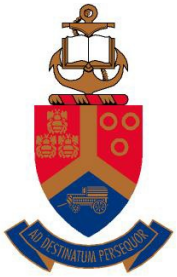


UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

NavUP
Testing Broadsword GIS Report
By Longsword GIS

Andriod-GIS

Mfana Masimula
12077713

Bongani Tshela
14134790

Joshua Moodley
14152152

Boikanyo Modiko
15227678

All Testing can be found here:
GitHub https://github.com/SirJosh/Android-GIS/tree/master/Testing-Phase4

Contents

1 Introduction

1.1 Scope

Services to gather, maintain, persist and provide information related to the world serviced by the system. It is about the creation and maintenance of a GIS Map of the campus by using WiFi signal strengths and other available sources of GIS information. This module provide services to search for locations such as landmarks, buildings as well as venues such as offices, lecture halls, labs, etc.

2 Functional Requirements

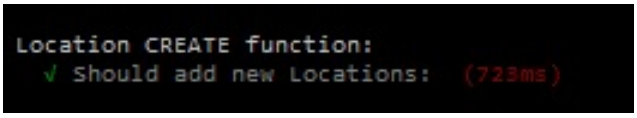
2.1 Create Functionality

2.1.1 Add a location

Pass/Fail	Mark (10)	Reason
✓	8	The function works well when given the correct parameters for single object insert and fails for batch inserts (tested in the code above).

```
//var assert = require('assert');
//var locations = require('../../controllers/locations/locations');
//var http_mocks = require('node-mocks-http');
var should = require('should');
var needle      = require('needle');

describe('Location CREATE function: ', function() {
  it('Should add new Locations: ', function(done)
  {
    var locationData = {
        "location_type" : "mochatest",
        "room" : "testRoom",
        "building" : "testBuilding",
        "lat" : -23.762481,
        "lng" : 28.293426,
        "level" : 4,
        "ground" : 2
    };

    /*var batLocationData = {

        "location_type" : "mochaTest",
        "room" : "testRoom",
        "building" : "testBuilding",
        "lat" : -23.762481,
        "lng" : 28.293426,
        "level" : 4,
        "ground" : 2

        "location_type" : "mochatest2",
        "room" : "testroom2",
        "building" : "testBuilding2",
        "lat" : -23.762491,
        "lng" : 28.293436,
        "level" : 5,
        "ground" : 3

        "location_type" : "mochatest3",
        "room" : "testRoom3",
        "building" : "testBuilding3",
        "lat" : -23.762471,
        "lng" : 28.2934728,
        "level" : 6,
        "ground" : 4
    }*/

    needle.post('localhost:3000/location', locationData, function(err, res) //for batch insert use the batLocationData varieble for batch insert.
    {
        if (err) { done(); return console.error(err.message); }

        res.statusCode.should.equal(200)
        res.body.should.have.property('data');

        // to display body content
        //console.log(res.body.data);

        done();
    });

  });
});
```

Figure 1: Test code for add location function



```
Location CREATE function:
  √ Should add new Locations:  (723ms)
```

Figure 2: Result after test is run: status code 200

2.2 Remove Functionality

2.2.1 Remove location

Pass/Fail	Mark (10)	Reason
✓	10	The remove function works well when given the correct parameters It will get the exact location that we want to remove, using both the building details and the coordinates

```
//var assert = require('assert');
//var locations = require('../../controllers/locations/locations');
//var http_mocks = require('node-mocks-http');
var should = require('should');
var needle      = require('needle');

describe('Location UPDATE function: ', function() {
  it('Should update specified Location: ', function(done)
  {
    var locationData = {
        "room" : "testRoom2",
        "building" : "testBuilding2",
        "lat" : -23.762492,
        "lng" : 28.293438,
        "level" : 2,
        "ground" : 2
    };

    needle.patch('localhost:3000/location', locationData, function(err, res)
    {
        if (err) { done(); return console.error(err.message); }

        res.statusCode.should.equal(200)
        //res.body.should.have.property('data');

        // to display body content
        //console.log(res.body);

        done();
    });

  });
});
```

Figure 3: Test code for remove location function

```
Location CREATE function:
  √ Should add another new Location:  (70ms)
```

