Bytes by Dr. Comp:

Team Members	Marissa Bueno, Dechen Chhemorito, Marc Vucovich, Kevin Kuwata, Zachary Tanverakul, Elijah Berumen
User Acceptance Test Plans	1. User login verification 2. Invalid user verification 3. Playlist generated based off of temperature of entered location

User Acceptance Test Plans:

User login verification:

- Step 1: User must press sign up button and create a new username and password. User should be sent to the playlist generator page.
- Step 2: Quit out of webpage, come back and try to login without signing up first. For this step the user should also be sent to the playlist generator page.

Invalid user verification:

- Step 1: Initialize webpage but user does not sign up with a valid username and password.
- Step 2: User attempts to login without a valid username or password. User should be sent back to login page with empty username and password slots.

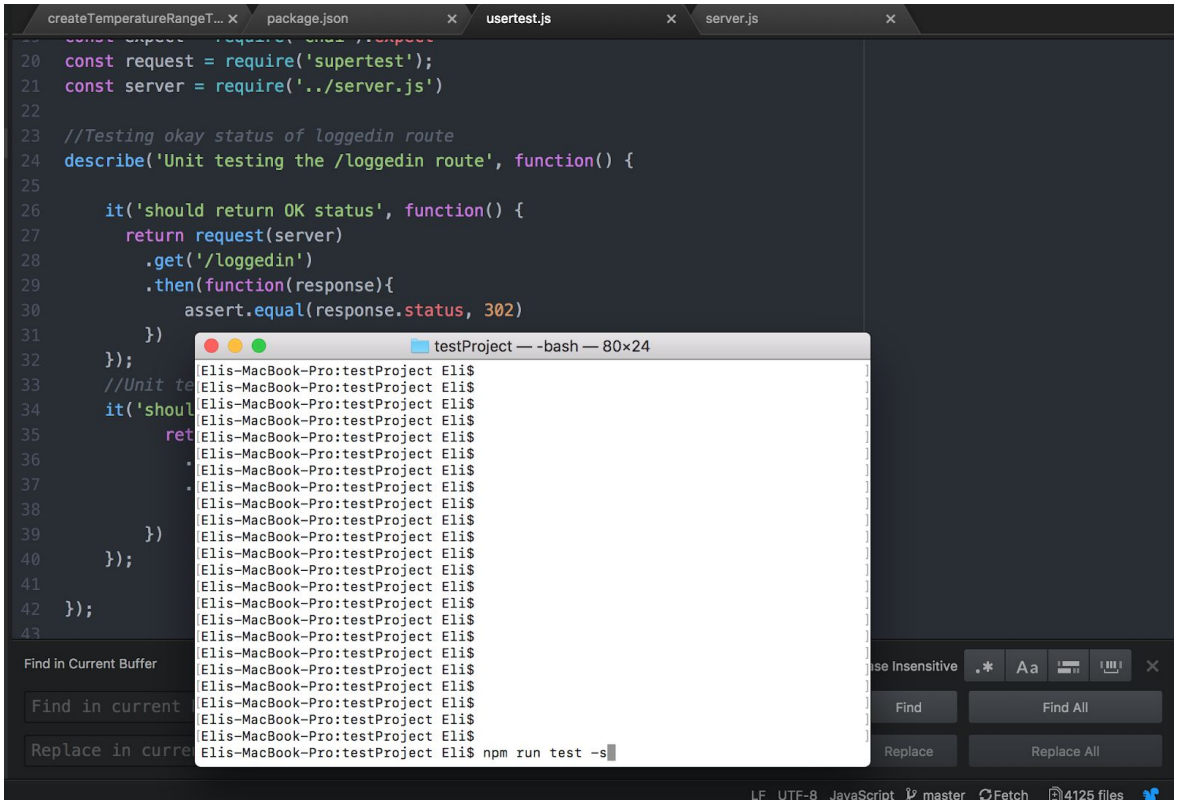
Playlist generated based off of temperature of entered location:

