16.007Z"

},

{

\_id: "5be026f25790eb591c5c9f35",

incident\_status: "RESOLVED",

crisis\_level: 2,

incident\_activities: "None",

date\_created: "2018-11-01T01:40:16.007Z",

date\_resolved: "2018-11-01T01:40:16.007Z"

},

{

\_id: "5be026b65790eb591c5c9f32",

incident\_status: "RESOLVED",

crisis\_level: 3,

incident\_activities: "Police,NEA",

date\_created: "2018-11-01T01:40:16.007Z",

date\_resolved: "2018-11-01T01:40:16.007Z"

}

]

},

{

'model': 'incident\_reporter',

'documents': [

{

\_id: "5bda597f06b20e02dd8ebe3f",

reporter\_name: "jack",

reporter\_email: "place@holder",

reporter\_contact\_number: "123456789"

},

{

\_id: "5be05c4c25e0070e22b35636",

reporter\_name: "Ben",

reporter\_email: "place@holder",

reporter\_contact\_number: "123456789"

},

{

\_id: "5be030f55790eb591c5c9f8e",

reporter\_name: "John",

reporter\_email: "place@holder",

reporter\_contact\_number: "123456789"

},

{

\_id: "5be08fe53a639734e30ccea6",

reporter\_name: "staff",

reporter\_email: "place@holder",

reporter\_contact\_number: "123456789"

},

{

\_id: "5be025e25790eb591c5c9f22",

reporter\_name: "Bob",

reporter\_email: "place@holder",

reporter\_contact\_number: "123456789"

},

{

\_id: "5be06b29f7ef0d1bd9e43d96",

reporter\_name: "admin",

reporter\_email: "place@holder",

reporter\_contact\_number: "123456789"

},

{

\_id: "5be026f25790eb591c5c9f34",

reporter\_name: "Trun",

reporter\_email: "place@holder",

reporter\_contact\_number: "123456789"

},

{

\_id: "5be026b65790eb591c5c9f31",

reporter\_name: "Truno",

reporter\_email: "place@holder",

reporter\_contact\_number: "123456789"

}

]

},

{

'model': 'resources',

'documents': [

{

\_id: "5bd8d989c4f70235f60ef137",

resource\_name: "NEA",

resource\_email: "NEA@gov.sg",

resource\_tel: "123456789",

resource\_sms: "123456789",

resource\_purpose: "Cleaniless"

},

{

\_id: "5bd8d989c4f70675f60ef1e7",

resource\_name: "Police",

resource\_email: "police@gov.sg",

resource\_tel: "123456789",

resource\_sms: "123456789",

resource\_purpose: "Security"

},

{

\_id: "5bd8d9c7c4f70675f60ef13e",

resource\_name: "SAF",

resource\_email: "SAF@gov.sg",

resource\_tel: "123456789",

resource\_sms: "123456789",

resource\_purpose: "Army"

}

]

},

{

'model': 'user',

'documents': [

{

username: "jack",

salt: "9ac926259b2c25ea25be7c8e2c326b8ae61c1032130f4c09e7bbb696a5618060",

hash: "",

role: "admin",

email: "place@holder",

contact\_number: "123456789"

},

{

username: "admin",

salt: "6064a79ee24cb84ae357d026454713cfcb9448d91afb296bd5d2f7f725ada485",

hash: "",

role: "admin",

email: "place@holder",

contact\_number: "123456789"

},

{

username: "staff",

salt: "ff39d194a31495656fced5fa407149022b16f547c48c395d181eeb888107166f",

hash: "",

role: "staff",

email: "place@holder",

contact\_number: "123456789"

},

{

username: "minister",

salt: "17f74278e14fda98339a74966acfd7845a4da0550524e1b31b260368828c6f90",

hash: "",

role: "minister",

email: "place@holder",

contact\_number: "123456789"

}

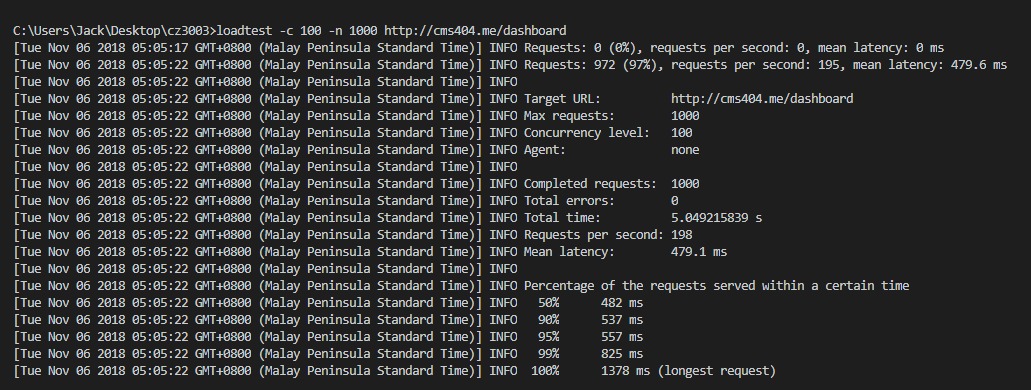
]

}

]

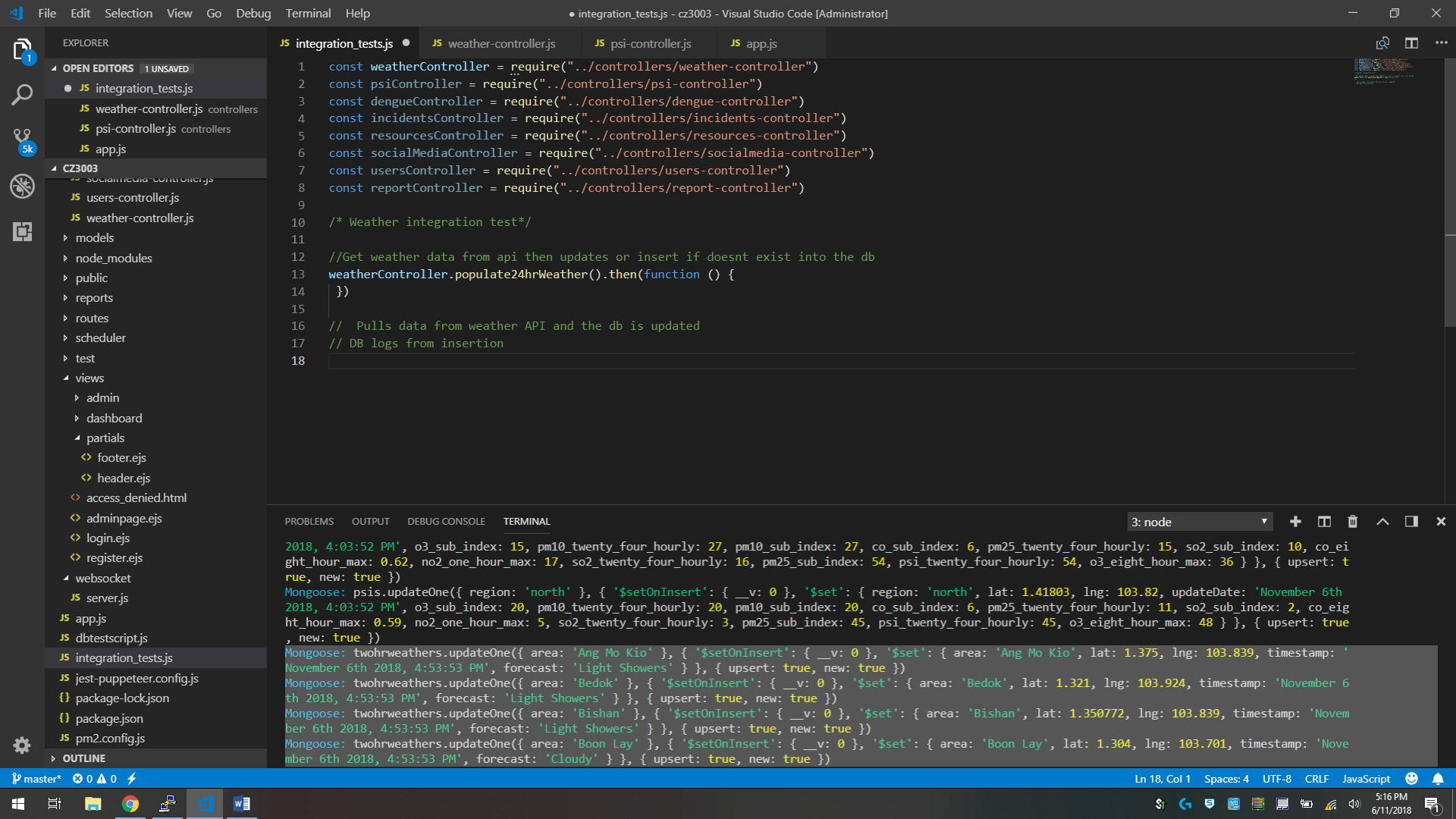
**Load Testing**

**Load testing carried out for cms404.me/dashboard with 100 concurrent users and 1000 total number of requests using loadtest npm package**

