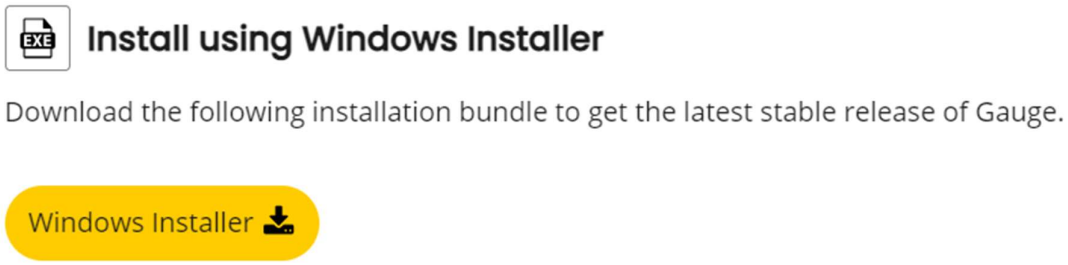
Automating Testing using Gauge and Taiko

Gauge vs Taiko

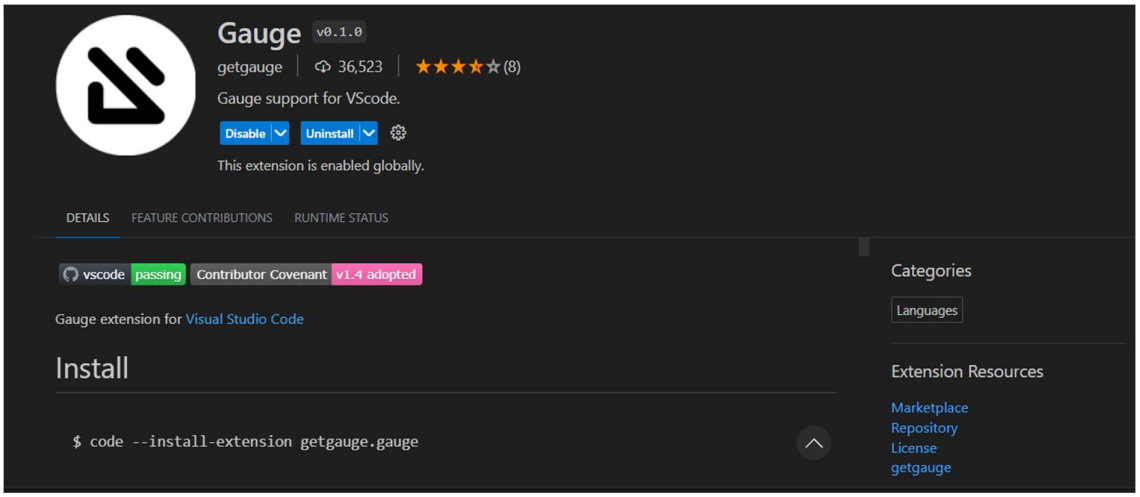
- Gauge: writing readable and reusable acceptance tests in markdown format
- Taiko: API for automating browsers

Setting up the Gauge environment

Installed Gauge using the windows installer.



Installed the Gauge Extension for VS Code Plugin



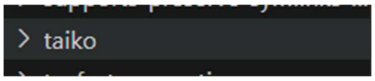
Checked whether Gauge was installed in system by *gauge --version* through CLI.

```
Microsoft Windows [Version 10.0.22621.1848]
(c) Microsoft Corporation. All rights reserved.

C:\Users\kr1st>gauge --version
Gauge version: 1.4.3
Commit Hash: f98dd40

Plugins
-------
html-report (4.2.0)
js (3.0.1)

C:\Users\kr1st>
```

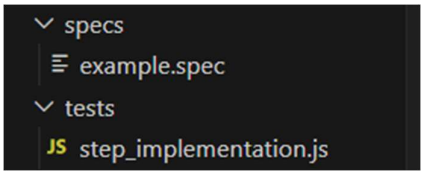
Change to working directory and create a new Gauge project by inputting *gauge init js*

(I chose to work with JavaScript as I noticed that most taiko documentation is in JavaScript)

Make sure the node_modules directory has taiko installed in it.



