Automated end to end testing/automation from user logs

Testing is a crucial part of web app development and can take around (and sometimes over) 30% of developer time. Modern commercial web applications often include logging to track user behaviour on a web site to better optimise web application development. By adapting these logs so that they can be ‘replayed’ automated tests can be created by manually testing a web application once and then using the logs to automatically verify the test each time (after some minor modification to check that some output states are correct)

The purpose of this project is to adapt a minimal user logging system such as (https://github.com/greenstick/interactor (Links to an external site.)) to create selenium scripts that can repeat the web operations that the user performed so that end to end tests of web applications can be trivially created just by manually using the site. The system can assume that controls on web pages have unique ids (so that minor changes to web pages don’t break the selenium scripts).

- 15/01/2021
    - So my project has ended up being “Automated end-to-end testing/automation from user logs.” I think the idea of this is writing user logs in a format that a separate “interactor” program can recognise and replay. I guess the idea is that you can just manually carry out a test and then re-do it using the log anytime it's needed (or potentially marking certain logs to be replayed automatically, like when you update the application). I think it's also specifically for web sites, and presumably it's gonna be designed to work on any website rather than just 1.
    - I have some experience with python but definitely need to refresh myself cause I always forget all its syntax and quirks and stuff. I also played a bit with selenium a bit in my work experience but I remember literally nothing about it aside from the basics. I haven't looked at Python Selenium yet but it's kinda spooking me off the bat. I'm not even sure how it would work in the first place, let alone how to use it.
    - Javascript is in the same boat as python where I've played about with it a fair lot but need a refresher. It might be worth doing a little refresher on some HTML stuff as well.
    - Something scaring me for both javascript and python (though less so for javascript) is using some of the external libraries/scripts (idk what to call them) - I think for javascript you can just pop it in the header and it gets executed or something like that, but for python I think there's a whole process with the package installer and stuff like that which is a bit intimidating.

- 16/01/2021
    - Laptop broke - I've sent it in for repairs but I can't really do anything until it gets back.
- 21/01/2021
    - Laptop still isn't back - i have my phone to look up stuff but without any way to test it i still can't do much - I've been doing a refresher on python and javascript, which is going well.
- 26/01/2021
    - In the middle of this javascript refresher, I'm kinda realising I don't really know HTML too well? I've been doing fine with the basic knowledge I remembered up until now, but a lot of the stuff it's talking about now is going over my head. Luckily though, after digging round dropbox, I found a nice chunky word document of HTML notes I made a while back for some course or something. Reading over them they seem decent enough, so I'm gonna just paste all of it onto a doc for easy reading.
    - I spent a while reading the notes and it's cleared up any questions or whatever I had when doing the javascript refresher. I'm hoping the webpage I'm going to use for testing doesn't have to be too presentable though, cause I remember HTML being a pain to work with when arranging all the different elements to line up and stuff like that.
    - I'm just now remembering we have to make a presentable webpage for week 4. Pain.
    - One thing I'm kinda worried about javascript-wise is that it seems a little...janky? For big web pages and scripts at least. Like everything to do with IDs seems to work entirely by just plaintext, which seems like a recipe for disaster when working with a large page or when making any changes. Probably something to worry about when I actually have any amount of code though.
    - Just got word that parts have arrived for my laptop and its booting up fine. Should have it back soon enough. Woot woot.
- 27/01/2021
    - Nothing really to note here - just worked some more on the refreshers with no issues.
    - I have to use Git for Software Design Principles but the re-learning I did for that will probably come in helpful here - the only real issue I have with Git (namely, interacting with remote repositories with more than one developer) isn't relevant here cause I'm the only one working on it.
- 28/01/2021
    - I kind of forgot about that interactor.js thing so I'm checking it out at the moment. It seems pretty simple to use, looks like it just collects a bunch of

