Mocha And Chai Testing REPORT on

COURSE GAME

DA-IICT

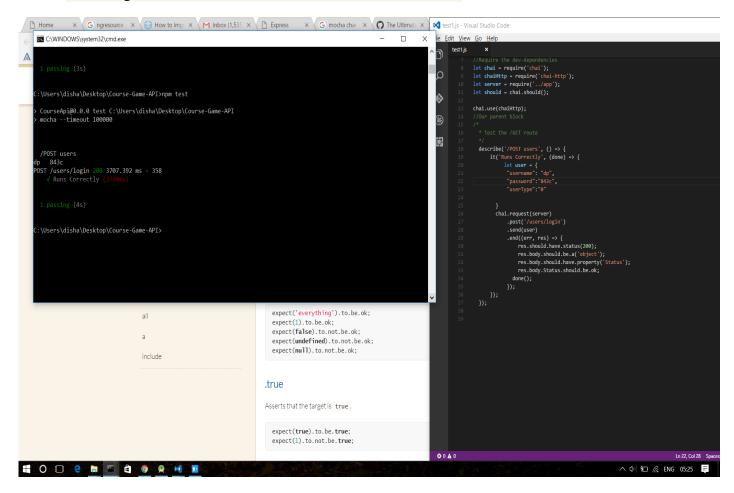**Prepared by Group #02**

For DA-IICT

9th September 2016

What is Mocha?

Mocha is a feature-rich JavaScript test framework running on Node.js and in the browser, making asynchronous testing *simple* and *fun*. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases.

What is Chai?

Chai is a BDD / TDD assertion library for node and the browser that can be delightfully paired with any javascript testing framework.

Following are the Screenshots of the code for mocha and chai

Top screenshot — terminal:

```
  1 passing (4s)

C:\Users\disha\Desktop\Course-Game-API>npm test

> CourseApi@0.0.0 test C:\Users\disha\Desktop\Course-Game-API
> mocha --timeout 100000


  /POST users
dp    84jhjhhj3c
POST /users/login 200 2717.319 ms - 64
    √ Runs Correctly (2783ms)


  1 passing (3s)


C:\Users\disha\Desktop\Course-Game-API>
```

Top screenshot — test1.js (Visual Studio Code):

```javascript
//Require the dev-dependencies
let chai = require('chai');
let chaiHttp = require('chai-http');
let server = require('../app');
let should = chai.should();

chai.use(chaiHttp);
//Our parent block
/*
 * Test the /GET route
 */
describe('/POST users', () => {
    it('Runs Correctly', (done) => {
        let user = {
            "username": "dp",
            "password":"84jhjhhj3c",
            "userType":"0"
        }
        chai.request(server)
            .post('/users/login')
            .send(user)
            .end((err, res) => {
                res.should.have.status(200);
                res.body.should.be.a('object');
                res.body.should.have.property('Status');
                res.body.Status.should.not.be.ok;
                done();
            });
    });
});
```
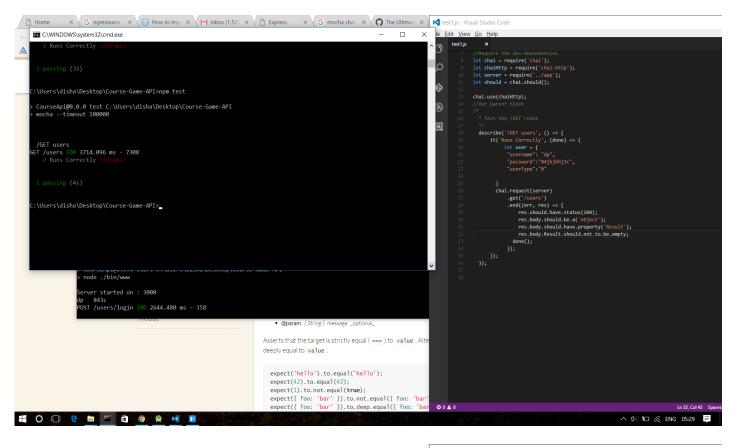
Browser documentation (top):

```
all
a
include
```

```
expect('everything').to.be.ok;
expect(1).to.be.ok;
expect(false).to.not.be.ok;
expect(undefined).to.not.be.ok;
expect(null).to.not.be.ok;
```

.true

Asserts that the target is true .

```
expect(true).to.be.true;
expect(1).to.not.be.true;
```

Bottom screenshot — terminal:

```
npm ERR! Test failed.  See above for more details.

C:\Users\disha\Desktop\Course-Game-API>npm test

> CourseApi@0.0.0 test C:\Users\disha\Desktop\Course-Game-API
> mocha --timeout 100000


  /GET users
GET /users 200 2823.224 ms - 7308
    √ Runs Correctly (2875ms)


  1 passing (3s)


C:\Users\disha\Desktop\Course-Game-API>
```

