DOCUMENTATION FOR ZENDESK TAMPERMONKEY MACROS FOR CSW T2

This is the explanation of the code and inner workings of the entire system, long overview includes all code. Short includes just main functions. This code uses a password protected spreadsheet in Smartsheet as a database and uses local data available on the ticket, the below described code does not communicate outside of the website or script, it saves data in the browsers allocated memory and saved data outside of the short term (deleted when site is refreshed) does not contain customer data.

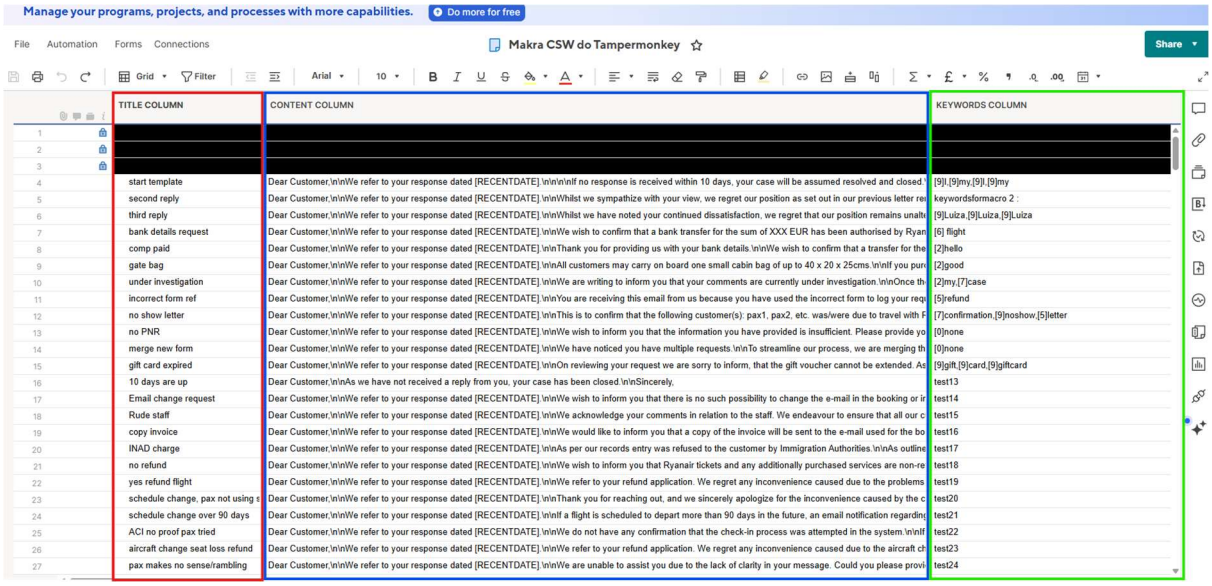
Usage

The main way to use the program is to upload and then use macros while responding to tickets, to use macros, these allow agents to solve certain specific repeatable cases in literal seconds:

Creation of the macro

Make one, replacing the numbered spaces, allocated for these macros

Point to point	Dear Customer,\n\nWe refer to your response dated [RECENTDATE].\n\nOur Terms and Conditions, agreed to at the time of booking, clearly state that we do not offer,
Pax sat next to fat passenger	Dear Customer,\n\nWe refer to your response dated [RECENTDATE].\n\nThank you for your email and for bringing this matter to our attention. We deeply regret that yo
Pax falsely claiming a stolen card	Dear Customer,\n\nWe refer to your response dated [RECENTDATE].\n\nSituations where a passenger makes a payment on behalf of another traveller on the same vo
Drunk pax denied boarding	Dear Customer,\n\nWe refer to your response dated [RECENTDATE].\n\nFollowing a thorough review of the matter you presented, we regret to inform you that no refun
Different email than booking	Dear Customer,\n\nWe refer to your response dated [RECENTDATE].\n\nWe wish to inform you that the e-mail address you have provided does not match the one in th
testingMacroDoNotUse	last convo date [RECENTDATE]\n\ncustomer name [CUSTOMER]\n\n pnr [PNR]\n\nflight number [FLIGHTNUMBER]\n\ndeparture airport [DEPARTURE] \n\n arrival air
ref made to inactive or expired ca	Dear Customer,\n\nWe refer to your response dated [RECENTDATE].\n\nPlease note that all refunds are returned to the original payment method used at the time of bc
53	53
54	54
55	55
56	56
57	57
58	58
59	59
60	60
61	61
62	62
63	63



