Boiler plate code in R for Section 2 Scenario 4

```
if(!require(RSelenium)){
  install.packages(RSelenium)
  library(RSelenium)
}

rD <- rsDriver(browser = "chrome", check = FALSE)
# If you encounter error message with the above line requiring your chrome driver to be one or more specific
# versions, try the following:
# 1. rerun the line with check = TRUE once to download the latest chrome webdriver versions (or other
# browsers if you don't use chrome);
# 2. go to your browser & check its version (should be found under settings or similar);
# 3. enter binman::list_versions("chromedriver") (or other browsers) to confirm you now have the webdriver
# for the version checked in step 2;
# 4. rerun the line once again with check = FALSE & your preferred version specified
# Please note that you may need to quit & restart your R session for everything to work smoothly.

driver <- rD[["client"]]
driver$open()
driver$navigate("https://sprs.parl.gov.sg/search/home")
Sys.sleep(2)

# Get search box and fill it up
search <- driver$findElement("css", "#divmpscreen2 > div.row > div:nth-child(1) > div > div:nth-child(1) > input")
search$sendKeysToElement(list("enter search term here"))

# Uncomment the following lines to only search in titles
# checkbox <- driver$findElement("css", "#divmpscreen2 > div.row > div:nth-child(1) > div > div:nth-child(2) > label >
input")
# checkbox$clickElement()

# This will select the 13th parliament
session <- driver$findElement("css", "#divmpscreen2 > div.row > div:nth-child(1) > div > div.form-group.byParText >
select > option:nth-child(14)")
session$clickElement()

# Find submit element and click
submit <- driver$findElement("css", "#divmpscreen2 > div.row > div.col-sm-12.text-right.pull-right > div > button:nth-
child(2)")
submit$clickElement()

print("Search parameters submitted.")
suppressWarnings(rm(search, checkbox, session, exact, submit))

# Sleep momentarily for result to load
Sys.sleep(2)

# helper code for switching windows (from https://github.com/ropensci/RSelenium/issues/143)
myswitch <- function (remDr, windowId) {
  qpath <- sprintf("%s/session/%s/window", remDr$serverURL, remDr$sessionInfo[["id"]])
  remDr$queryRD(qpath, "POST", qdata = list(handle = windowId))
}

# Switch window and check for number of search results
myswitch(driver, driver$getWindowHandles()[[2]])
num_results <- driver$findElement("class", "showingResults")$getElementText()
res <- rev(strsplit(num_results[[1]], " ")[[1]])
num_pages <- ceiling(as.integer(res[1]) / as.integer(res[3]))

# Create empty list to store results, & tracker for current result under examination
res_list <- vector("list", as.integer(res[1]))
current_res <- 0
print(paste("There are", res[1], "results in", num_pages, "pages to click through."))
rm(num_results, res)

# Nested for loop to click through all search results
for(click in seq(1, num_pages)){
  print(paste("Search result page:", click, "of", num_pages))

  # This assumes that 20 search results are returned
  for(item in seq(1, 20)){
    # Switch to search results tab
    myswitch(driver, driver$getWindowHandles()[[2]])

    # stop if all results have been examined (this handles the exception on the last
    # page, which may have fewer than 20 search results)
    current_res <- current_res + 1
    if(current_res > length(res_list)) break

    # Get element to click on, to see each individual page with content
    elem <- driver$findElement("xpath",
                               paste0("//*[@id='searchResults']/table/tbody[",
                                      item,
                                      "]/tr[1]/td[2]/a"))
    elem$clickElement()

    # Sleep momentarily for result to load (if your results contain nothing but
    # html tags, increase sleep time)
    Sys.sleep(2)

    # Switch to page with content and get URL name
    myswitch(driver, driver$getWindowHandles()[[3]])
    item_key = driver$getCurrentUrl()[[1]]
    item_key = gsub("\\?", "_", rev(strsplit(item_key, "/")[[1]])[[1]])

    # Append result to list for later processing
    res_list[[current_res]] <- driver$getPageSource()[[1]]
    names(res_list[[current_res]]) <- item_key

    # Write out each page source as a file
```

```
    writeLines(capture.output(XML::htmlParse(res_list[[current_res]])),
               con = paste0(item_key, ".txt"))

    # Close tab
    driver$closeWindow()
  }

  # Switch back to search results tab
  myswitch(driver, driver$getWindowHandles()[[2]])

  # Once 20 results have been saved, stop if on last page, click on next page otherwise
  # (Next page button's relative location changes after first 20 results are shown,
  # hence the need for alternative xpaths)
  if(click == num_pages) break
  next_page <- if(click == 1){
    driver$findElement("xpath", "//*[@id='searchResults']/div[3]/section/ul/li[1]/a/em")
  } else {
    driver$findElement("xpath", "//*[@id='searchResults']/div[3]/section/ul/li[3]/a/em")
  }
  next_page$clickElement()

  # Sleep momentarily because next page takes a while to load
  Sys.sleep(2)
}

print(paste("Search completed.", num_pages, "pages clicked through for", current_res - 1, "results."))

