Scope and overview

The purpose of the test project is to conduct tests on the internet service Google Maps. Google Maps is a popular geographic location tracker, GPS, and map service, intended to be used in viewing and finding locations. The tests are intended to verify that the website is navigable to users attempting to find a myriad of locations using differing methodologies. This may be implemented in many ways, though the focus will be on first-result search queries, with search result geographical accuracy being a possible stretch goal. Since the project involves conducting UI testing it is important to only utilise UI elements to assess whether the test was successful or not. Therefore the tests will utilise items such as the search bar on both Google.com and Google Maps; buttons such as drop-down menus leading to Google Maps, and other miscellaneous map indicators which may serve the same purpose; as well as a lack of directly navigating through background linkage, as the tests must only use elements available to the user considering their lack of interaction with the background html code. This means features such as location-specific data (websites, phone numbers, directions, etc), as well as other features will be largely ignored if they do not serve the purpose of finding a location on Google Maps.

The overall project timeline is one week, during which the developer will move from one house to another, and will therefore have to contend with unclear work hours. The client company has generously offered to extend the deadline should issues occur, so this shouldn't prove much of a hurdle. A working test environment should be established early, along with a plan for which tests are to be studied. The working time on the implementation should not exceed five full work days, whereupon the remaining two days should be reserved for pre-production preparation, post-production documentation and reporting, as well as post-production last-minute changes before submission.

Test approach

The testing process will span the user navigating to Google Maps through their URL - either directly to the website or through *www.google.com* as a mediator - and inquiring about at least four addresses from the website. If the test manages to reach the Google Maps website (*www.google.com/maps*) and search for an address, the test will be seen as a success, regardless of whether the address matches what the user had in mind; this is because the user was able to navigate to, and through, Google Maps successfully without being prohibited by technological hurdles. Should the stretch goal of geographical accuracy be implemented, a coordinate must be provided alongside the search query. The intent of providing the coordinate is to have a point of cross-reference, whereupon the coordinate of the search result may be used in an AABB collision-check to certify that it is within search parameters of the correct location. This would extend the navigability test scope to the browser providing accurate feedback to the user, and, though not fully necessary, may give some insight as to the search engine's navigability by ways of how well it interprets a user query.

The testing will make use of automated success and failure cases, and mainly be an automated headless environment. A degree of manual testing will be necessary to accommodate for cases in which the test program was unable to find the desired destination, or yield any search result. Though instances like these will be logged, it would be necessary to inspect why such an incident may have occurred. Also of note is the testing software's inability to find addresses which do not clearly make themselves apparent as Google Maps locations, nor cases where more than one Google Maps location exists.

Test environment

The testing environment will be built upon a behavioural strategy pattern, wherein the test function will hold testing data and delegate the different tests to one of many testing environments. These test environments are class instances which handle the same webdriver and data in differing ways to reach the same end goal. The results are then reported back to the main function which logs the test as successful or failed based upon the outcome within that specific test environment. This is to ensure the test environments are given the exact same input data, as well as streamline the implementation in the event of future test implementations or further browser support. Each test run will require a list of addresses to be tested, or the program will simply not run. Additionally, the tests will be carried out across most known or common browsers to ensure that as many methods of navigability across as many different user cases as possible are covered.

The testing environment will be developed on a computer running Windows 10, with internet access, and the tests should take no more space than 500MB on the harddrive of the subject computer. A .text document will be present under *GoogleMapsSeleniumProject*\ in the working directory under the name '*addresses.txt*'- this file must be present and contain at least one row of text for the tests to run. Each address may be however long the conductor of the tests wishes, but, for it to be registered as a separate address to the rest, it must be on a separate row. The provided example data will span four legitimate addresses and one fabricated one to ensure it can handle unsuccessful cases.

To ensure the iterative development model isn't destructive, overall error logging will remain to ascertain that the code hasn't lost functionality, ensuring that a means of regression testing remains. The code should be built and run after each feature implementation and/or major rework, and submitted to the GitHub repository at the end of each work day. This will ensure that if an iteration removes previous functionality the developer will be able to fetch from the previous day's progress.

Testing tools

For the project to be implemented *Google Chrome*, *Mozilla Firefox*, and *Microsoft Edge* should be present on the computer used for the tests. These are included to ensure the program works on as many browsers as possible, whereof the web-drivers will require them to function. In addition to the browsers, WebDriverManager, NUnit, and Selenium, are utilised, but need not be installed as they're included in the project files.

The installation of the browsers should be available from each company's respective downloads page, whereas the additional software is available through the NuGet package manager, or through commands in the PM console. Most of the browsers aren't open source, save for Firefox, though none require a licence to make a testing tool for. All three pieces of development software (WebDriverManager, Selenium, NUnit) are open-source, however, and may therefore be used to make the program without licensing fees, should it ever be released (not going to happen, but it's always good to check) .

Release control

For the submission of the work test I'm planning to have at least one full day to ensure that the program may be imported and set up without issue, as well as to make sure no major bugs or runtime issues remain. Any remaining time should be allocated to writing documentation on the usage of the product, as to make sure no major bugs persist.

The product will be acceptable for release once it can navigate to google maps in four different ways, and reach search results for all of its queries. The searches should result in reaching a Google or Google Maps page, and ideally never crash under incorrect usage. Should coordinate cross-referencing be implemented, an acceptable release would also entail the ability to ascertain if the address search results match with an existing coordinate.

Risk analysis and damage control

Bugs, incompatibilities, and issues regarding the developer's lack of knowledge with Selenium (considering this will be the first time he's used the software) will surely arise. Trouble-shooting will most likely take up one to two days of the five-day software development period, whereof most issues should be addressed. Issues whose trouble-shooting time surpasses half a day should be put to the side and gauged whether their priority is high enough to be continued on. Should the tests be able to be carried out and reach their goal, the issue should be treated as a minor hindrance and given a lesser priority, whereas if an issue prevents the tests from completing their runtime (or even starting it) they should be given greater priority.