**The specifications of 100 concurrent users and 1000 total number of requests were selected to stress test the server, as well as being a much higher margin than normal real-world working conditions.**

**Unit tests for subsystems**

Importing controllers to call functions for subsystems



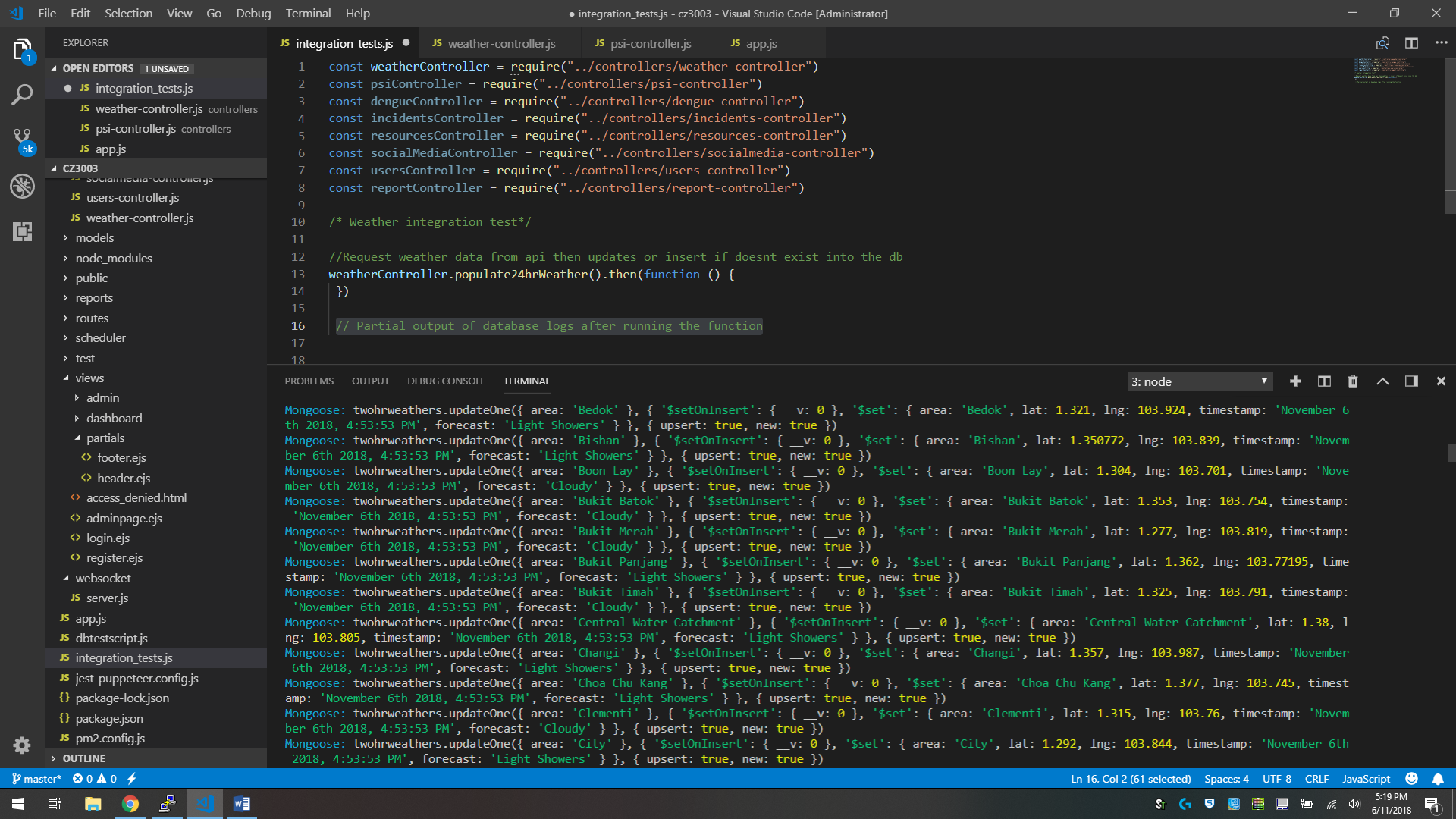
**Weather**

//Request 2hr weather data from API then updates or insert if it doesn’t exist, into the database

weatherController.populate2hrWeather().then(function () {

})