Title

Should be only conventional letters, as the program filters junk info before making it readable and may filter out the title on accident.

Content

Dear Customer,\n\nWe refer to your response dated [RECENTDATE].\n\n\n\nIf no response is received within 10 days, your case will be assumed resolved and closed.\n\nSincerely,

Works, as **\n** is converted into an ENTER, **Customer** is converted to the display name of the customer or stays as Customer if no name can be recognised. This makes the above macro convert into

“Dear John,

We refer to your response dated 13th of December.

If no response is received within 10 days, your case will be assumed resolved and closed.

Sincerely, **“**

Please remember, this may not support every character, especially if it is less common

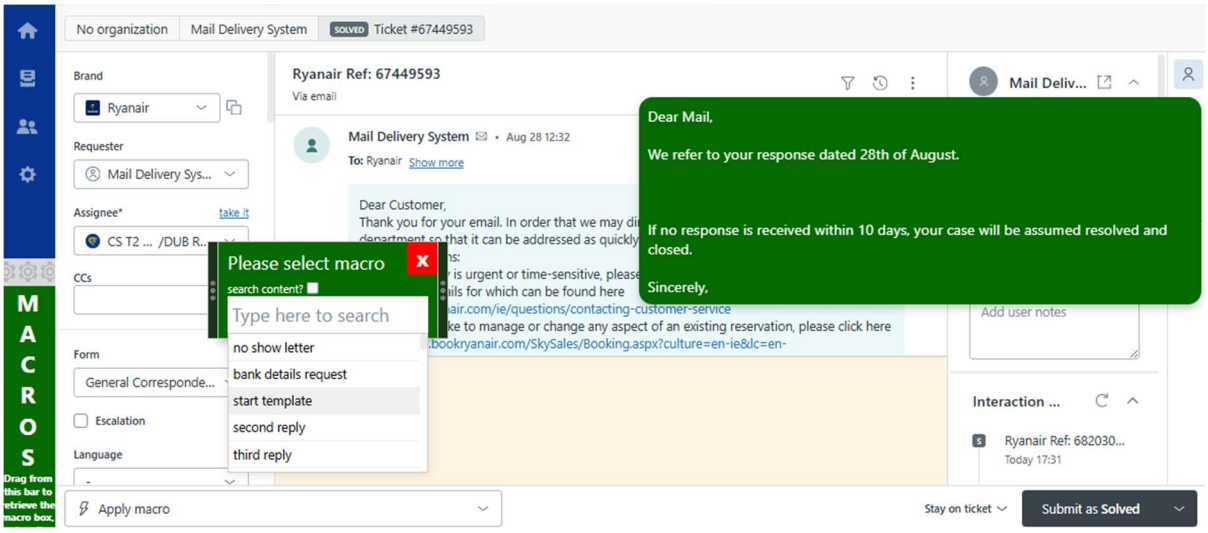
Keywords

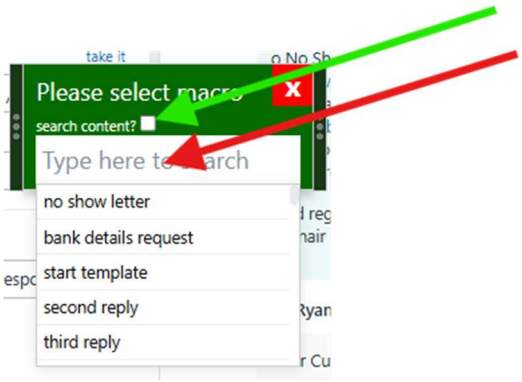
The keyword system is used to sort macros by relevancy, if the macro is about bags going missing, you may write **[5]bag,[9]missing** this way if the word bag or missing is said by the passenger the associated macro will be visible first. **[1]** word will be less important to this system than the **[9]** word.