data about how a user interacts with the site and logs it. Only thing that could be an issue is that it sends the log data to a “user-defined server endpoint”, which sounds complex, but hopefully I can just set it up to bung them in a folder on my machine. I *really* hope this doesn’t mean I need to get a server set up.
    - After reading a little more, it seems like you might have to put the script in particular elements within a page? I assumed you could just put it in the header and it would be page-wide, but it seems to be more complicated than that. Additionally, you seem to need to set up a separate script with parameters for interactor, but that seems simple enough, and there appears to be default parameters that I can use when getting it set up anyway.
    - The devs were very nice and left an example site in their documentation, which is quite helpful - it helps a lot with understanding how to format it. Also, in a worst case scenario where I can’t get this working on my own site, I can at least frankenstein the code from their site into mine as a temporary solution. The documentation in general is very good - a few terms are going over my head right now but aside from that it’s very clear.
- 30/01/2021
    - It's more than likely that my initial laptop isn't going to get repaired within an acceptable time period (and even if it does it's probably not a permanent repair) so we've just bitten the bullet and ordered a new one - it should arrive within 3-5 days, which i doubt would have been faster than anything queens could supply.
    - I have some of the IDEs I use on an external drive (visual studio + visual studio code, eclipse) so I should be able to get into using those very quickly. The others I'll need to download but that's just IDLE and Atom which I'm pretty sure are rather small programs so the setup should be quick enough.
    - If I'm being honest I'm extremely bored of those refreshers so I'm gonna shift mainly (not exclusively) to working on coursework for other subjects until the replacement laptop comes.
- 31/01/2021
    - Good news - managed to get this absolutely hulking old desktop that dad had working after a fair bit of tinkering - its very old but was pretty high spec for its time so it still runs pretty well - installing everything I need now.
    - Don’t know what we’re gonna do with the laptop currently on its way lol.
    - Got visual studio code and IDLE downloaded and am setting up a folder with all the resources I have right now.
- 01/02/2021

    - It’s tempting to get straight into developing but I’m thinking it’s probably a better idea to get this how-to-guide website done first, since it’s time sensitive and all.
    - Ok so the idea with the how-to guide is just to document your findings on some part of your project in a format that others can use for help. This could be an issue, considering I don’t currently have *anything* to document aside from just like. Python and javascript. As languages.
    - Ok so the plan is to get interactor.js working at least somewhat and make the how-to based on that - I’ve been pretty busy today but I’m mostly free tomorrow so I can spend time trying to get that working.
- 02/02/2021
    - So the plan is to go through interactor.js line by line and add my own comments describing what I understand it to do, then once that's done, attempt to get it working on a testing page.
    - Ok so that didn’t go as planned. I thought that because interactor is a pretty compact script, I could get through it in a bit more than half a day, but it's pretty complex by my level of understanding. I got about half of it commented overall, skipping over the parts I didn’t understand. I’m getting pretty bored of this and don’t think I’m gonna make much headway like this, so I’m switching over to starting to write the webpage for the how-to guide for the rest of the day.
- 03/02/2021
    - Plan for today is to finish figuring out interactor.js and then add it to a dummy page for testing, making a note of anything I can use in the how-to guide. I can add all of it to the website tomorrow. If I’m being honest, I’m not sure if I can get this done before friday - the whole laptop situation was a bit of a hamstring there - but I may as well try, and see if I’m allowed an extension or whatever after the fact.
    - I’m getting kinda overwhelmed right now and don’t really know what I can do. Like, I thought that I had interactor.js mostly solved but I’m now thinking that was **really** incorrect. I can’t even get either of the sample pages working and I don’t have a clue why. I’ve got them downloaded to my machine, made sure any file paths are relative and it’s able to access anything it needs to and it just does nothing. The more advanced one at least seems to be responding but it doesn’t save a log file *anywhere* and the HTML page itself is really complex for me, let alone how it interfaces with interactor. Looking back over interactor I don’t even know how to start deciphering the complex stuff I saw before, and now I’m not even sure I understand the stuff that I thought I understood.
    - The only thing I know of that could be the problem is this:

```
var interactor  = this,
    // Initialize Cross Header Request
    xhr         = new XMLHttpRequest();

// Close Session
interactor.__closeSession__();

// Post Session Data Serialized as JSON
xhr.open('POST', interactor.endpoint, interactor.async);
xhr.setRequestHeader('Content-Type', 'application/json; charset=UTF-8');
xhr.send(JSON.stringify(interactor.session));
```