// Partial output of database logs after running the function

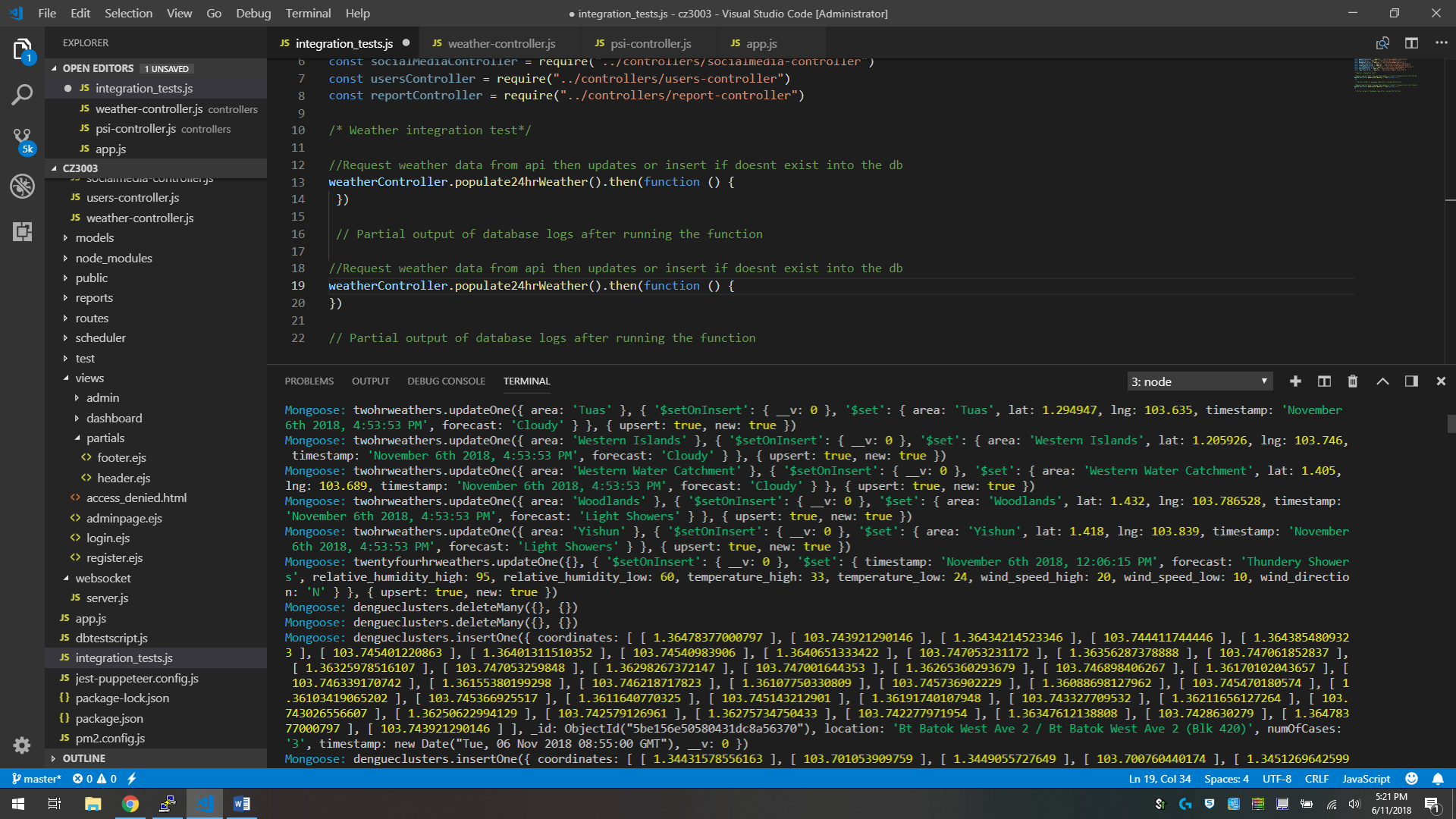


//Request 24hr weather data from API then updates or insert if it doesn’t exist, into the database

weatherController.populate24hrWeather().then(function () {

})

// Output of database logs after running the function



//Query the db and retrieve all data from the 2hr weather database collection

weatherController.get2hrWeather().then(function () {

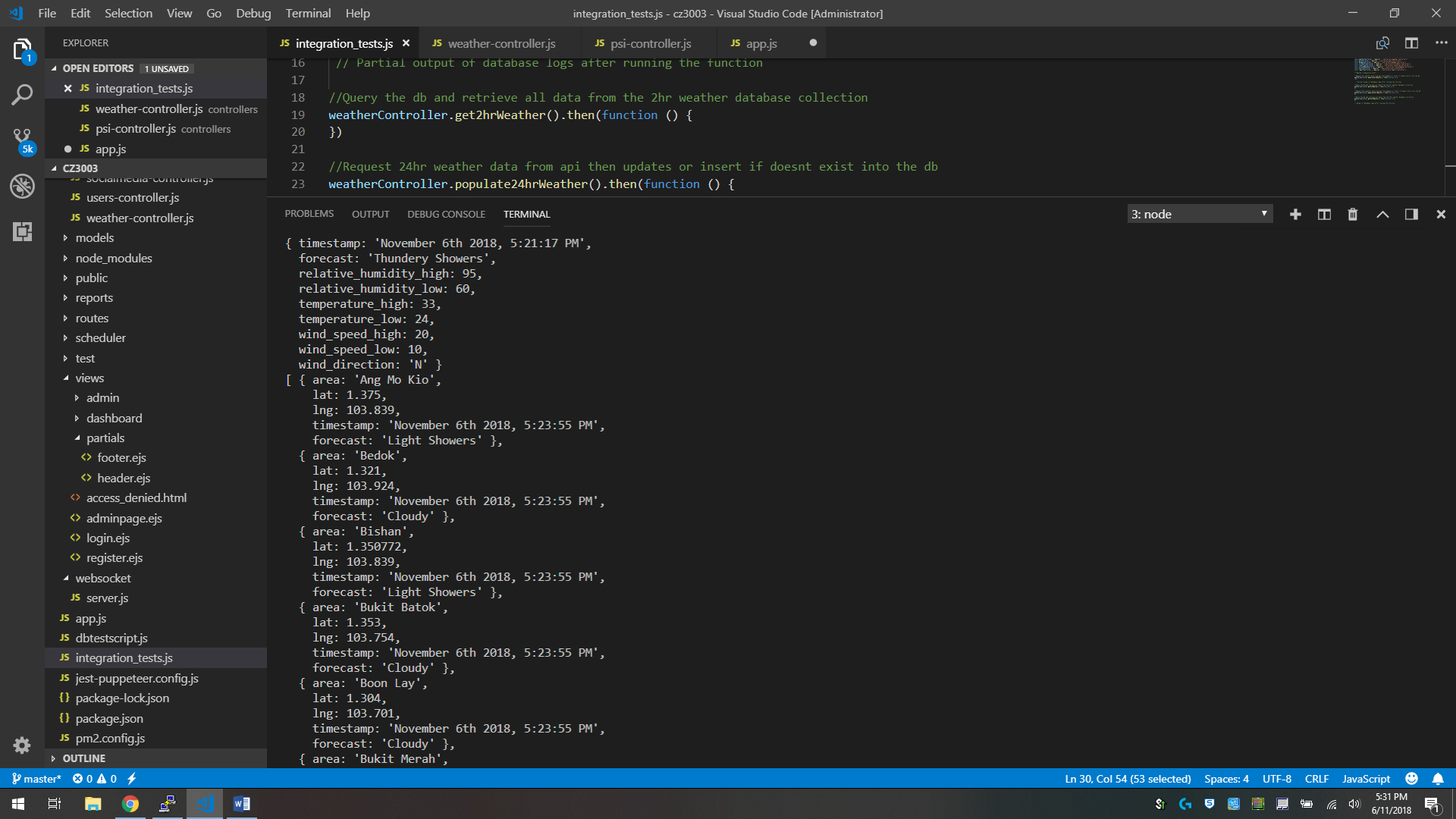
})

// Database log shows the query that is executed

Mongoose: twohrweathers.find({}, { projection: {} })

// Function returns a json containing 2hr data

// Partial part of the json output



//Query the db and retrieve all data from the 24hr weather database collection

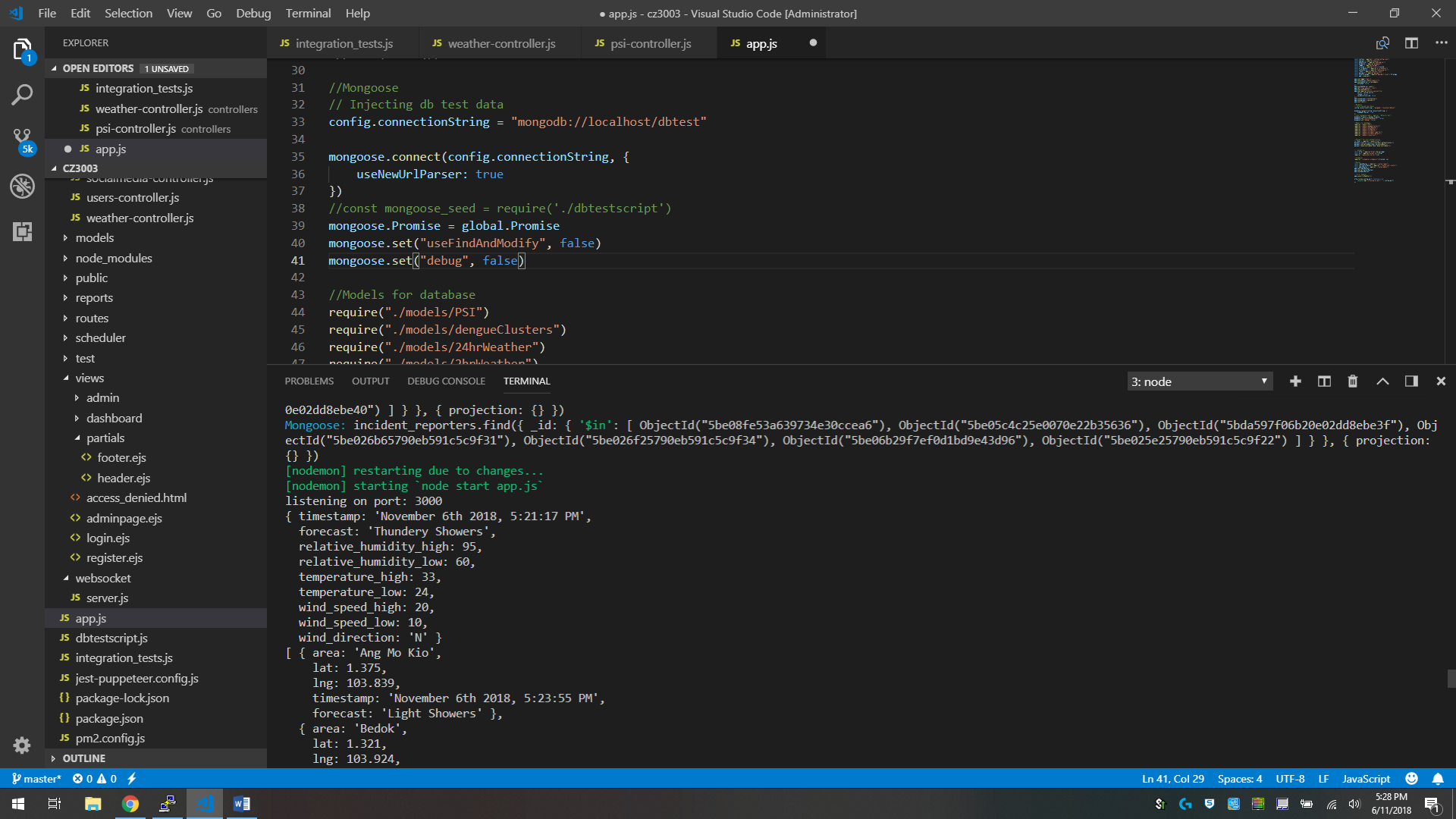
weatherController.get24hrWeather().then(function () {

