SamiRoom Website - Team1 Report

Shelly Miron - 316607589
Lea Brodesky - 318191764
Liat Golber -313301129
Beni Segal - 316532886

Our GitHub repository :

https://github.com/Shelly875/SPM-webteam1-SamiRoom

Our Slack :

https://app.slack.com/client/T011959R9P1/C010X3SA89F

Our ClubHouse :

Invite: https://app.clubhouse.io/invite-link/5e7b72e9-35bf-4844-b56b-833e1a053fa8

Our ClubHouse dashboard - https://app.clubhouse.io/webteam2/dashboard

Our Website hosting cloud :

https://infinityfree.net/

Our Website Domain name :

http://samiroom.epizy.com/

Admin FileManager with password and username:

https://cpanel.epizy.com/

username: epiz_25463531
password: bUCx8Cip2mh

Our DB:

MySQL

integrated with the website host server (username&pass of admin):

https://cpanel.epizy.com/panel/indexpl.php?option=mysql&ttt=776627963145224192

Our CI/CD Tool & How we are going to use it in our project

npm:

node package manager, comes pre-installed with node.js.
will be used to download and install packages and manage the dependencies on different machines.
the command:

```
npm init
```

will be used from the command in the project's folder the first time to create package.json.
package.json contains metadata about the project, will used to manage dependencies .
install a package, on the folder of the project using the command : npm install <package name>

if the project was pulled from git, or downloaded, there a need to install all packages dependencies .
to do so, enter the relevant folder for the project using command line and:
1. install all packages used in the project using the command: npm install
2.to check the packages installed used : npm list

ESLint:

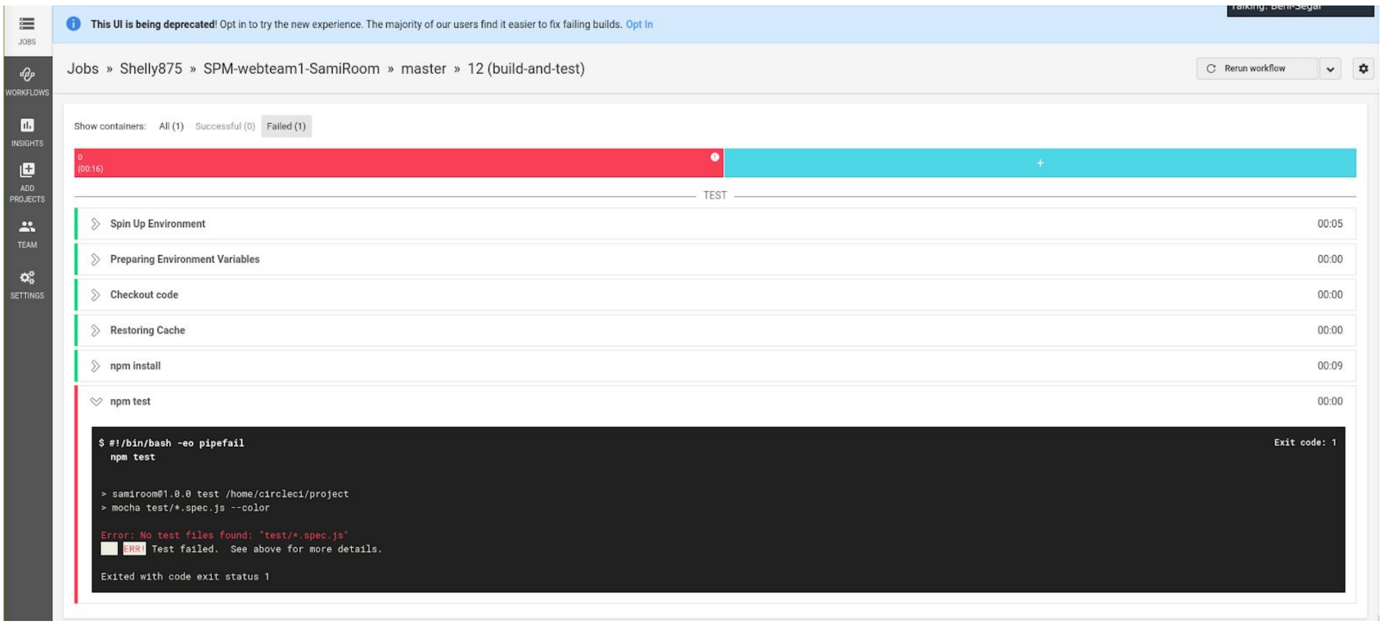
used to identify and report patterns in the javascript code , requires Node.js
stored as a .eslintrc file and configured to lint the code according to eslint rules and our naming convention.