Remember to refresh the program often, many anti tampering processes are working against this program and it may break, in such cases a refresh helps tremendously. This program was made on lunch breaks and is actively fighting with the sites functions to work and access data/not be deleted.

Using the macros

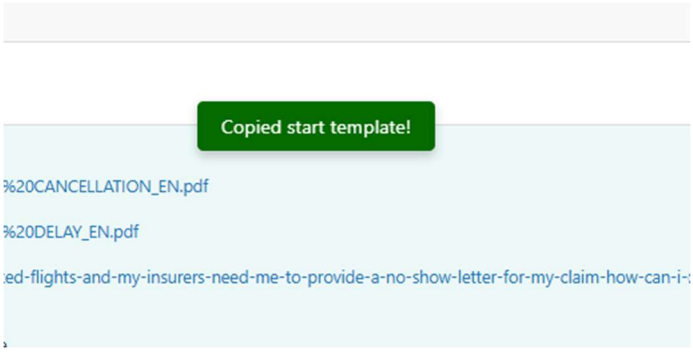
In order to use the macro you have created, enter the ticket and wait for the window to pop up, then hover over it to preview and check if its correct.



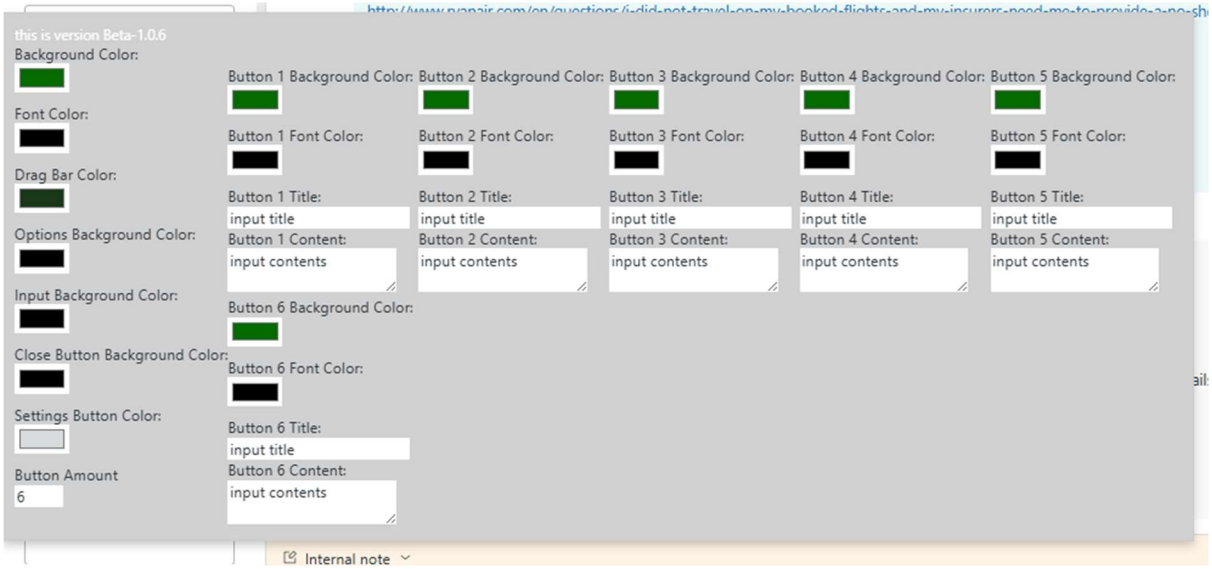


Type to search titles, if you wish to search the macros content, select that option instead

When the hovered option is the correct one, click it to copy into your clipboard and paste! That's it, the macro was used.