Bottom screenshot — test1.js (Visual Studio Code):

```javascript
//Require the dev-dependencies
let chai = require('chai');
let chaiHttp = require('chai-http');
let server = require('../app');
let should = chai.should();

chai.use(chaiHttp);
//Our parent block
/*
 * Test the /GET route
 */
describe('/GET users', () => {
    it('Runs Correctly', (done) => {
        let user = {
            "username": "dp",
            "password":"84jhjhhj3c",
            "userType":"0"
        }
        chai.request(server)
            .get('/users')
            .end((err, res) => {
                res.should.have.status(200);
                res.body.should.be.a('object');
                res.body.should.have.property('Result');
                res.body.Result.should.be.a('array');
                done();
            });
    });
});
```

Browser documentation (bottom):

```
all
a
include
```

```
expect('everything').to.be.ok;
expect(1).to.be.ok;
expect(false).to.not.be.ok;
expect(undefined).to.not.be.ok;
expect(null).to.not.be.ok;
```

.true

Asserts that the target is true .

```
expect(true).to.be.true;
expect(1).to.not.be.true;
```

Top screenshot - Command prompt (cmd.exe):

```
√ Runs Correctly (2875ms)

1 passing (3s)

C:\Users\disha\Desktop\Course-Game-API>npm test

> CourseApi@0.0.0 test C:\Users\disha\Desktop\Course-Game-API
> mocha --timeout 100000


  /GET users
GET /users 200 3714.096 ms - 7308
    √ Runs Correctly (3764ms)


  1 passing (4s)


C:\Users\disha\Desktop\Course-Game-API>
```

```
> node ./bin/www

Server started on : 3000
dp   843c
POST /users/login 200 2644.480 ms - 358
```

Asserts that the target is strictly equal (===) to value . Alte
deeply equal to value .

```
expect('hello').to.equal('hello');
expect(42).to.equal(42);
expect(1).to.not.equal(true);
expect({ foo: 'bar' }).to.not.equal({ foo: 'bar'
expect({ foo: 'bar' }).to.deep.equal({ foo: 'bar'
```

Top right - VS Code test1.js:

```javascript
//Require the dev-dependencies
let chai = require('chai');
let chaiHttp = require('chai-http');
let server = require('../app');
let should = chai.should();

chai.use(chaiHttp);
//Our parent block
/*
 * Test the /GET route
 */
describe('/GET users', () => {
    it('Runs Correctly', (done) => {
        let user = {
            "username": "dp",
            "password":"84jhjhhj3c",
            "userType":"0"

        }
        chai.request(server)
            .get('/users')
            .end((err, res) => {
                res.should.have.status(200);
                res.body.should.be.a('object');
                res.body.should.have.property('Result');
                res.body.Result.should.not.to.be.empty;
                done();
            });
    });
});
```

Bottom screenshot - Command prompt (cmd.exe):

```
  /GET users
POST /users 404 7150.918 ms - 9092
    1) Runs Correctly


  0 passing (7s)
  1 failing

  1) /GET users Runs Correctly:
     Uncaught AssertionError: expected { Object (domain, _events, ...) } to have status code 200 but got 404
      at chai.request.post.end (C:\Users\disha\Desktop\Course-Game-API\test\test1.js:29:33)
      at Test.Request.callback (C:\Users\disha\Desktop\Course-Game-API\node_modules\superagent\lib\node\index.js:797:3)
      at IncomingMessage.<anonymous> (C:\Users\disha\Desktop\Course-Game-API\node_modules\superagent\lib\node\index.js:9
90:12)
      at endReadableNT (_stream_readable.js:975:12)
      at _combinedTickCallback (internal/process/next_tick.js:74:11)
      at process._tickCallback (internal/process/next_tick.js:98:9)



npm ERR! Test failed.  See above for more details.

C:\Users\disha\Desktop\Course-Game-API>
```

```
> node ./bin/www

Server started on : 3000
dp   843c
POST /users/login 200 2644.480 ms - 358
```

```
            "year": 2015,
            "semester": 3
        }
    }
}
```

Bottom right - VS Code test1.js:

```javascript
//Require the dev-dependencies
let chai = require('chai');
let chaiHttp = require('chai-http');
let server = require('../app');
let should = chai.should();

chai.use(chaiHttp);
//Our parent block
/*
 * Test the /GET route
 */
describe('/GET users', () => {
    it('Runs Correctly', (done) => {
        //    let user = {
        //        "username": "dp",
        //        "password":"843ghgc",
        //        "userType":"0"

        // }
        chai.request(server)
            .post('/users')
            .end((err, res) => {
                res.should.have.status(200);
                res.body.should.be.a('object');
                res.body.should.have.property('Result');
                done();
            });
    });
});
```

So with Mocha we actually have the environment for making our tests but how do we do test HTTP calls.Moreover, How do we test whether a GET request is actually returning the JSON file we are expecting, given a defined input?

We need an assertion library for that and that's when mocha is not enough .So here it comes the use of chai.