- I'm not gonna pretend I understand half of this, but the fact that it's an HTTP request, and that it's using POST, makes me very worried that it means that something has to be ready to receive it - the phrasing of "server endpoint" in the documentation might support this. I tried to find some way to output the JSON.stringify somewhere to make sure it's at least recording the interactions, but I couldn't find a way to since it triggers on the beforeunload event - console.log, document.write and alert don't work cause the only time its called is when the page is being closed. Really don't know how I'm gonna be able to write a webpage about how to solve something when I have yet to get anything in my project working.
- Had a break to calm down a bit and I've decided I'm gonna try and re-jig interactor to try and save a file locally - even if I don't understand most of it I can at least save it's output to a file to get started and have something to write about. Apparently this is impossible to do with client-side javascript, so I have to use node.js and a module called fs (File System) - there seems to just be a method in fs called writefile, so if I can figure out how to include it in interactor it should be a cakewalk to just save the output of JSON.stringify(interactor.session). If I can get this working then I should be able to write about that in the how-to.
- I can't get it working - node.js, turns out, is completely beyond my understanding, and the methods I found online (for client side javascript) of making a blob and offering it as a download, or of storing it in local storage, don't work.
- Ok so I tried to get the blob working again and I have the same method working on a test page, so I'm gonna transplant it to the interactor.js test site and, fingers crossed, it does.
- It works! Partially. I modified the blob download script (although I have no idea how this modification works, it was just a crapshoot) and stuck it at the end of the interactor.js test page - this allows me to download an interactor log file, but it only shows the data collected on the pages initial load - any metadata sent directly with the request, presumably. This was

expected, but I have no idea how to have the json file download with the actually useful interaction information, how I'm going to write my how-to about this, or how **at all** most of this works. It is, however, very late, so tomorrow me can work that out.

- 04/02/2021
    - (Writing this retroactively but this day was all just exceptional circumstances stuff).
- 05/02/2021
    - So the current plan is to set up a local web server with something that can receive the XMLhttpRequest from interactor.js and, with that, those juicy juicy log files. I've installed a very basic version of XAMPP with just Apache and php, which works very well (it was almost *too* easy - I hope I haven't left out something crucial or something like that) - I can access a test page completely fine in the browser using the loopback address. I'm *hoping* that I can just set up a GET xmlhttprequest on this site, and with some fiddling of parameters, use document.write to just output the log file onto this site. If this works, then I have something that's actually useful to write about in the how-to guide (as well as just making progress in the project overall).
    - If this works, then the next hurdle is finding a way to actually save the log files - even if I do manage to get the full log file sent, then I'm still just offloading the file writing problem to a different page.
    - Ok so XAMMP is fine for hosting a page that can send the POST request, but I can't find a way to have the receiver page, well, receive the data from that request. Apparently the best way to do this is to set up a webserver using flask instead.
    - Got a flask server set up with the basic application they had in the quickstart guide. When running flask using the command prompt, it seems to log any get requests - hopefully these extend to post requests, so I can check that interactor output is working well enough.
    - The flask app doesn't immediately seem to be receiving the post request, so I'm gonna leave that until tomorrow and work on some more content for the how to page.
- 06/02/2021
    - Flask required a lil tinkering to get working again today - for some reason it wanted an absolute path when setting FLASK_APP, despite a relative one working yesterday? Its not important, just weird.
    - (Is it bad that I had to google what def was in a python script?)
    - The flask server still isn't logging any POST requests from the interactor page.

- I'm realising I'm probably getting a little ahead of myself - I'm going to try and get a very basic page with a POST request working first, then move onto getting a more advanced one set up - after all, I'm writing about how to set up interactor on *any* website, not just this particular one.
- OK so when the flask server isn't running, the firefox console says that a POST request was sent and returned a 404 - this seems exactly like what we want.
- So the console displays 2 different responses depending on which one of these lines I use:

```
//xhr.open('POST', '127.0.0.1:5000/postmethod', true);
xhr.open('POST', 'http://127.0.0.1:5000/postmethod', true);
```

- If I use the first one, then it returns a 403 forbidden error, and the flask server doesn't show any logs for it. From what I understand, this essentially means that the request is fundamentally invalid in some way.
- If I use the second one, then it returns this error:

- After some reading, CORS just seems like a fancy way to specify which addresses are allowed to access resources (which encompasses most interactions I think?), and I'm guessing this Same Origin Policy is just an inbuilt policy that automatically denies request from the same origin - i.e. requests coming from the same address (in this case localhost). Not exactly sure why this is a thing.
    - OK so I had this backwards sorta - the same origin policy does the exact opposite of what I thought it did, so it actually automatically denies requests that come from *different* origins, not the same origin. I think the reason I'm getting one here is because they're using different ports - the flask server is using 5000 and the POST requester site is, naturally, using 80.
- Ughhhh apparently the way to get rid of this is to install flask-cors but that seems like such a hassle. I'm gonna check if there's another way.
- There wasn't but turns out flask-cors was really simple to install. The CORS errors are gone now (from what I can parse from the flask logs flask-cors allow requests from the loopback address). There's a 500 error now, which is good because it means that the post request is reaching the server fine.
- The 500 was due to a random line I put in while debugging the CORS error - after removing it I get a 200 response :)
- Next step is to try and have the python script flask is using save the data sent by the post request.

    - I imported request from flask, which seems to hold all the data associated with the POST request - feeding it into str just returns the address and request type, so I'm going to dig around in its members for a bit to see where the actual data is kept.
    - Doin a lil dance right now cause it's properly working! Request.json contains exactly what it's supposed to - I'm going to modify the script in the example site a bit to see if I can get it to do the same thing.
    - AAAAAA it's working! I literally just had to change the endpoint in the example page and it sends the data exactly like it's supposed to when it unloads! The json data is big and messy, but I can actually see it now which is a big plus.
    - I don't need to understand the json data right now, since that's all for the second part of replaying the logs, so that can be put aside for now. The next step, after what I feel is a well-deserved break, is to figure out how to implement interactor into any webpage, not just the premade sample one. Once I have that set up, I can go onto writing all the content of the how to guide, and at this pace I'll have plenty of time to pretty it all up.
    - I'm gonna try and write the how-to guide as a sort of beginner-intermediate tutorial rather than an advanced one - I don't think anyone particularly experienced with web dev would have much need for a tutorial since they can probably get what to just by looking at the code or documentation or whatever. I'll go through each of the steps in setting up interactor, maybe leaving out some of the really simple stuff like opening the command terminal or following a tutorial to install python or whatever. The idea is mainly just to help those who don't have much web dev experience and also need to get an interaction tracker running on their site. A *bit* niche, but I imagine some people would find it helpful.
- 07/02/2021
    - (I didn't have time to do much today cause of personal stuff so I don't have anything to write here)
- 08/02/2021
    - So the plan for today is to figure out how to set up interactor on a webpage - how to customise it and stuff like that.
    - Ok so that was easier than expected. You literally just call the script in the body, change the class attribute of any elements you want to track to "interaction", and optionally add a script with some input parameters.
    - As far as the how-to guide is concerned, that's all the practical stuff that's required, so I'm just gonna spend the rest of the extension writing the content for the how-to guide and making all pretty n stuff.
- 09/02/2021

    - I wrote some more of the guide content and then spent a while formatting it all nice.
- 10/02/2021
    - The guide content is all written up now, I think, and I've formatted it all in a way I'm happy with. Just need to fix a file download, maybe add some more links and images, and I should be good.
- 12/02/2021
    - So I had a little poke around the json interactor outputs, and made a few findings:
        - loadTime, unloadTime, platform, and port aren't useful - they aren't relevant to testing (or rather, I don't know of a way they could be replicated for testing).
        - Language might be useful? I think that's just sent with the GET request?
        - clientStart/clientEnd isn't useful
        - page isn't useful
        - endpoint isn't useful
        - interactions is all the useful stuff - contains an array of interaction objects with the following attributes:
            - type - i think this is either interaction or conversion, but I don't really know what a conversion is.
            - event - the event that triggered the interaction to be logged
            - targetTag - just the tag of the element being tracked? What is a tag.
            - targetClasses - just seems to be interaction in all the logs I've made - I imagine it could also be conversion, or perhaps any additional classes the element has?
            - content - just the content of the element being tracked.
            - clientPosition - the x y coordinates of the event, relative to the browser window.
            - screenPosition - the x y coordinates of the event, relative to the entire screen.
            - createdAt - the time when the event occurred - I don't imagine this could be useful.
    - One glaring thing missing is the ID of the element being tracked, which is near essential for recreating the tests. I don't know if any of the elements I tried tracking had an ID, so I'm gonna see if interactor would track it if it had one, or if some modifications need to be made.