driver$close()
rD[["server"]]$stop()
rm(driver, rD, click, current_res, elem, item, item_key, next_page, num_pages, myswitch)
```

Boiler plate code in Python for Section 2 Scenario 4

```
from bs4 import BeautifulSoup as bs # Note: Hint: suggest using bs for subsequent parsing of HTML source
from selenium import webdriver
import time

driver = webdriver.Chrome('input path to chromedriver if not added to PATH')

page_url = 'https://sprs.parl.gov.sg/search/home'
driver.get(page_url)

# Get search box and fill it up
search = driver.find_element_by_css_selector('#divmpscreen2 > div.row > div:nth-child(1) > div > div:nth-child(1) >
input')
search.send_keys('COS')

# Uncomment following two lines to only search in titles
#checkbox = driver.find_element_by_css_selector('#divmpscreen2 > div.row > div:nth-child(1) > div > div:nth-child(2) >
label > input')
#checkbox.click()

# This will select the 13th parliament
session = driver.find_element_by_css_selector('#divmpscreen2 > div.row > div:nth-child(1) > div > div.form-
group.byParText > select > option:nth-child(14)')
session.click()

# Find submit element and click
submit = driver.find_element_by_css_selector('#divmpscreen2 > div.row > div.col-sm-12.text-right.pull-right > div >
button:nth-child(2)')
submit.click()

print('Search parameters submitted.')

# Create empty dictionary to store results
res_dict = {}

# Switch window and check for number of search results
driver.switch_to.window(driver.window_handles[1])
num_results = driver.find_element_by_css_selector('#searchResults > div:nth-child(1) > div')
res = num_results.text.split(' ')
num_clicks = int(res[-1]) // int(res[-3]) + 1

print('There are {} pages to click through.'.format(num_clicks))

# Nested for loop to click through all search results
for click in range(num_clicks):

    # This assumes that 20 search results are returned, which are 1-indexed in the xpaths
    for item in range (1, 21):

        # Switch to search results page
        driver.switch_to.window(driver.window_handles[1])

        # Get element to click on, to see each individual page with content
        # Last page will have fewer than 20 elements, so need to handle this exception
        try:
            elem = driver.find_element_by_xpath('//*[@id="searchResults"]/table/tbody[{}]/tr[1]/td[2]/a'.format(item))
            elem.click()
        except:
            break

        # Switch to page with content and get URL name
        driver.switch_to.window(driver.window_handles[2])
        item_key = driver.current_url.split('/')[-1]
        item_key = item_key.replace('?', '_') # Replace ? because it would be an invalid filename

        # Append result to dictionary for later processing
        res_dict[item_key] = driver.page_source

        # Write out each page source as a file
        with open(item_key + '.txt', encoding = 'utf-8', mode = 'w+') as file:
            file.write(driver.page_source)

        # Close tab
        driver.close()

    # Switch back to search results tab
    driver.switch_to.window(driver.window_handles[1])

    # Click on next page once 20 results have been saved
    # Next page button changes after first 20 results are shown, hence need to enclose the xpath in a try block
    try:
        next_page = driver.find_element_by_xpath('//*[@id="searchResults"]/div[3]/section/ul/li[3]/a/em')
    except:
        next_page = driver.find_element_by_xpath('//*[@id="searchResults"]/div[3]/section/ul/li[1]/a/em')
    next_page.click()

    # Sleep momentarily because next page takes a while to load
    time.sleep(2)

# Check that all results are stored
assert len(res_dict.keys()) == int(res[-1]), "It looks like not all the results were stored!"
```