CircleCI:

using the circleci app connected to github for ci/cd
circleci will perform the tests we write in the folder test after each push.

Heroku:

cloud platform used to deploy our project from master branch on github


This UI is being deprecated! Opt in to try the new experience. The majority of our users find it easier to fix failing builds. Opt In

Jobs » Shelly875 » SPM-webteam1-SamiRoom » master » 12 (build-and-test)

Show containers: All (1) Successful (0) Failed (1)

TEST

Spin Up Environment	00:05
Preparing Environment Variables	00:00
Checkout code	00:00
Restoring Cache	00:00
npm install	00:09
npm test	00:00

```
$ #!/bin/bash -eo pipefail                                                    Exit code: 1
  npm test

> samiroom@1.0.0 test /home/circleci/project
> mocha test/*.spec.js --color

Error: No test files found: "test/*.spec.js"
    ERR! Test failed.  See above for more details.

Exited with code exit status 1
```

currently not passing because we don’t have a test to run.

Name Conventions

JavaScript Files -

1. JavaScript programs should be stored in and delivered as .js files
2. JavaScript code should not be embedded in HTML files unless the code is specific to a single session

Whitespace-

1. A keyword followed by (left parenthesis should be separated by a space
2. there should be space after if or while.
   ex: while (true) { }
3. A blank space should not be used between a function value and its invoking (
4. The word function is always followed with one space
5. Every , comma should be followed by a space or a line break
6. Each ; semicolon at the end of a statement should be followed with a line break
7. Every statement should begin aligned with the current indentation.

Comments-

1. Make comments meaningful
2. Use line comments, not block comments. The comments should start at the left margin.

Variable Declarations-

1. All variables should be declared before used.

   ex:
   ```
   let myTitle;       // This is the title of the page
   let startButton;  // This is the button responsible to start the clock

   // Function that use the title
   // Function that use the button
   ```

2. Implied global variables should never be used.

Function Declaration-

1. All functions should be declared before they are used
2. There should be no space between the name of a function and the (left parenthesis of its parameter list.
3. There should be one space between the) right parenthesis and the { left curly brace that begins the statement body.
4. The body itself is indented four spaces. The } right curly brace is aligned with the line containing the beginning of the declaration of the function
5. If a function literal is anonymous, there should be one space between the word function and the (left parenthesis.
6. Use of global functions should be minimized
7. When a function is to be invoked immediately, the entire invocation expression should be wrapped in parens so that it is clear that the value being produced is the result of the function and not the function itself.

Names-

1. Names should be formed from the 26 upper and lower case letters (A .. Z, a .. z), the 10 digits (0 .. 9), and _ underbar.

2. Do not use $ dollar sign or \ backslash in names.
3. Do not use _ underbar as the first or last character of a name
4. Most variables and functions should start with a lowercase letter.
5. Constructor functions that must be used with the new prefix should start with a capital letter.
6. Global variables should be avoided, but when used should be in ALL_CAPS.

Statements-

1. Each line should contain at most one statement
2. Put a ; semicolon at the end of every statement that does not end with a {block}.

Labels-

1. Statement labels should be avoided. Only these statements should be labeled: while, do, for, switch
2. The return value expression must start on the same line as the return keyword in order to avoid semicolon insertion
3. A if statement should have one of these forms:

   ```
   if (condition) {
       statements
   }

   if (condition) {
       statements
   } else {
       statements
   }

   if (condition) {
       statements
   } else if (condition) {
       statements
   } else {
       statements
   }
   ```
4. The for should be avoided, preferring the array methods if possible. When the for statement is used, it should one of these forms:

```
for (initialization; condition; update) {
    statements
}

for (
    initialization;
    condition;
    update
) {
    statements
```
}

5. A while statement should have the following form:

```
while (condition) {
    statements
  }
```

6. A do statement should have this form:

```
do {
       statements
} while (condition);
```

7. A switch statement should be avoided, but when used should have this form:

```
switch (expression) {
case expression:
    statements
default:
    statements
}
```

8. The try statement should have this form:

```
try {
    statements
} catch (variable) {
    statements
  }
```

Assignment Expression-

1. Avoid doing assignments in the condition part of if and while statements.
2. Use the === and !== operators. The == and != operators produce false positives and false negatives, so they should not be used.
3. The eval function is the most misused feature of JavaScript. Avoid it.