    - Yeah unfortunately interactor doesn't seem to log element IDs, which is gonna be a bit of a problem. I'll have a poke around interactor and see if there's a way to add it in easily.
    - Fortunately, the fix for adding ID was really easy - just add the ID to the attributes the json saves, in the same format as all the others. It outputs the ID as it should now.
- 13/02/2021
    - I *think* that I won't need to mess around with interactor anymore - the next thing on the agenda is to have a mess around with selenium to see how it works, and then do the same thing with Python Selenium.
    - If interactor already logs everything needed to replicate an interaction, then I should be able to leave it alone, and shift to working entirely on the log replay function.
    - I made a github repo for the project - a bit overdue but whatever.
    - Setting up Python Selenium requires downloading some chrome driver and adding it to $PATH, whatever that is - I'm gonna do some research to see what that means.
    - OK so adding a folder to $PATH is really easy, though I'm not sure why Python Selenium needs it. I installed the rest with pip and it seems to be set up now.
- 15/02/2021
    - So the Python Selenium docs have a little test script so I'm gonna test that out and then dissect it a little.
    - Ughhhh it doesn't even work because of the obnoxious amount of pop ups on a fresh chrome install. Pain.
    - So most of Python Seleniums functions use css paths to specify elements, not IDs. Gonna go figure out what these are.
    - I can't figure out what css paths are so I'm just gonna see if interactor has something I can use to save them.
    - It does. Just an attribute like ID. This should work fine.
- 17/02/2021
    - So Python Selenium uses a pretty simple set of functions for interacting, so I'm gonna go through them and see if there's anything I can't replicate.
- 19/02/2021
    - I can't find any obvious holes in things that Python Selenium can't replicate using its own commands so that's good. I'm a little worried though cause in C# you could make your own events to trigger whenever, so if javascript has a similar deal going on that could be problematic.
    - I know you can use Javascript inside of Python Selenium but I'm not entirely sure that's a good idea to use ubiquitously. I imagine debugging a

program inside a program that's also written in a different language could get wacky real fast, especially as this project grows from little 10 line scripts to an actual working project file.
    - I think it might be an idea to have the Javascript section as a fallback maybe? Like I'm gonna assume that a switch statement is gonna be how the program decides which Python Selenium equivalent event to run, and in that case, it might be wise to have some javascript in the default case that can run any event fed into it. This is mostly just speculation at this point, I haven't tinkered with the program itself too much.
- 21/02/2021
    - I got some work done on the structure of the blog part of this, but I don't think I'm gonna be getting much work done this week. A whole load of SDP coursework just presented itself, as well as a lovely little databases test.
- 25/02/2021
    - The coursework and test are done but I am completely barren of motivation at the moment. I'm just gonna do busy work today like planning the blog post and styling it up and stuff.
- 27/02/2021
    - I'm only now just finding out that there's a deadline for the first draft of the blog post so it may be scramble time in that regard. I think I'm set content wise though, I have plenty to write about and all.
    - The idea with the blog post just seems to be a pretty detailed summary of the project, the work done so far, and any future goals. The intent is to allow anyone to understand the project's concept, its current progress and what they need to work on it more without too much effort (the assignment says they should be able to build on it with less than a week's worth of work).
    - There's also an emphasis on making sure that a student can get the code running easily, so it's almost a continuation of the how-to guide for the rest of the project.
    - Once I get tired of blog post stuff today I'm going to continue going through that big list of HTML events and writing their selenium equivalents if needed.
- 28/02/2021 > 03/03/2021
    - Just continuing with blog post/event sorting and conversion stuff.
- 05/03/2021
    - I'm a little paranoid that my blog post is a little too anecdotal at the moment - the example one was similar, but focused a little more on the technical detail of the project and all the hurdles and stuff. I feel like mine

is less “focusing on technical detail” and more “recounting my repeated failed attempts at development in entirely the wrong areas, followed by me realising what I’m actually supposed to be doing and finishing it in a day.” Perhaps making it more brief would make it easier to digest, even if it does mean less content. It’s kinda got the recipe website problem right now where you’re having to read my life story instead of just the technical details you want. It’s still the first draft though, so I think I’m going to add the content first and figure out what to cull later.
    - I also need to think of a name for this fake blog because currently it’s just