Figure 4: Result after test is run: status code 200

2.3 Update Functionality

2.3.1 Update location

Pass/Fail	Mark (10)	Reason
✓	7	The update function works correctly given the correct parameters. The concern is that it might not get the right or the only location given only the room and the building (as opposed to room, building and the coordinates).

```
//var assert = require('assert');
//var locations = require('../../controllers/locations/locations');
//var http_mocks = require('node-mocks-http');
var should = require('should');
var needle      = require('needle');

describe('Location DELETE function: ', function() {
  it('Should remove specified Locations: ', function(done)
  {
    var locationData = {
        "room" : "testRoom",
        "building" : "testBuilding"
    };

    needle.delete('localhost:3000/location', locationData, function(err, res)
    {
        if (err) { done(); return console.error(err.message); }

        res.statusCode.should.equal(200)
        //res.body.should.have.property('data');

        // to display body content
        //console.log(res.body);

        done();
    });

  });
});
```

Figure 5: Test code for update function

```
Location READ function:
  √ Should retreive all Locations:  (144ms)
```

Figure 6: Result after test is run: status code 200

2.4 Request Functionality

2.4.1 Get all locations

Pass/Fail	Mark (10)	Reason
✓	10	The retrieve all locations request returns all the location's stored within the database on campus. The request works when new locations are added. Thus showing that the database performs the request correctly. This also holds true when you call this function multiple times sequentially. Overall the function performs what it is defined to do, without causing strain on the database.

```
//var assert = require('assert');
//var locations = require('../../controllers/locations/locations');
//var http_mocks = require('node-mocks-http');
var should = require('should');
var needle      = require('needle');

describe('Location READ function: ', function() {
  it('Should retreive all Locations: ', function(done)
  {
    needle.get('localhost:3000/locations', function(err, res)
    {
        if (err) { return console.error(err.message); }

        res.statusCode.should.equal(200)
        res.body.should.have.property('data');

        // to display body content
        //for(var i=0; i < res.body.data.length; i++) console.log(res.body.data[i]);

        done();
    });

  });
});
```

Figure 7: Test code for display all locations


```
{"data":[{"type":"locations","id":"5909c7a4d063dc70755e708a","attributes":
{"location_type":"Entrance","room":"N/A","building":"IT","lat":-25.75864,"lng":28.233146,"level":2,"ground":2}},
{"type":"locations","id":"5909c7a4d063dc70755e708b","attributes":
{"location_type":"Entrance","room":"N/A","building":"IT","lat":-25.755972,"lng":28.233055,"level":2,"ground":2}},
{"type":"locations","id":"5909c7a4d063dc70755e708c","attributes":
{"location_type":"Catwalk","room":"N/A","building":"IT","lat":-25.755705,"lng":28.23345,"level":4,"ground":4}},
{"type":"locations","id":"5909c7a4d063dc70755e708d","attributes":
{"location_type":"Catwalk","room":"N/A","building":"EMB","lat":-25.755607,"lng":28.233481,"level":4,"ground":4}},
{"type":"locations","id":"5909c7a4d063dc70755e708e","attributes":
{"location_type":"Entrance","room":"N/A","building":"EMB","lat":-25.755393,"lng":28.233289,"level":2,"ground":2}},
{"type":"locations","id":"5909c7a4d063dc70755e708f","attributes":
{"location_type":"Entrance","room":"N/A","building":"Centenary","lat":-25.753976,"lng":28.232996,"level":1,"ground":1}},
{"type":"locations","id":"5909c7a4d063dc70755e7090","attributes":
{"location_type":"Entrance","room":"N/A","building":"Thuto","lat":-25.75288,"lng":28.231594,"level":1,"ground":1}},
{"type":"locations","id":"5909c7a4d063dc70755e7091","attributes":{"location_type":"Entrance","room":"N/A","building":"Student
Services","lat":-25.755257,"lng":28.231293,"level":1,"ground":1}},{"type":"locations","id":"5909c7a4d063dc70755e7092","attributes":
{"location_type":"Entrance","room":"N/A","building":"Humanities","lat":-25.75533,"lng":28.230773,"level":3,"ground":3}},
{"type":"locations","id":"5909c7a4d063dc70755e7093","attributes":{"location_type":"Entrance","room":"N/A","building":"Merensky
Library","lat":-25.755304,"lng":28.230537,"level":3,"ground":3}},{"type":"locations","id":"5909c7a4d063dc70755e7094","attributes":
{"location_type":"Entrance","room":"N/A","building":"Theology","lat":-25.755046,"lng":28.229466,"level":1,"ground":1}},
{"type":"locations","id":"5909c7a4d063dc70755e7095","attributes":
{"location_type":"Entrance","room":"N/A","building":"Chemistry","lat":-25.753052,"lng":28.230763,"level":1,"ground":1}},
{"type":"locations","id":"5909c7a4d063dc70755e7096","attributes":
{"location_type":"Entrance","room":"N/A","building":"EMB","lat":-25.755062,"lng":28.232988,"level":2,"ground":2}},
{"type":"locations","id":"5909c7a4d063dc70755e7097","attributes":
{"location_type":"Entrance","room":"N/A","building":"EMB","lat":-25.755207,"lng":28.232936,"level":2,"ground":2}},
{"type":"locations","id":"5909c7a4d063dc70755e7098","attributes":
{"location_type":"Entrance","room":"N/A","building":"Piazza","lat":-25.754648,"lng":28.232036,"level":0,"ground":0}},
{"type":"locations","id":"5909c7a4d063dc70755e7099","attributes":
{"location_type":"Entrance","room":"N/A","building":"Piazza","lat":-25.754266,"lng":28.231505,"level":0,"ground":0}},
{"type":"locations","id":"5909c7a4d063dc70755e709a","attributes":
{"location_type":"Entrance","room":"N/A","building":"Piazza","lat":-25.753976,"lng":28.231567,"level":0,"ground":0}},
{"type":"locations","id":"5909c7a4d063dc70755e709b","attributes":
{"location_type":"Point","room":"N/A","building":"N/A","lat":-25.755355,"lng":28.232237,"level":0,"ground":0}},
{"type":"locations","id":"5909c7a4d063dc70755e709c","attributes":
{"location_type":"Point","room":"N/A","building":"N/A","lat":-25.755274,"lng":28.231658,"level":0,"ground":0}},
{"type":"locations","id":"5909c7a4d063dc70755e709d","attributes":
{"location_type":"Point","room":"N/A","building":"N/A","lat":-25.755211,"lng":28.231305,"level":0,"ground":0}},
{"type":"locations","id":"5909c7a4d063dc70755e709e","attributes":
{"location_type":"Point","room":"N/A","building":"N/A","lat":-25.755137,"lng":28.230924,"level":0,"ground":0}},
{"type":"locations","id":"5909c7a4d063dc70755e709f","attributes":
{"location_type":"Point","room":"N/A","building":"N/A","lat":-25.755125,"lng":28.23081,"level":0,"ground":0}},
{"type":"locations","id":"5909c7a4d063dc70755e70a0","attributes":
{"location_type":"Point","room":"N/A","building":"N/A","lat":-25.755072,"lng":28.230897,"level":0,"ground":0}},
```