})

// Database log shows the query that is executed

Mongoose: twentyfourhrweathers.find({}, { projection: {} })

// Function returns a json containing 24hr data



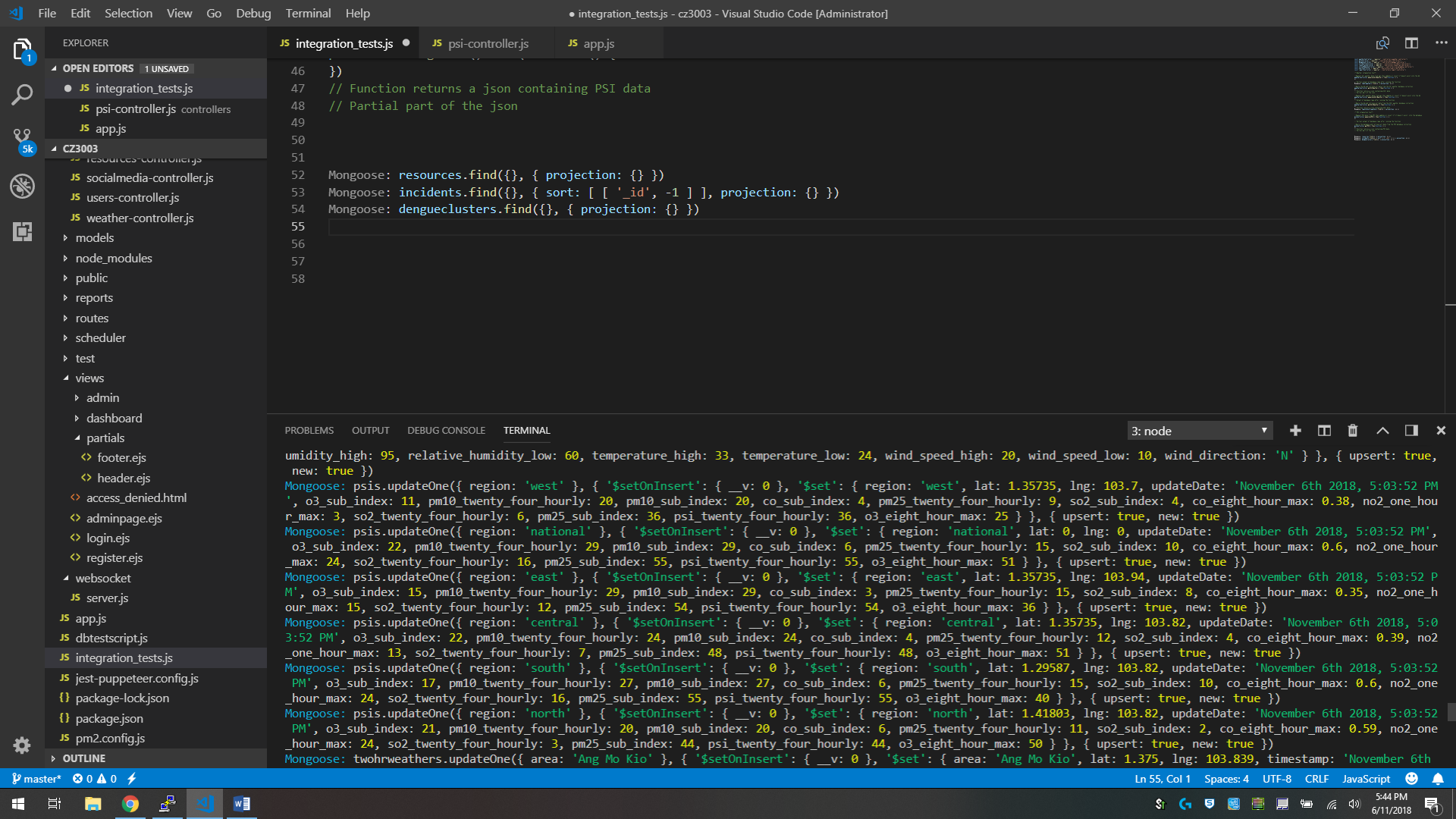
**PSI**

//Request PSI data from API then updates or insert if it doesn’t exist, into the database

psiController.populatePSI().then(function () {

})

// Output of database logs after running the function



//Query the database and retrieve all data from the PSI database collection

psiController.getPSI().then(function () {

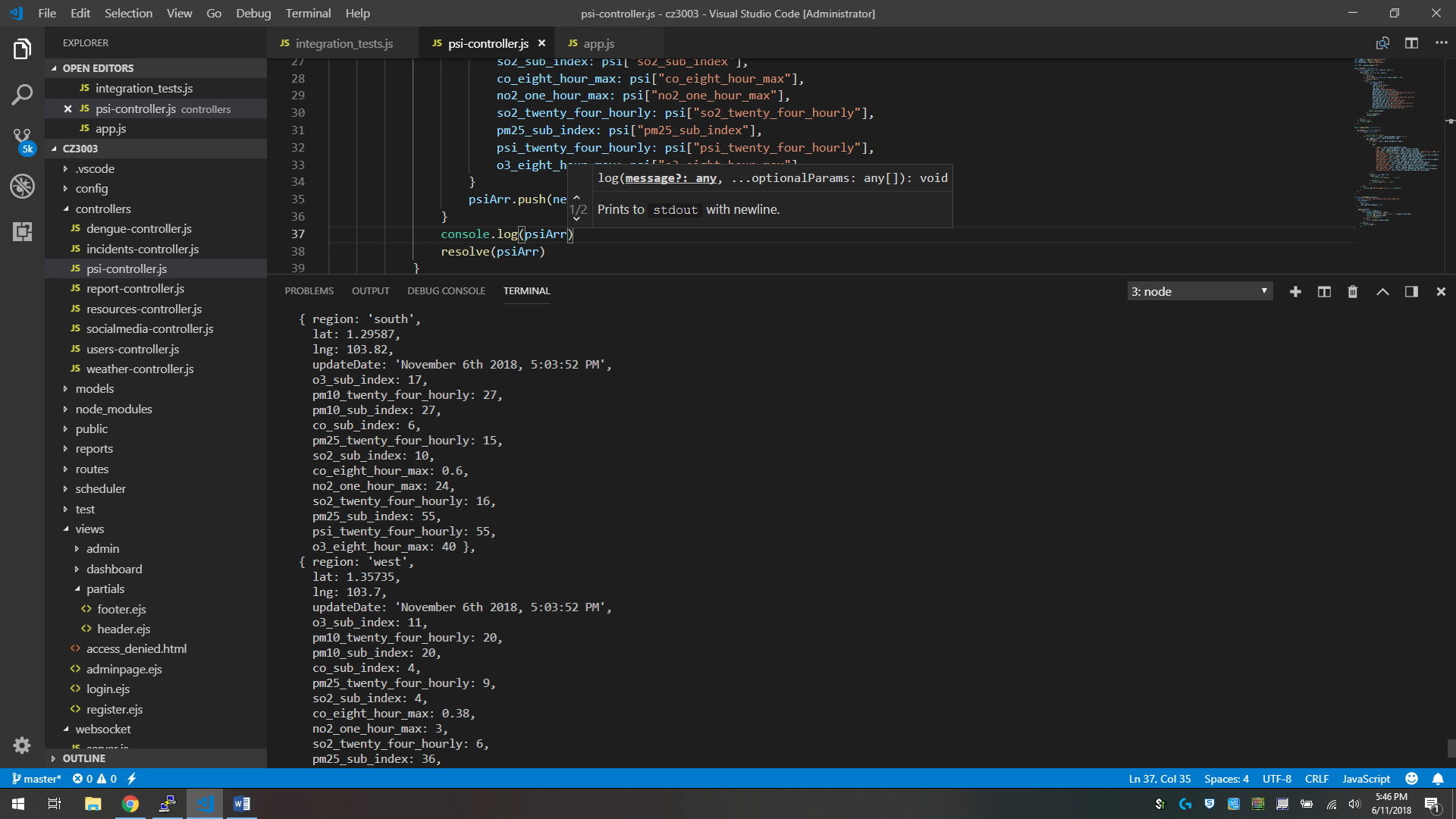
})