- 06/03/2021
    - Submitted the blog post.
- 08/03/2021
    - I made some changes to the blog post in areas I wasn't entirely satisfied with - mainly just fleshing out details, adding more instructions on how to do particular parts (or adding links to other places that do) and adding more screenshots of code and stuff.
    - I've also been continuing with the selenium conversion stuff, and I'm near done with it, but I'm not sure if I've sorted all the events correctly - there's a few that are hard to separate between replicable or non replicable. Regardless, I'm gonna push on and try to get a working prototype soon - the event stuff seems like it can be figured out after that.
- 10/03/2021
    - All the events bar a few are converted now so I'm just writing the bit of the program that parses which event gets executed. Simple enough, just a matter of peeking at the json file and feeding the event into a switch statement.
- 13/03/2021
    - So after getting started with the parser (it wasn’t as simple as i thought it was), I’m now realising that there are 2 main ways I could write this part:
        - I make the application explicitly replicate only specific events - modify interactor to save certain properties of the event object depending on what its type is, enough so that a Python Selenium script can replicate it close enough for almost all uses. In the future I could theoretically add more events recognised by interactor and able to be replayed by the Python Selenium script, but this would require manually adding each event - not particularly good, design-wise. This may seem like the obvious choice just for simplicity, but there are a ***lot*** more user-triggered events in HTML

than you'd think - it ain't just clicks and types we're dealing with here, there's doubleclick events, cut events, copy events, scroll events, touch events, drag events, PRINT events (like actually printing the webpage on paper who even DOES this), and I could go on. And a website could potentially respond uniquely to each of these events, so if they're not being tracked or replicated then the testing could be completely useless for certain pages.
        - I abandon Python Selenium for everything except for just navigating to the webpage and running javascript code, and use javascript for simulating all the events. I'd written this off as unnecessarily complex before, but I've recently come across something that's starting to change my mind - the dispatchEvent method. This is a method able to be called by any element on a page, and executes an Event it takes as a parameter on that element. For example, if i created a click event and had a button element call it, it would be the same as if the user has triggered that event. This means that if I modified interactor to just save the event object (or enough info to recreate it), then I could just feed the event into dispatchEvent on the element it was targeting - this would work with literally any event, making it much more future-proof (and not to mention easier).
    - In case it wasn't obvious, I'm leaning towards the second, despite the sunk-cost fallacy having pulled me towards the first for a while - RIP to all the selenium equivalents of those events I wrote.
    - The only thing stopping me from just going ahead with the 2nd one is a *miniscule* snag in my modified interactor script that I just can't figure out. I just can't figure out a way to get this object containing all the event properties to send with the xmlhttprequest. For some unknown reason, the toJSON method of this object has been overwritten to only provide 1 random property - I can see everything it holds if I use console.log, but if i try to convert it to JSON using JSON.stringify then it just gives me that property and not the actual object, since stringify will try to use toJSON in place of its default function if it is specified. I've tried so many things, up to and including literally undefining toJSON (I can SEE it as undefined in the dev console) and it still gives me that useless property. It's infuriating.
    - Don't know how to fix this so I'm giving up for tonight.
- 18/03/2021
    - Taking another crack at this but I'm not sure where to start. My first thought was just to see if I could send the object containing the data as XML instead, but there's no native support for xml serialisation, and I don't

know how I'd set up flask to receive it. I would just make my own stringify method but I don't know how to do that when this object could be a whole lot of different events, all with their own properties.

- 19/03/2021
    - Turns out you *can* write a method that can print the properties of an object, no matter what type that object is? Like you can loop through the properties and print their name and value (or rather, key and value). Javascript is like that apparently. Neat.
    - Currently trying to write my own stringify method as mentioned above, or as its properly known, a deep copy method. All the existing deep copy functions don't work so I have to do this.
    - Gonna be honest, I think I'm gonna have to throw in the towel on my preferred method. I can't find a way to deep copy this event, almost at all. The entire thing is filled with circular references, and I can't just disregard the ones which trigger those errors since I can't dispatch the event without them. I've tried a *lot* of different deep copy methods and each of them have had something I can't fix. With only 10 days until the deadline, I'm planning on taking the safe option - committing the pretty meager progress from this onto another branch and hashing out that selenium parser and CLI for the events I have converted, then modifying interactor to save properties needed to replicate each event depending on a switch. After that I can revise the blog post and the how-to guide (though I'm not sure exactly what we're supposed to revise for the how-to? I thought that was a done thing by now), write that little social media post, and then a testing plan if I have time. If I have time left I might tinker some more with it.
        - If I have time (and I don't think I will) I could try a hybrid version that seems easier, although not as versatile. Dispatchevent is *still* a thing after all. The plan would be to just save the very barebones of the event's properties - literally just the name, the element it targets, the time it happened and anything needed on a per-event basis to create the event. I can then just put a switch determining the event type inside Python Selenium and then use the constructor of that method in javascript to create and execute it at the target element. This isn't as future-proofed or elegant as the second method but its better than the first.
    - I'm really not feeling too good about this. With the second method it felt like I was making something genuinely useful - hell, I would use that if it functioned how I planned but like. With this method? It feels like I've worked hard on this and achieved absolutely nothing. Like I spent all this time and all I have to show for it is a modified version of someone else's