You will also get this confirmation just in case.



There are additional options the colors are changeable for almost every aspect of the created windows and elements.

Button amount refers to the buttons you can create, select the amount you would like, refresh the page, then input the colors of the button and font, displayed title, and the content for your own personalised buttons with whatever lines you seems to need on every second ticket, to have at the ready

Here are 3 example buttons, clicking them instantly copies whatever the agent inputted previously.

Short version

The short theoretical version will go over main functions and the larger actions the code performs.

This code runs in Zendesk tickets and a specific Smartsheet spreadsheet

First, it determines where it is

If it is in Smartsheet

It waits for it to load a file containing

home?formName=ajax&formAction=fa_loadSheet&ss_v=

It contains a list of every change that this sheet received, in it is a list of cell data within 20 pages of junk information. It saves that information and attempts to close the tab, if it was opened via code, it closes automatically.

If it is in Zendesk

It waits for it to load, and then reads the saved 20 pages of data and requests the opening of Smartsheet, which does the above section then closes, updating the macros. It also loads user preferences for colors and the buttons data, and displays it.

It filters data, and loads in all of the ticket data it needs, mainly the Name of the customer and the most recent Date of their conversation, then processes it.

When the macro is hovered over or clicked, it reads whether the macro contains '\n' , 'Customer', '[RECENTDATE]' and changes it to the appropriate text elements, if no Name is found, it does not change 'Customer'

After that the processed text is put into the clipboard, this is due to the Reacts input mutation protection which either switches the response field back, or as in the case of full site translation it **crashes** (this is due to the Zendesk's input field being modified back and forth by the translation and input protection fighting each other, the below code fixes that, although badly as it was just a one time experiment)

```
let singlefire = true;let testArray = [];repeat();function repeat(){if (testArray.length<1){testArray =document.querySelectorAll('[role="textbox"]');setTimeout(()=>{repeat()},1000)};else{testArray.forEach((element) => {element.setAttribute("translate", "no");if(singlefire){element.addEventListener("change", (event) => {repeat()});!singlefire}})};setTimeout(()=>{repeat()},10)};
```

If the buttons are clicked, their content is read from the browser data and put into the clipboard.

If the buttons are edited the code sends the data to the browser.

Upon reloading od the site anything but the smartsheet data and the user preferences is wiped.

Long overview

This section lays out the name version ect, tells the script where to run and what permissions it has, here it gets mainly memory permissions and opening tabs

```
// ==UserScript==
// @name         Zendesk custom macros by Grzegorz Ptaszynski merge attempt
// @namespace    http://tampermonkey.net/
// @version      Beta-1.0.6
// @description  macro helper to ease the pasting of templates
// @author       Grzegorz Ptaszynski
// @match        https://ryanairsupport.zendesk.com/agent/*
// @icon         
// @grant        GM_openInTab
// @grant        GM_deleteValue
// @grant        GM_getValue
// @grant        GM_addStyle
// @match        https://app.smartsheet.com/sheets/qG3Jjrg3fVPXmRQgP9rHx2X6CjQhWCPCH2XRPQ51?view=grid
// @grant        GM_log
// @grant        GM_setValue
// @grant        GM_listValues
// ==/UserScript==
```

This checks where the script is

```
if (
  window.location.href ==
  "https://app.smartsheet.com/sheets/qG3Jjrg3fVPXmRQgP9rHx2X6CjQhWCPCH2XRPQ51?view=grid"
) {
```

This runs on Smartsheet

```
  ; (function () {
    "use strict"
    //Failsafe
```

This is for one agent who had troubles with logging in and it would open and not close 20 smartsheet tabs. This closes it after 10 min of inactivity

```
    setTimeout(()=>{window.close()},600000)
    // Save original open method
    const originalOpen = XMLHttpRequest.prototype.open
    // Override the open method
    XMLHttpRequest.prototype.open = function (method, url, async, user, pass) {
      const self = this
      if (url.includes("home?formName=ajax&formAction=fa_loadSheet&ss_v=")) {
        this.addEventListener("load", function () {
          if (self.status === 200) {
```