Figure 8: JSON string returned

```
Location READ function:
  ✓ Should retreive all Locations:  (54ms)
```

Figure 9: Result after test is run: status code 200

2.4.2 Get all buildings

Pass/Fail	Mark (10)	Reason
✓	9	After testing the getBuildingNames request it retrieves all the buildings that are on campus. The request also works when new buildings are added. Thus showing that the database handles the request correctly. This also holds true when you call this function multiple times sequentially. Overall the function performs what it needs to do, without causing the database to crash or hang.

```
//var assert = require('assert');
//var locations = require('../../controllers/locations/locations');
//var http_mocks = require('node-mocks-http');
var should = require('should');
var needle      = require('needle');

describe('Retrieve building names function: ', function() {
  it('Should retreive a list of all available buildings: ', function(done)
  {
    needle.get('localhost:3000/locations/getBuildingNames', function(err, res)
    {
        if (err) { return console.error(err.message); }

        res.statusCode.should.equal(200)
        res.body.should.have.property('data');

        // to display body content
        //for(var i=0; i < res.body.data.length; i++) console.log(res.body.data[i]);

        done();
    });

  });
});
```

Figure 10: Test code for display all function

```
Retrieve building names function:
  ✓ Should retreive a list of all available buildings:
```

Figure 11: Result after test is run: status code 200


```
{"data":["IT","EMB","Centenary","Thuto","Student Services","Humanities","Merensky Library","Theology","Chemistry","Piazza","N/A","testBuilding2"]}
```


Figure 12: JSON string returned

2.4.3 Get location by building name

Pass/Fail	Mark (10)	Reason
✓	10	The function successfully returns a location when given a valid building name

```
//var assert = require('assert');
//var locations = require('../../controllers/locations/locations');
//var http_mocks = require('node-mocks-http');
var should = require('should');
var needle      = require('needle');

describe('Retrieve Locations by building name function: ', function() {
  it('Should retreive all Locations in building=IT: ', function(done)
  {
    var building = "IT";

    needle.get('localhost:3000/locations/getByBuildingName/'+building, function(err, res)
    {
        if (err) { return console.error(err.message); }

        res.statusCode.should.equal(200)
        res.body.should.have.property('data');

        // to display body content
        //for(var i=0; i < res.body.data.length; i++) console.log(res.body.data[i]);

        done();
    });

  });
});
```

