Testing Framework For

# Mars Map Maker

By Best Team
Cormac Conahan, Robert Niggebrugge, Seth Hinson

## CIRDLES

CIRDLES is an undergraduate software engineering program at the College of Charleston. They create open-source scientific software and are creating a cyber infrastructure for geochronology. **Mars Map Maker**, for which this testing framework is designed, is one such project.

## Best Team

Seth Hinson- Team Lead

Robert Niggebrugge- Team Technical Lead

Cormac Conahan- Team Scribe

## Testing Architecture

TestAutomation is the main folder where the framework lives. Once opened one will see docs, oracles, project, scripts, reports, temp, testCases, and testCasesExecutables.

**scripts: this directory is where the driver file and main scripts are located that execute the tests**

- runAllTests.sh
- runAllTests.js
- testRunner.js
- verifyResults.js

**oracles: this directory is where the oracle file(s) are created and stored during testing**

- testOracles.json

**testCases: this directory contains the json test cases files used during testing, they are structured with all the info needed for each test to be completed such as the module name, function name, input, and expected output. Each test case name is formatted as TestCaseN.json where N is the test case's specific ID number.**

**testCasesExecutables: this directory contains the modules that are being tested (this is for ease of access for humans, these files are not used by the script and only allow for easy listing of the tested modules)**

**reports: this directory is where the finalReport.css is located, this is also where the actualResults.json and finalReport.html will be created during testing.**

- finalResults.css
- finalReport.html
- actualResults.json

## Test Case and Code Examples

Below is a requirements table for the first 5 test cases. Each of the test cases follow the same format and unit-test similar pieces of code with arithmetic and logic tests. A complete requirements table is available on the team's GitHub.

| ID | Component | Requirement | Method | Input | Expected Output |
|---|---|---|---|---|---|
| 1 | DropDown.js | No object found with key of "value" and non empty string pair, return -1. | entWithContent() | [] | -1 |
| 2 | DropDown.js | No object found with key of "value" and non empty string pair, return -1. | entWithContent() | [{value: ~~}] | -1 |
| 3 | DropDown.js | Object found with key of "value" and non empty string pair, return the last index of the occurence. | entWithContent() | [{value: ~~}, {value: "Foo"}] | 1 |
| 4 | CardList.js | Input Value is an integer, return "number". | typeField() | "8" | "number" |
| 5 | CardList.js | Input Value is empty string, return "both". | typeField() | "" | "both" |

**The code format of any given test case is as follows:**

"module": "DropDown", // chooses the module from which the tested function is retrieved

"ID": 1, // sets the test case ID
"functionName": "entWithContent", // chooses which function from the module to test

"metaData": "Must take in an array of objects and return the last index that contains non empty strings that are present with the key \"value\" or return -1", // short description of expected inputs and expected outputs

"metaDataShort": "DropDown.entWithContent(input) must return -1", // precise expectations of this test case

"input": [], // input for the function to test functionality described above

"expectedOutput": -1 // expected output of the function with given input

## Setup Instructions

**Hardware Requirements:**
Physical machine with resources to run Linux Distribution or physical machine with resources to run Linux distribution via virtual machine.

**Software Requirements:**
Ubuntu 20.04.1 or greater or similar Linux distribution
Node Package Manager
Git

**Step 1:** Clone the repository "git clone https://github.com/csci-362-01-2020/Best-Team"
**Step 2:** Move into the TestAutomation/MarsMapMaker/ and run the command "git submodule update --init"
**Step 3:** In the same directory, run command "git pull https://github.com/hafey1/MarsMapMaker"
**Step 4:** Install npm "npm install"
**Step 5:** Once install is complete, change to the scripts directory and run the quickSetUp.sh script (bash quickSetUp.sh)
**Step 6:** Once the quick setup is complete, run the runAllTests.sh script (bash runAllTests.sh)

### Fault Injections

In order to properly determine whether test cases were properly passing test cases and not just letting everything pass, it is important to inject faults into the code being tested such that it creates an error for the test suite to catch. This is exactly what is happening in the second branch of this testing framework, "mutationBranch."
While in the Best-Team folder type the command: "git checkout mutationBranch". This will move one into the branch where the faults have been injected so that one can run the framework on the newly faulted code. From there, move to the TestAutomation directory and run 'bash runAllTests.sh' to see that several of the test cases now fail.