This grabs the data and closes tab

```
            const fileContent = self.responseText // Get response as text (assuming it's a text file)
            GM_setValue("sheetData", fileContent)
            window.close()
          }
        })
      }
      // Call the original open method
      originalOpen.call(this, method, url, async, user, pass)
    }
  })()
} else {
  //C O N F I G
```

Main config of the cript,default colors and other minor stuff

```
  //change whatever you want in here, remember to save a working version first and make this a copy just in case

  //what works is replacing the part after the = sign with the color in quotations:
  //name of color ex. "blue";
  //hexcodes ex. "#056b00"
  //rgb ex. "rgb(155, 102, 102)"
  //save the code and reload zendesk to check if it works
```

```
  //the background color of messages and prompts(default is dark green "#056b00")
  const backgroundColor = GM_getValue('backgroundColor', "#056b00")

  //the color of the font of the messages and prompts(default is white "white")
  const fontColor = GM_getValue('fontColor', "white")

  //the color of the drag bars present at the sides of the prompt box (default is dark grey "#193818")
  const dragBarColor = GM_getValue('dragBarColor', "#193818")

  //the background color of the options in the list of macros(default is white "white")
  const optionsBackgroundColor = GM_getValue('optionsBackgroundColor', "white")

  //the background color of the search box(default is white "white")
  const inputBackgroundColor = GM_getValue('inputBackgroundColor', "white")

  //the background color of the close button(default is "red")
  const closeButtonBackgroundColor = GM_getValue('closeButtonBackgroundColor', "red")

  //the color of the drag bars present at the sides of the prompt box (default is grey "#D1D1D1")
  const settingsColor = "#D1D1D1"
  const settingsButtonColor = GM_getValue('settingsButtonColor', "#D8DCDE")
  //E N D  O F  C O N F I G

This glob of variables is because this was faster to make and I was on borrowed time, swinging in the dark

  let isDraggedOut = undefined
  let isDragging = false;
  let isMacroContainerPresent = false
  let currentTicketNr = window.location.href.toString().split("/").pop()
  let sheetData = undefined
  let titleArray = []
  let contentArray = []
  let keywordArray = []
  let articles = []
  let recentConvoDate = undefined
  let isPromptBoxActive = false
  let boxCount = 0
  let debugCount = 0
  let macroArray = []
  let lastX, lastY
  let offsetX, offsetY;
  let isMenuOpen = false
  let customerName = 'Customer'
  let test = undefined
  function DataInsertStart() {
    titleArray = []
    contentArray = []
    keywordArray = []
This is so that less tabs open constantly
    if (GM_getValue('recentOpenTime', 1) <= Date.now() - 60000) {
      GM_openInTab(
This opens the smartsheet tab
        "https://app.smartsheet.com/sheets/qG3Jjrg3fVPXmRQgP9rHx2X6CjQhWCPCH2XRPQ51?view=grid",
        { loadInBackground: true, insert: true },
      )
      GM_setValue('recentOpenTime', Date.now())
    }
    setTimeout(() => {
Timeout prevents breaking, this filters the file to only get the relevant data
      sheetData = GM_getValue("sheetData", undefined)
      const matches = Array.from(
        sheetData.matchAll(
Filtering regex
          /(?<=\')(?!\,)[A-Z, .0-9\\%£$€\[\];:*"@\-+=|_\(\)x&~#?!\{\}\`´/]+(?=\')/gi,
        ),
      )
      const words = matches.map((match) => match[0])
      const filteredWords = words.filter(
        (x) => x != "DO NOT CHANGE column titles or this column",
      )
      filteredWords.pop()
      filteredWords.splice(0, 4)
      for (let i = 0; i < filteredWords.length; i++) {
        switch (filteredWords[i]) {
          case "TITLE COLUMN":
```

```
                titleArray.push(filteredWords[i + 1])
                break;
              case "CONTENT COLUMN":
                contentArray.push(filteredWords[i + 1])
                break;
              case "KEYWORDS COLUMN":
                keywordArray.push(filteredWords[i + 1])
                break;
            }
          }

        }, 1000)//timeout needed, breaks otherwise
      }

      DataInsertStart()
      GM_addStyle(`
          #custom-prompt-input {
            width: 300px;