// Database log shows the query that is executed

Mongoose: psis.find({}, { projection: {} })

// Function returns a json containing PSI data

// Partial part of the json output



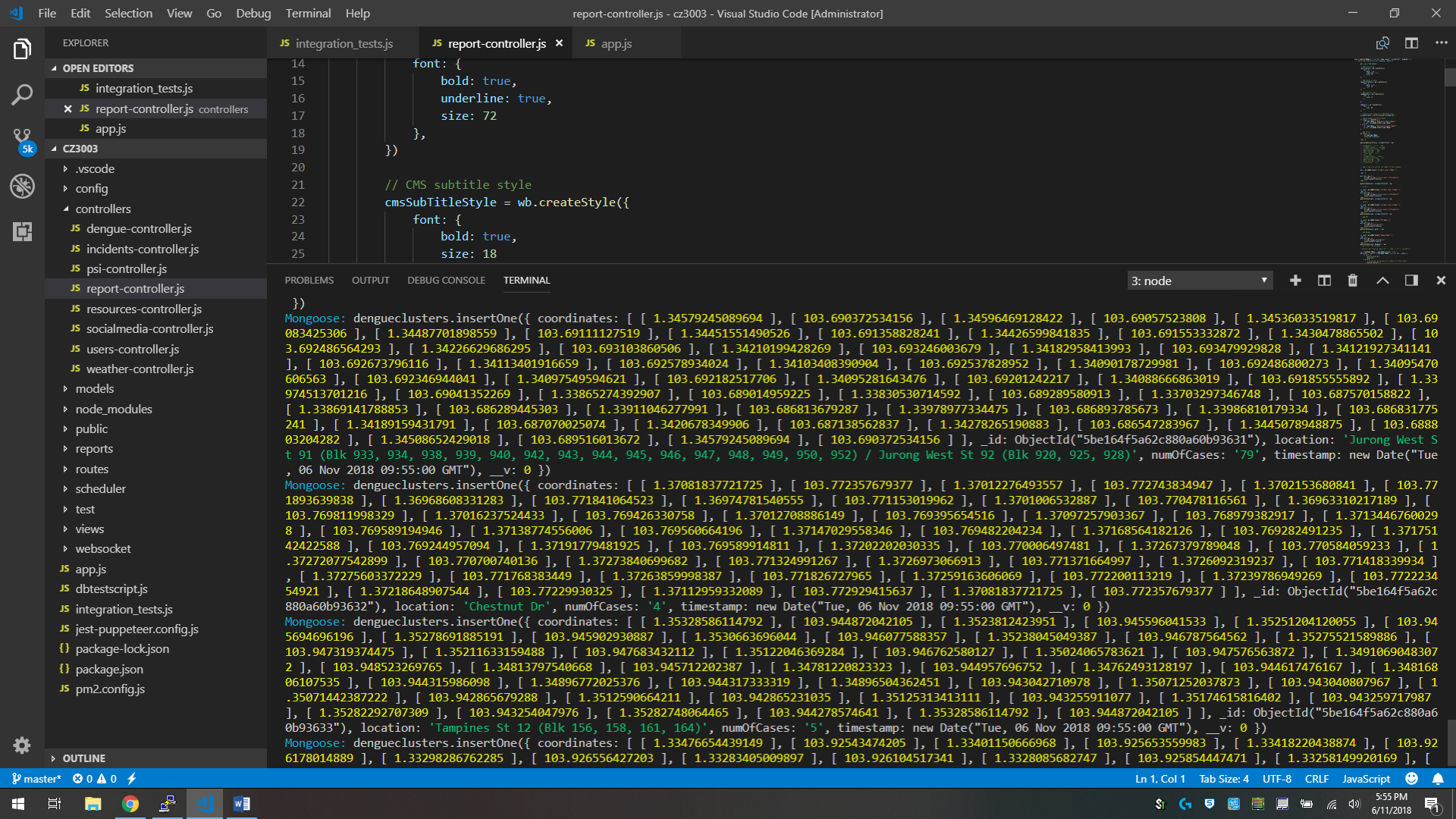
**Dengue Clusters**

//Request dengue clusters data from API then updates or insert if it doesn’t exist, into the database

dengueController.populateDengueClusters().then(function () {

})

// Partial output of database logs after running the function



//Query the database and retrieve all data from the PSI database collection

dengueController.getDengueClusters().then(function () {

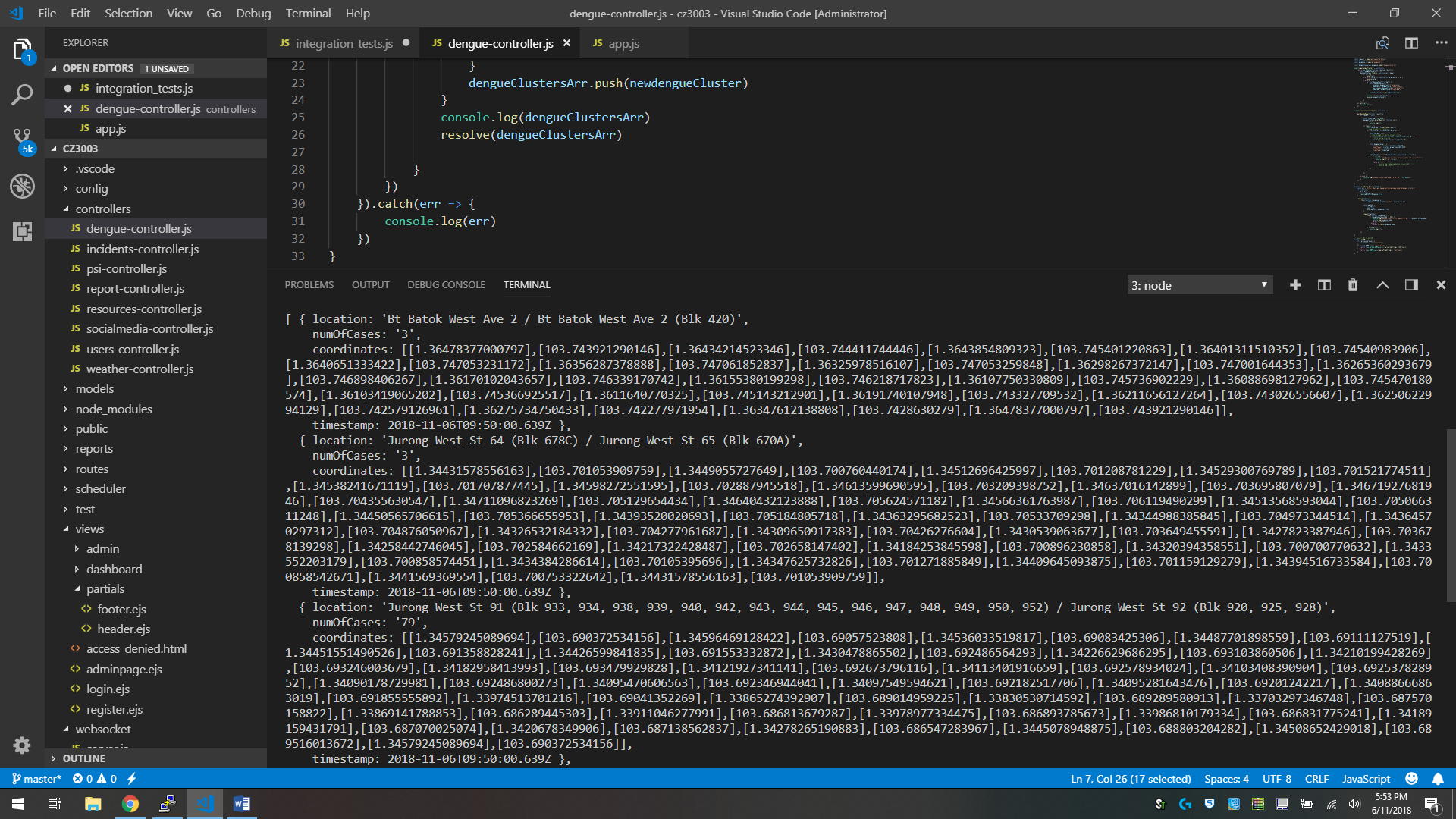
})