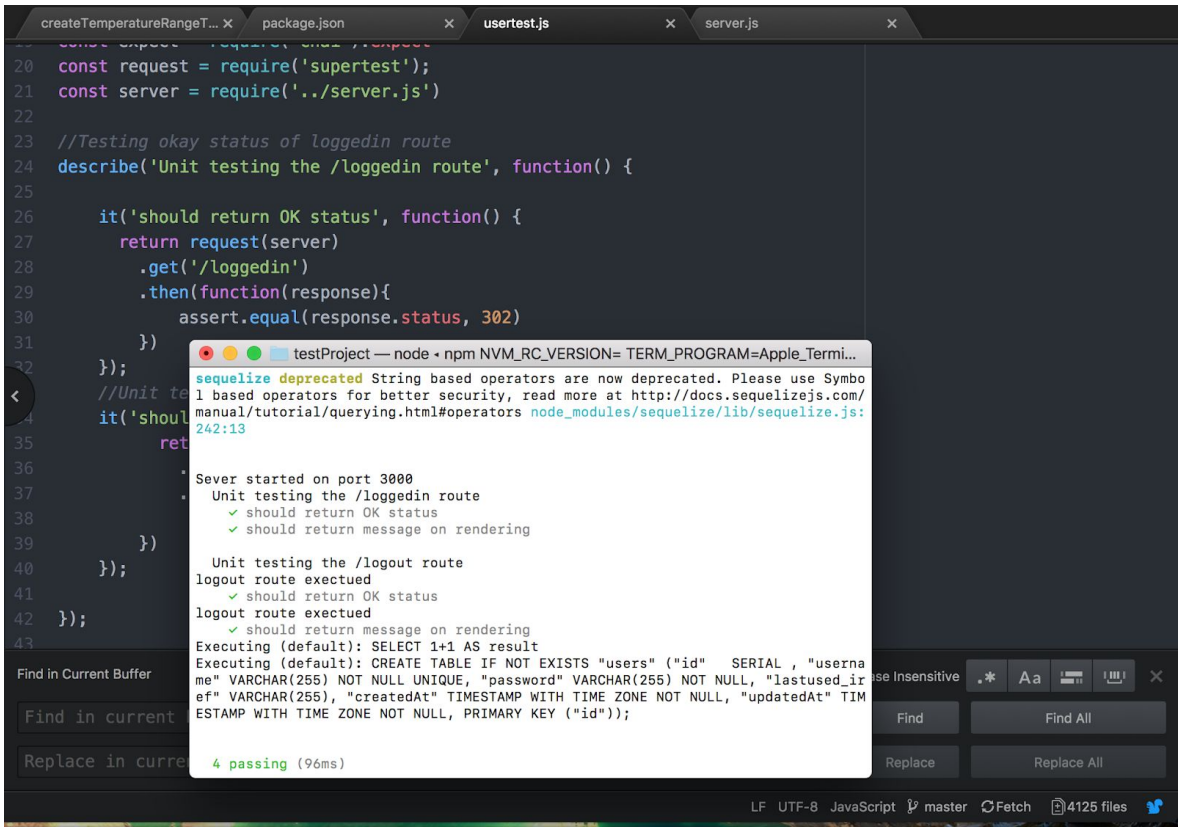
- Step 1: User logs in and chooses a city from the pulldown bar. Generator should display correct weather pattern of city based off of a three tier scale.
- Step 2: Widget at bottom of page must display playlist that invokes the same feeling that that weather does. User will thoroughly enjoy the music as it corresponds to his/her current mood.
- Step 3: User chooses song off playlist, or chooses to play playlist from the begin. Audio can be heard from the computer, and songs get played autonomously until playlist is over.

Unit Testing:

Utilizing Mocha/Chai for Node.js testing:

- Mocha used as the testing structure and to generate/display results(as seen in the package.json test script)
- Chai used for its assertion functions
- Unit tests written in usertest.js found in ‘Elijah’ directory of repository
- ‘Test’ script in package.json updated
- Run these with command ‘npm run test’


createTemperatureRangeT...
package.json
usertest.js
server.js
const request = require('supertest');
const server = require('../server.js')
//Testing okay status of loggedin route
describe('Unit testing the /loggedin route', function() {
it('should return OK status', function() {
return request(server)
.get('/loggedin')
.then(function(response){
assert.equal(response.status, 302)
})
});
});
testProject — -bash — 80×24
Elis-MacBook-Pro:testProject Eli$
Elis-MacBook-Pro:testProject Eli$ npm run test -s
Find in Current Buffer
Find
Find All
Replace
Replace All
LF UTF-8 JavaScript master Fetch 4125 files


createTemperatureRangeT...
package.json
usertest.js
server.js
const request = require('supertest');
const server = require('../server.js')
//Testing okay status of loggedin route
describe('Unit testing the /loggedin route', function() {
it('should return OK status', function() {
return request(server)
.get('/loggedin')
.then(function(response){
assert.equal(response.status, 302)
})
});
});
testProject — node ◂ npm NVM_RC_VERSION= TERM_PROGRAM=Apple_Termi...
sequelize deprecated String based operators are now deprecated. Please use Symbol based operators for better security, read more at http://docs.sequelizejs.com/manual/tutorial/querying.html#operators node_modules/sequelize/lib/sequelize.js:242:13
Sever started on port 3000
Unit testing the /loggedin route
✓ should return OK status
✓ should return message on rendering
Unit testing the /logout route
logout route exectued
✓ should return OK status
logout route exectued
✓ should return message on rendering
Executing (default): SELECT 1+1 AS result
Executing (default): CREATE TABLE IF NOT EXISTS "users" ("id" SERIAL , "username" VARCHAR(255) NOT NULL UNIQUE, "password" VARCHAR(255) NOT NULL, "lastused_iref" VARCHAR(255), "createdAt" TIMESTAMP WITH TIME ZONE NOT NULL, "updatedAt" TIMESTAMP WITH TIME ZONE NOT NULL, PRIMARY KEY ("id"));
4 passing (96ms)
Find in Current Buffer
Find
Find All
Replace
Replace All
LF UTF-8 JavaScript master Fetch 4125 files