Figure 13: Test code for given IT building room test

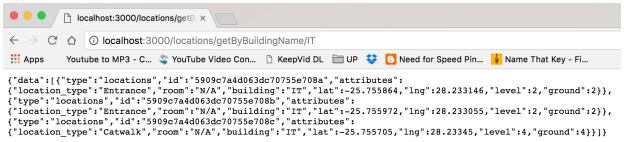


Figure 14: Test results for IT building room test

```
//var assert = require('assert');
//var locations = require('../../controllers/locations/locations');
//var http_mocks = require('node-mocks-http');
var should = require('should');
var needle      = require('needle');

describe('Retrieve Locations by building name function: ', function() {
  it('Should retreive all Locations in building=EMB: ', function(done)
  {
    var building = "EMB";

    needle.get('localhost:3000/locations/getByBuildingName/'+building, function(err, res)
    {
        if (err) { return console.error(err.message); }

        res.statusCode.should.equal(200)
        res.body.should.have.property('data');

        // to display body content
        //for(var i=0; i < res.body.data.length; i++) console.log(res.body.data[i]);

        done();
    });

  });
});
```

Figure 15: Our own test to check EMB

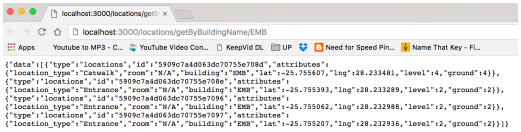


Figure 16: Returned succesfully

```
Retrieve Locations by building name function:
  ✓ Should retreive all Locations in building=IT:

Retrieve Locations by building name function:
  ✓ Should retreive all Locations in building=EMB:
```

Figure 17: Result after test is run: status code 200

2.4.4 Get route

Pass/Fail	Mark (10)	Reason
✓	8	The algorithm returns a valid route from one point to another. Running multiple test cases with different variables proved that the function is efficient.

```
var should = require('should');
var needle      = require('needle');

describe('get Route function: ', function() {
  it('Should retreive route from IT 2-27 - EMB Entrance: ', function(done)
  {
    var routeStartEnd = {
        "locationA" : {
            "location_type" : "Venue",
            "room" : "2-27",
            "building" : "IT",
            "lng" : -25.755990,
            "lat" : 28.233137,
            "level" : 2,
            "ground" : 2
        },
        "locationB" : {
            "location_type" : "Entrance",
            "room" : "N\/A",
            "building" : "EMB",
            "lng" : -25.755391,
            "lat" : 28.233297,
            "level" : 2,
            "ground" : 2
        }
    };

    needle.post('localhost:3000/locations/getRoute', routeStartEnd, function(err, res)
    {
        if (err) { return console.error(err.message); }

        res.statusCode.should.equal(200)
        res.body.should.have.property('data');

        // to display body content
        //for(var i=0; i < res.body.data.length; i++) console.log(res.body.data[i]);

        done();
    });

  });
});
```

Figure 18: Test code between IT and EMB

```
get Route function:
  ✓ Should retreive route from IT 2-27 - EMB Entrance:
```

Figure 19: Result after test is run: status code 200

3 Non-Functional Requirements Tested

3.1 Reliability

Pass/Fail	Mark (10)	Reason
✓	8	The database successfully supports all CRUD operations. It is very consistent and works smoothly.

3.2 Availability

Pass/Fail	Mark (10)	Reason
✓	10	The GIS subsystem is always available and downtime was never experienced during testing.

3.3 Data Integrity and Security

Pass/Fail	Mark (10)	Reason
✓	9	The system is fairly scalable and can handle multiple interactions without actually slowing down

3.4 Transparency

Pass/Fail	Mark (10)	Reason
✓	9	The methods are named clearly and also describe their functionality. It returns locations and all the information relating to the locations(buildings).

3.5 Documentation

Pass/Fail	Mark (10)	Reason
✓	9	All method are documented

3.6 Usabilty

Pass/Fail	Mark (10)	Reason
✓	7	There aren't many accessor to allow full use of the data objects

3.7 Interoperability

Pass/Fail	Mark (10)	Reason
✓	10	This module runs with Javascript which allows it to interchange with other Web based modules easily.

3.8 Scalability

Pass/Fail	Mark (10)	Reason
✓	10	The methods allow for batch processing as opposed to one at a time which is redundant with a module that recieves lots of data

3.9 Performance

Pass/Fail	Mark (10)	Reason
✓	10	Node.js is being used, so the same language can be used on the backend and frontend. Which means it breaks down the boundaries between front- and back-end development.

3.10 Maintainability

Pass/Fail	Mark (10)	Reason
✓	9	Using Javascript objects allows for properties to be added to the object and its easier to retrieve the data and can easily be converted to json.