// Database log shows the query that is executed

Mongoose: dengueclusters.find({}, { projection: {} })

// Function returns a json containing PSI data

// Partial part of the json output



**Resources**

//Query the database to find all documents in the resources database collection

resourcesController.getResources().then(function () {

})

// Database log shows the query output that is executed

Mongoose: resources.find({}, { projection: {} })

// Function returns a json containing resources data

[ { resource\_id: 5bd8d989c4f70675f60ef1e7,

resource\_name: 'Police',

resource\_email: 'police@gov.sg',

resource\_tel: '123456789',

resource\_sms: '123456789',

resource\_purpose: 'Security' },

{ resource\_id: 5bd8d9c7c4f70675f60ef13e,

resource\_name: 'SAF',

resource\_email: 'SAF@gov.sg',

resource\_tel: '123456789',

resource\_sms: '123456789',

resource\_purpose: 'Army' },

{ resource\_id: 5be17197ac928021440cc6cc,

resource\_name: 'asdsa',

resource\_email: 'dsasd@asdas',

resource\_tel: '432432492',

resource\_sms: '92343242',

resource\_purpose: 'sdsfsf' } ]

//Query the database to find the resource document based on id input in the resources database collection and update its attribute values

resourcesController.updateResource().then(function () {

})

// Database log shows the query that is executed

Mongoose: resources.findOne({ \_id: ObjectId("5be17197ac928021440cc6cc") }, { projection: {} })

Mongoose: resources.updateOne({ \_id: ObjectId("5be17197ac928021440cc6cc") }, {

'$set': {

resource\_email: 'dsasd@asdas',

resource\_name: 'asdsa', resource\_purpose: 'sdsfsf', resource\_sms: '92343242',

resource\_tel: '432432492', updatedAt: new Date("Tue, 06 Nov 2018 10:52:52 GMT")

}

})

//Query the database to add a new document to the resources database collection

resourcesController.addResource().then(function () {

})

// Database log shows the query output that is executed

Mongoose: resources.insertOne({

\_id: ObjectId("5be17197ac928021440cc6cc"),

resource\_name: 'asdasda', resource\_email: 'asd@asdas', resource\_tel: '432432432',

resource\_sms: '2342343242', resource\_purpose: 'sdfsdfsdfsdf',

createdAt: new Date("Tue, 06 Nov 2018 10:48:55 GMT"), updatedAt: new Date("Tue, 06 Nov 2018 10:48:55 GMT"), \_\_v: 0

})

//Query the database to find and delete the resource document based on id input in the resources database collection

resourcesController.deleteResource().then(function () {

})

// Database log shows the query output that is executed

Mongoose: resources.deleteOne({ \_id: ObjectId("5bd8d989c4f70235f60ef137") }, {})

**Users integration test**

//Query the database to find all documents in the user database collection

usersController.getUsers().then(function () {

})

// Database log shows the query output that is executed

Mongoose: users.find({}, { projection: { hash: 0, salt: 0 } })

// Function returns a json containing resources data

[ { id: 5be15598cb32322544eb3c40,

username: 'admin',

role: 'admin',

contact\_number: '123456789',

email: 'place@holder' },

{ id: 5be15598cb32322544eb3c3f,

username: 'jack',

role: 'admin',

contact\_number: '123456789',

email: 'place@holder' },

{ id: 5be15598cb32322544eb3c41,

username: 'staff',

role: 'staff',

contact\_number: '123456789',

email: 'place@holder' },

{ id: 5be15598cb32322544eb3c42,

username: 'ministerr',

role: 'staff',

contact\_number: '1234567891',

email: 'place@holder1' } ]

//Query the database to find the user document based on id input in the user database collection and update its attribute values

usersController.updateUser().then(function () {

})

// Database log shows the query that is executed

Mongoose: users.findOne({ \_id: ObjectId("5be15598cb32322544eb3c42") }, { projection: { hash: 0, salt: 0 } })

Mongoose: users.updateOne({ \_id: ObjectId("5be15598cb32322544eb3c42") },

{

'$set': {

contact\_number: '1234567891', email: 'place@holder1', role: 'staff', updatedAt: new

Date("Tue, 06 Nov 2018 11:33:52 GMT"), username: 'ministerr'

}

})

//Adding of new users is done by express router using the passport.js middleware

// This is the code that creates a new user in the user collection database

var user = new User({

username: req.body.username,

role: req.body.role,

contact\_number: req.body.contact,

email: req.body.email

})

User.register(user, req.body.password, function (err, user) {

if (err) {

console.log(err);

res.render("register", {

page: 'Register'

})

}

// Database log shows the query output that is executed

Mongoose: users.findOne({ '$or': [{ username: 'dsfsdfsdf' }] }, { projection: { hash: 0, salt: 0 } })

Mongoose: users.insertOne({

role: 'staff', \_id: ObjectId("5be17b2c28cfb02090836b45"), username: 'dsfsdfsdf',

contact\_number: '432423423', email: 'asds@sdfdsf.com', salt: '267730e372212f9cc9574266a47104902247ad8e23aa9da7f11018eaebb727a2',

hash: '', createdAt: new Date("Tue, 06 Nov 2018 11:29:49 GMT"),

updatedAt: new Date("Tue,06 Nov 2018 11:29:49 GMT"), \_\_v: 0

})

//Query the database to find and delete the user document based on id input in the user database collection

usersController.deleteUser().then(function () {

})

// Database log shows the query output that is executed

Mongoose: users.deleteOne({ \_id: ObjectId("5be17b2c28cfb02090836b45") }, {})

**Incidents**

//Query the database to find all documents in the incidents database collection

incidentsController.getIncidents().then(function () {

})

// Database log shows the query output that is executed

Mongoose: incidents.find({}, { sort: [['\_id', -1]], projection: {} })