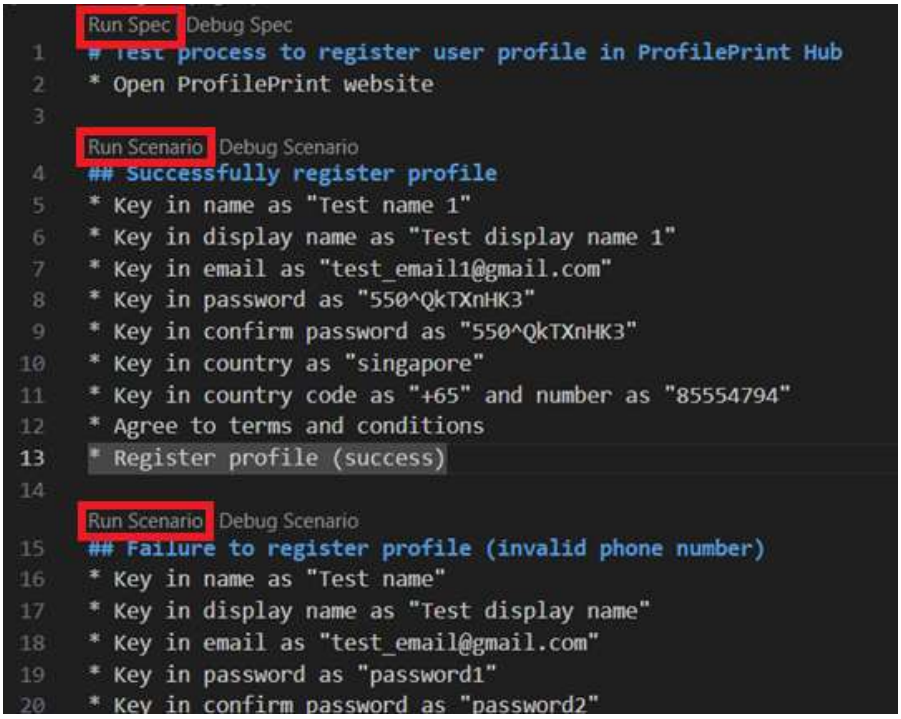
The setup came with sample specification files and test files that will be edited later.



Running the tests

Tests can be run by the command `$ npm run test` in the working directory.

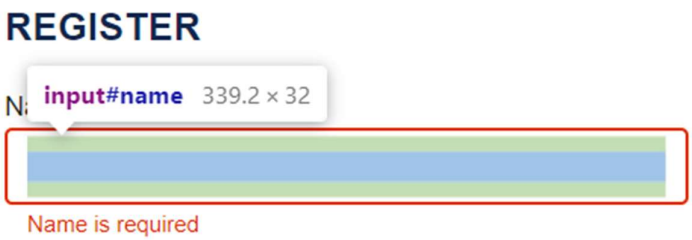
However, using the Gauge VS Code plugin, tests can be run by clicking on Run Spec/Run Scenario (highlighted in red) from the specifications file (registerpage.spec) in the specs directory.



Thinking and Strategy

As this is an automated test to simulate a typical user's journey through the website, I mimicked each step in the scenario to how I first registered my own profile; going step by step through the fields. I also thought of different scenarios as to how an "alert" would be triggered to test the form logic by detecting alerts during fail scenarios.

For any element such as buttons to be clicked and text fields to be filled, I used the respective element id through the “inspect element” feature in chrome.



I can then refer to the element by its element id in code (highlighted in red).

```
step("Key in name as <name>", async function (name) {
    await write(name, into(textBox({id:'name'})),{force:true});
});
```

The different scenarios that I included in the specification were:

- successfully register profile
- failure to register profile (invalid phone number)
- failure to register profile (passwords do not match)
- failure to register profile (password does not meet requirements)
- failure to register profile (invalid email)
- failure to register profile (missing fields)

and it was shown in the gauge generated report that all were successful, which showed that the process was seamless with their respective pop ups to each scenario. I was however unable to add the screenshots to the html report as gauge only adds screenshots when error occurs (such as steps that were unable to run). The html-report (index.html) can be found in the repository under the reports/html-report folder. I also printed the report to pdf and added it to the repository (gauge_report.pdf). The screenshots can be found in the screenshots folder or viewed from README.md.

All in all, I am grateful that I was introduced to Gauge and Taiko as prior to this, I did not know the user interface testing can be automated. I thoroughly enjoyed this project. I would also like to add that I felt the user experience could be improved by enabling the “REGISTER” button when all the form fields have been filled up, instead of always enabling the “REGISTER” button when the user clicks on the “terms and conditions” checkbox even though some fields have yet to be filled up.