script and a shoddy little Python Selenium script near useless to most people doing proper web testing. I think I'd feel a lot happier about it if I can get the hybrid version I mentioned above up and running, but I'm aware I don't particularly have a lot of time left to do it. I'm going to do the revisions and leave a space to modify them depending on which version I end up keeping with, depending on how fast I get that done.

- 20/03/2021
    - The main revisions to the blog post I want to add are:
        - Update progress section to add current progress (minus version of implementation as mentioned above)
        - Add more technical detail for each section, explaining the process of each part more and adding screenshots and links to guides - currently its a bit sparse in that regard
        - Trim anything too wordy for its own good - this needs to be readable for people trying to pick up the project - keep technical detail but don't get bogged down in too much else.
            - Considering removing the parts of the post that essentially just document my various failings during development. Unfortunately this is the main bulk of the content. It's probably better for readability even if it does leave the post kinda short.
- 21-24/03/2021
    - Continuing work on the blog post - I'm pretty sure I won't have the time to do the good method now.
- 26/03/2021
    - Ok so turns out I actually have way more time than i thought i did and i almost definitely have time to do the hybrid method.
- 27/03/2021
    - Current plan is just to get a mousedown and keypress replicator working and then expand from there, after finishing the blog post and testing plan and the like.
    - Modifying the logging to save different attributes depending on the event has worked with no issue - I'm going to make the replicator with dummy data first before integrating them.
    - Roadblock with Python Selenium - it doesn't navigate to the webpage anymore, just to what is apparently its default value.
    - I forgot that the test gets run at the end of the script and there was an error in the way. Fixed now.

    - Another roadblock with some error about a default adapter and bluetooth? For some reason? Fixed it with a code snippet that suppresses that error so let's hope that won't come back to bite me.
- 29/03/2021
    - Current roadblock is a weird one - the page doesn't seem to be showing the events that are dispatched. Like, a mousedown event on a button isn't triggering the alert that should happen, and a keypress inside a textbox isn't showing any letters. I assumed it just wasn't working, but interactor is logging them. Like, I get a full log of all the events that supposedly happened but the page shows none of them. I think it might have something to do with them getting cancelled somehow after interactor logs them but before they actually happen. Will figure it out tomorrow.
- 30/03/2021
    - Still can't find the solution to this and there doesn't seem to be anything online with a similar issue, except for a single stackoverflow comment that basically amounted to "yeah sometimes it doesn't show up what can you do." Easter break is coming up (or maybe its already started?) so I'm planning on taking the time to finish up other coursework for the majority of it.
- 11/04/2021
    - Databases and SDP coursework is ready for submission now so I'm coming back to this. Kinda forgot about that bug.
    - After about a day of debugging I'm disheartened, to say the least. Nothing - at **all** - has even slightly fixed this. Complete radio silence from the page. Need to decide what to do next.
    - I've decided to try and make a complete version of the selenium version of the script tomorrow. I know I said that it sucked and I didn't like it, but I am getting absolutely nothing from this method and that deadline is getting closer. I'm gonna get a version working that just replicates clicks and keystrokes, almost as insurance. Once that's done, I get 1 day of trying to fix the javascript version before I abandon it completely and focus on something I can actually deliver.
- 11/04/2021
    - Insurance script is more or less done, gonna tweak it in a minute. It's taken a load off my chest for sure.
    - Yeah it's done now. Handles keypresses and clicks and I need to work on making it not throw an error with alerts.
- 12/04/2021
    - I know I said I'd try and get the old script working today but I'm gonna spend it on the write up.