// Function returns a json containing incidents data

// Partial output of the json data

[{

reporter\_name: 'staff',

reporter\_email: 'place@holder',

reporter\_contact\_number: '123456789',

incident\_id: 5be08fe53a639734e30ccea8,

incident\_name: 'Fire',

incident\_location: 'Choa Chu Kang Avenue 4, Lot One Shoppers\' Mall, Singapore',

incident\_lat: 1.3851131,

incident\_lng: 103.744937,

incident\_description: 'Fire in mall',

incident\_crisis\_level: 0,

incident\_status: 'ONGOING',

incident\_activities: 'None',

incident\_date\_created: 2018 - 11 - 01T01: 40: 16.007Z,

incident\_date\_resolved: 2018 - 11 - 01T01: 40: 16.007Z

},

{

reporter\_name: 'Ben',

reporter\_email: 'place@holder',

reporter\_contact\_number: '123456789',

incident\_id: 5be05c4c25e0070e22b35638,

incident\_name: 'Dengue',

incident\_location: 'Asia Square Tower 2, Singapore',

incident\_lat: 1.2781459,

incident\_lng: 103.8511983,

incident\_description: 'A',

incident\_crisis\_level: 2,

incident\_status: 'ONGOING',

incident\_activities: 'None',

incident\_date\_created: 2018 - 11 - 01T01: 40: 16.007Z,

incident\_date\_resolved: 2018 - 11 - 01T01: 40: 16.007Z

}]

//Query the database to find the incident document based on id input in the incidents database collection

// and the incident\_details id in the incident document is used to locate a correspoding incident\_details document

// and update the crisis level value in it

incidentsController.updateIncidentCrisisLevel().then(function () {

})

//Database log shows a sample query output that is executed

Mongoose: incidents.findOne({ \_id: ObjectId("5be18e13f3f2673108a90d6d") }, { projection: {} })

Mongoose: incident\_details.findOne({ \_id: ObjectId("5be18e13f3f2673108a90d6c") }, { projection: {} })

Mongoose: incident\_details.updateOne({ \_id: ObjectId("5be18e13f3f2673108a90d6c") }, { '$set': { crisis\_level: 1 } })

//Query the database to add a new document to the incidents database collection

incidentsController.uploadIncident().then(function () {

})

// Database log shows a sample query output that is executed

Mongoose: incident\_reporters.insertOne({

\_id: ObjectId("5be18e13f3f2673108a90d6b"),

reporter\_name: 'jack', reporter\_email: 'place@holder', reporter\_contact\_number: '123456789',

createdAt: new Date("Tue, 06 Nov 2018 12:50:27 GMT"), updatedAt: new Date("Tue, 06 Nov 2018 12:50:27 GMT"), \_\_v: 0

})

Mongoose: incident\_details.insertOne({

incident\_status: 'ONGOING', crisis\_level: 0,

\_id: ObjectId("5be18e13f3f2673108a90d6c"), incident\_activities: 'None', date\_created: new Date("Tue, 06 Nov 2018 12:50:27 GMT"),

date\_resolved: new Date("Tue, 06 Nov 2018 12:50:27 GMT"), \_\_v: 0

})

Mongoose: incidents.insertOne({

\_id: ObjectId("5be18e13f3f2673108a90d6d"),

incident\_location: 'Choa Chu Kang, Singapore', incident\_lat: 1.3839803, incident\_lng: 103.7469611,

incident\_name: 'Dengue', incident\_description: 'adasdas', incident\_details: ObjectId("5be18e13f3f2673108a90d6c"),

incident\_reporter: ObjectId("5be18e13f3f2673108a90d6b"), \_\_v: 0

})

//Query the database to find the incident document based on id input in the incidents database collection

// and the incident\_details id in the incident document is used to locate a correspoding incident\_details document

// and update the activities field

incidentsController.updateIncidentActionsTaken().then(function () {

})

//Database log shows a sample query output that is executed

Mongoose: incidents.findOne({ \_id: ObjectId("5be08fe53a639734e30ccea8") }, { projection: {} })

Mongoose: incident\_details.findOne({ \_id: ObjectId("5be08fe53a639734e30ccea7") }, { projection: {} })

Mongoose: incident\_details.updateOne({ \_id: ObjectId("5be08fe53a639734e30ccea7") }, { '$set': { incident\_activities: 'Police' } })

Posted sms\_info! Incident: Fire, Date: 01 - 11 - 2018, Description: Fire in mall, Location: Choa Chu Kang Avenue 4, Lot One Shoppers' Mall, Singapore

//Query the database to find the incident document based on id input in the incidents database collection

// and the incident\_details id in the incident document is used to locate a correspoding incident\_details document

// and update the incident status from "ONGOING" to "RESOLVED"

incidentsController.updateIncidentStatus().then(function () {

})

//Database log shows a sample query output that is executed

Mongoose: incidents.findOne({ \_id: ObjectId("5be18e13f3f2673108a90d6d") }, { projection: {} })

Mongoose: incident\_details.findOne({ \_id: ObjectId("5be18e13f3f2673108a90d6c") }, { projection: {} })

Mongoose: incident\_details.updateOne({ \_id: ObjectId("5be18e13f3f2673108a90d6c") },

{ '$set': { date\_resolved: new Date("Tue, 06 Nov 2018 13:12:57 GMT"), incident\_status: 'RESOLVED' } })

//Query the database to find all documents in the incidents database collection

// then depending on the type input of either "ONGOING" or "RESOLVED"

// the incidents data will be filtered to only contain either one of them and return the filtered data

incidentsController.getReportIncidents(type).then(function () {

})

// Database log shows the query output that is executed

Mongoose: incidents.find({}, { sort: [['\_id', -1]], projection: {} })

// Function returns a json containing incidents data

// Partial output of the json data

[{

reporter\_name: 'jack',

reporter\_email: 'place@holder',

reporter\_contact\_number: '123456789',

incident\_id: 5be18e13f3f2673108a90d6d,

incident\_name: 'Dengue',

incident\_location: 'Choa Chu Kang, Singapore',

incident\_lat: 1.3839803,

incident\_lng: 103.7469611,

incident\_description: 'adasdas',

incident\_crisis\_level: 1,

incident\_status: 'RESOLVED',

incident\_activities: 'None',

incident\_date\_created: 2018 - 11 - 06T12: 50: 27.325Z,

incident\_date\_resolved: 2018 - 11 - 06T13: 12: 57.439Z

},

{

reporter\_name: 'Truno',

reporter\_email: 'place@holder',

reporter\_contact\_number: '123456789',

incident\_id: 5be026b65790eb591c5c9f33,

incident\_name: 'Fire',

incident\_location: 'Singapore',

incident\_lat: 1.352083,

incident\_lng: 103.819836,

incident\_description: 'F',

incident\_crisis\_level: 3,

incident\_status: 'RESOLVED',

incident\_activities: 'Police,NEA',

incident\_date\_created: 2018 - 11 - 01T01: 40: 16.007Z,

incident\_date\_resolved: 2018 - 11 - 01T01: 40: 16.007Z

}]

**Report**

// Data returned from other controllers will be passed into the generateReport function

// An excel file will be generated and the data that was passed in previously

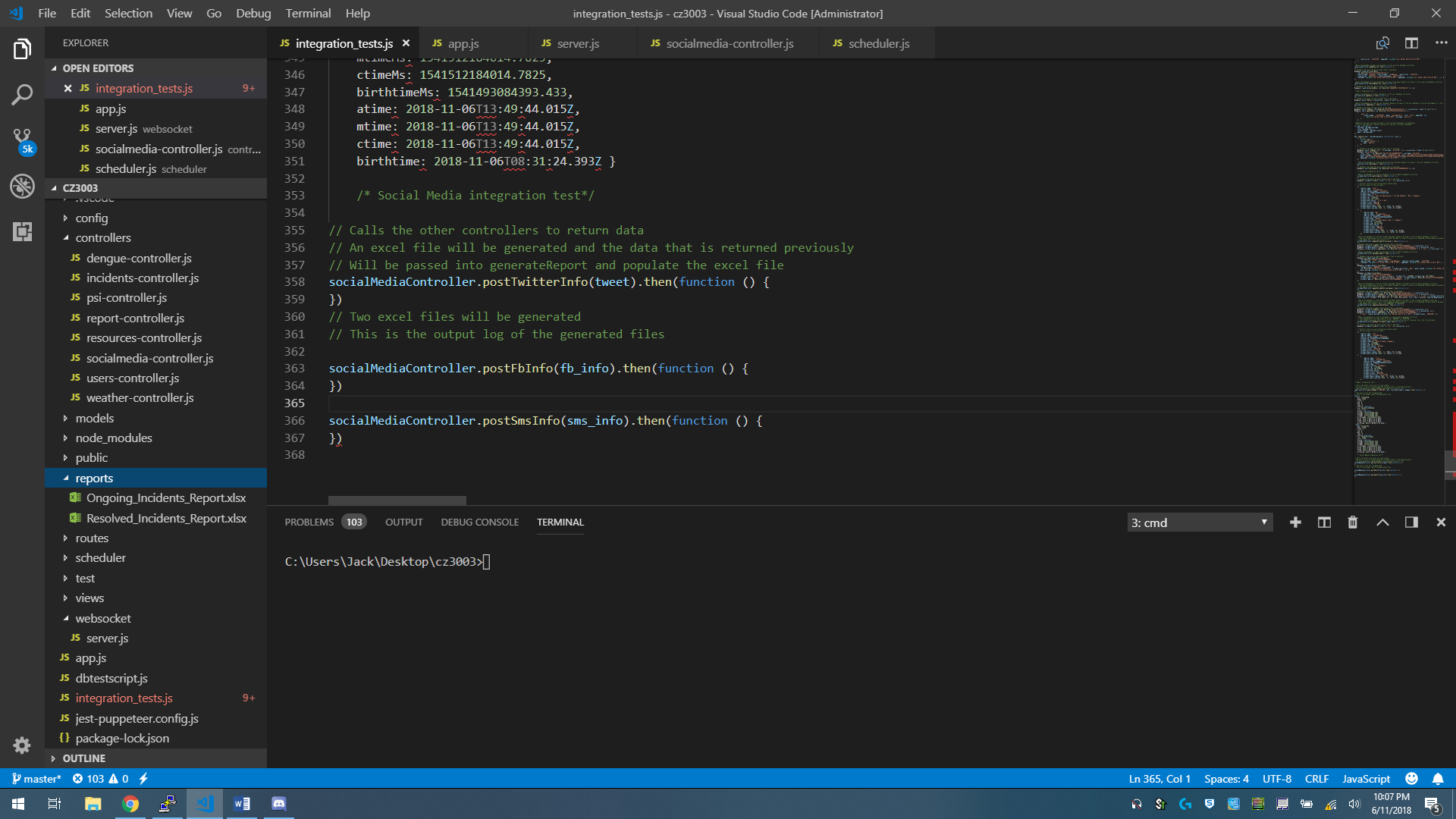
// will be used to populate the excel content body

reportController.generateReport("ONGOING", psi, resolvedincidents, dengue).then(function () {

})

// Two excel files will be generated into the reports folder

// One for ongoing incidents, one for resolved



// This is the output log of the generated files

Stats {

dev: 2761646796,

mode: 33206,

nlink: 1,

uid: 0,

gid: 0,

rdev: 0,

blksize: undefined,

ino: 10414574138325040,

size: 11454,

blocks: undefined,

atimeMs: 1541512184006.7861,

mtimeMs: 1541512184006.7861,

ctimeMs: 1541512184006.7861,

birthtimeMs: 1541493084352.8647,

atime: 2018-11-06T13:49:44.007Z,

mtime: 2018-11-06T13:49:44.007Z,

ctime: 2018-11-06T13:49:44.007Z,

birthtime: 2018-11-06T08:31:24.353Z }

**Social Media**

// Server authenticates the twitter account and sends a post request with a message body which is to be posted

socialMediaController.postTwitterInfo(tweet).then(function () {

})

// Output log after successfully posting the twitter post will contain the stats of the tweet

created\_at: 'Tue Nov 06 14:54:23 +0000 2018',

id: 1059821300930220000,

id\_str: '1059821300930220032',

text: 'Incident: Fire, Date: 01-11-2018, Description: Fire in mall, Location: Choa Chu Kang Avenue 4, Lot One Shoppers\' Mall, Singapore',

truncated: false,

entities: { hashtags: [], symbols: [], user\_mentions: [], urls: [] },

source: '<a href="https://CMS404.me" rel="nofollow">CMS404</a>',

in\_reply\_to\_status\_id: null,

in\_reply\_to\_status\_id\_str: null,

in\_reply\_to\_user\_id: null,

in\_reply\_to\_user\_id\_str: null,

in\_reply\_to\_screen\_name: null,

user:

{ id: 1054947428422557700,

id\_str: '1054947428422557696',

name: 'cms404',

screen\_name: 'behroyston',

location: '',

description: '',

url: null,

entities: { description: [Object] },

protected: false,

followers\_count: 0,

friends\_count: 0,

listed\_count: 0,

created\_at: 'Wed Oct 24 04:07:21 +0000 2018',

favourites\_count: 0,

utc\_offset: null,

time\_zone: null,

geo\_enabled: false,

verified: false,

statuses\_count: 16,

lang: 'en',

contributors\_enabled: false,

is\_translator: false,

is\_translation\_enabled: false,

profile\_background\_color: 'F5F8FA',

profile\_background\_image\_url: null,

profile\_background\_image\_url\_https: null,

profile\_background\_tile: false,

profile\_image\_url: 'http://abs.twimg.com/sticky/default\_profile\_images/default\_profile\_normal.png',

profile\_image\_url\_https: 'https://abs.twimg.com/sticky/default\_profile\_images/default\_profile\_normal.png',

profile\_link\_color: '1DA1F2',

profile\_sidebar\_border\_color: 'C0DEED',

profile\_sidebar\_fill\_color: 'DDEEF6',

profile\_text\_color: '333333',

profile\_use\_background\_image: true,

has\_extended\_profile: false,

default\_profile: true,

default\_profile\_image: true,

following: false,

follow\_request\_sent: false,

notifications: false,

translator\_type: 'none' },

geo: null,

coordinates: null,

place: null,

contributors: null,

is\_quote\_status: false,

retweet\_count: 0,

favorite\_count: 0,

favorited: false,

retweeted: false,

lang: 'en' }

// Server authenticates the Facebook account and sends a post request with a message body which is to be posted

socialMediaController.postFbInfo(fb\_info).then(function () {

})

// Outlog log after successfully posting the Facebook post will contain the facebook post's id

id: '1099218763620046\_1104955429713046'

// Server authenticates the Twilio account and sends a request with a message body which is to be sent in the SMS

socialMediaController.postSmsInfo(sms\_info).then(function () {

})

// Output log after successfully sending the SMS will contain the message's sid

SM11360ba97db3486589c5e9b370ef319f

// Server creates a Nodemailer transporter using SMTP transport and a Gmail account is authenticated with credentials

// The email header, body and attachment options are created for Nodemailer

// and the email will be sent out using the transporter

socialMediaController.postEmailInfo("Manually", email)

// Output log after successfully sending the composed email

Email sent: 250 2.0.0 OK 1541515782 v20-v6sm103105174pfj.107 - gsmtp

**Black box integration test for generate report**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Test Case Objective | Test Case Description | Expected Result | Actual Results |  |
| 1 | Check the link between the “Generate Report” button in interface to the generateReport socket.io event | Click on the “Generate Report” button | generateReport socket.io event is called | As expected |  |
| 2 | Check the flow from getting data from the database and generating a report with the data | generateReport socket.io event calls psiController.getPSI(), dengueController.getDengueClusters() and incidentsController.getReportIncidents() and passes the data returned from the 3 functions to reportController.generateReport() function | reportController.generateReport() function generates an excel file and populates it with the data that was passed to it and saves the excel file in a local directory | As expected |  |
| 3 | Check the flow from generating a report to sending it through email | reportController.generateReport() function completes generating a report and calls the socialMediaController.postEmailInfo() function | socialMediaController.postEmailInfo() function creates an email and attaches the excel files in the local directory and sends the email | As expected |  |