- 13/04/2021
    - Tried to get the old script working today but to no avail - scrapping it from now on but I'll mention it in the potential improvements section.
    - Also made some amendments to both this diary and the blog post to correct something - turns out I've been using *Python Selenium* this whole time and not *pySelenium* - which are apparently entirely separate things. I thought all the tutorials and stuff I've been using were for pySelenium but they were for Python Selenium - in fact, I can't find anything about pySelenium anywhere but it's github page. Weird.
- 14/04/2021
    - Spent all of today sprucing up the blog post - adding images and links and updating some of the content. I think the first draft for the final version (if that makes sense) should be done tomorrow and I can ask some people for criticism and stuff either late tomorrow or the day after.
- 15/04/2021
    - Finished the final first draft and sent it to the group chat for Justin and Alex for feedback.
    - Kinda confused about what exactly we're supposed to do for the testing plan. If it is the whole shebang with that huge table of each and every test, it's not ideal but I have enough time to do it - I've already made a little note of what kinda tests need done for each part and it's more just time consuming rather than hard to figure out. I'm not sure that's what we have to do though - this is the only description I could find of what to do:

      > A testing plan documenting how your project can be manually or automatically evaluated to check it is working as intended and to determine how well it works.

    -
    - The way this is worded + the fact that there wasn't really any guide or forewarning about this *could* imply that it's more just an informal doc saying in general what types of tests should be done on each part and explaining it
- 18/04/2021
    - I'm just gonna move ahead with the table of tests for now. Pretty easy, if time consuming - I'm just converting the general test bullet points I have into proper ones, then filling in normal, erroneous and extreme data for any that can have test data.
    - I'm also gonna lay out some sample tests for the features in the "potential improvements" section of the blog post.
    - I'll probably need a half day or so to actually carry out the test plan (although if I'm really pressed for time I can just pretend I did them).
- 19/04/2021

    - Test plan is done aside from test data, which I wanna do either today or tomorrow.
    - Not exactly sure how to update the how-to guide - probably just tweak the writing a bit, maybe add a troubleshooting page if I have time.
- 20/04/2021
    - Test data is filled in - didn't know how certain tests would have test data so they've been greyed out.
    - I also added some long overdue functionality to receiver to make a new dated file for each log. Very simple to implement.
- 21/04/2021
    - Found out that the test plan doesn't have a set structure and can be anything reasonable provided you justify it. I think my one is a pretty normal test plan, if a bit general - the tests are pretty clear and easily repeatable for anyone making tweaks on the project, and provides a good lot of tests for future features.
    - For the blog post, I've decided to add another page - a quick startup guide that strips away all the anecdotal stuff in the blog post and just concisely tells the reader how to get the project up and running - this is the best of both worlds in my opinion - readers just having a look at the project can skip straight to the startup guide and those so inclined can listen to me systematically documenting my various developmental incompetencies.
- 22/04/2021
    - Wrote the social media post and most of the startup guide today - planning to just post it on r/python - it's what most of my content is written in and they seem to be marginally less aggressive towards amateurs there than on places like r/webdev, judging by the reception in the comments and rating. I don't want to waffle 'cause redditor developers can be weird and they seem to get really mad if something is too wordy. They scare me.
- 23/04/2021
    - Startup guide is finished - I don't really have anything else to do, but I may as well wait until the extension deadline before submitting in case I suddenly remember something I've forgotten to do or the like.
    - I'm not sure how I feel about this project. I feel like in its current state it is marginally useful, and that with some work to make it more user friendly it could actually be a solid application, but, as it stands, I could think of nothing worse than continuing to work on it for at least another month or two. I don't think any developers will be picking it up either - it's just too barebones at the moment to pique much interest, I think. Still, maybe I'll blow the dust off this in a while and flesh it out properly, and I certainly know python and javascript a lot better than I did before (although

javascripts eldritch machinations can still never be comprehended by mortal minds).

- 24/04/2021
    - I got bored and tweaked some of the write up.
- 25/04/2021
    - Apparently I have powers of premonition ‘cause I *did* forget something! Haven’t updated the github repo in a while. I managed to partition the commits somewhat through creative use of file copying and git stashes. If a scramble to fix something shortly before the submission isn’t emblematic of this project, I don’t know what is.