            padding: 8px;
            font-size: 16px;
          }
        `)
      function returnArticleData() {
This grabs the articles(mail message elements on tickets)
        let cutNumber = 0
        const articleData = Array.from(articles)
          .map((article) => {
            const innerHTML = Array.from(article.querySelectorAll("span"))
            let parent = article.closest("div")
            let targetDiv = parent.querySelector(
.this checks whether the grabbed thing is from this ticket
              `div[elementtiming="omnilog/${currentTicketNr}"]`,
            )
            const paragraphs = article.querySelectorAll(".zd-comment") //:not(blockquote):not(tr)
            const elements = Array.from(
.checks whether it's the customers message
              document.querySelectorAll('[data-test-id="tooltip-requester-name"]'),
            )
            const requesterCheck = elements.some((element) => {
              return element.textContent === innerHTML[0].textContent
            })
            if (
.final full check, otherwise it loads all active tabs
              article.querySelector('div[type="end-user"]') !== null &&
              requesterCheck &&
              Boolean(targetDiv)
            ) {
              return Array.from(paragraphs).map((p) => p.textContent.trim())
            } else {
              return " "
            }
          })
.this below deletes random elements that get tagged on and would disrupt the keyword system
          .map((element) => {
            if (Array.isArray(element)) {
              return element.join(" ")
            } else {
              return element
            }
          })
          .map((element) => {
            return cutNs(
              element
                .replaceAll(
                  "The content of this e-mail or any file or attachment transmitted with it may have been changed or altered without the consent of the author. If you are not the intended recipient of this e-mail, you are hereby notified that any review, dissemination, disclosure, alteration, printing, circulation or transmission of, or any action taken or omitted in reliance on this e-mail or any file or attachment transmitted with it is prohibited and may be unlawful. If you have received this e-mail in error please notify Ryanair Holdings plc by contacting Ryanair Holdings plc (Company No. 249885) / Ryanair DAC. (Company No. 104547). Registered in the Republic Of Ireland. Airside Business Park, Swords, Co Dublin, Ireland.",
                  "",
                )
                .replaceAll("EXTERNAL EMAIL:", "")
                .replaceAll(
                  "This email originated from outside of the Organisation. Do not click links or open attachments unless you recognise the sender and know the content is safe.",
```

```
                    "",
                )
                .replaceAll(
                    "EXTERNAL EMAIL:\n\t\t\n\t\n\tThis email originated from outside of the Organisation. Do not click links or open attachments unless you recognise the sender and know the content is safe.",
                    "",
                )
                .replaceAll(/[\\.,/|\-():0-9"']/g, ''),
            )
                .split("\t")
                .join(" ")
        })
        .join(" ")
        .split(" ")
    function cutNs(inputString) {

        // Split the string by '\n' into an array
        let index = 0
        let parts = inputString.split("\n")
        parts.forEach((element) => {
            if (element === "") { cutNumber++ } else { cutNumber = 0 }
            index++
            if (cutNumber >= 4) { parts = parts.slice(0, index); return }
        })
        return parts.join(" ")
    }
    const articleDataNoEmpty = articleData.filter((element) => element !== "")
    return articleDataNoEmpty
}
async function getResult() {
    try {
        const result = await articleGrabber() // Wait for the result from articleGrabber
        return result // Return the result when it's ready
    } catch (error) {
        console.error(error) // Handle any errors (though in this case, it's unlikely)
    }
}
.this puts the settings menu into the site, this sort of setup is bad, but is the only working way to do it as not much worked otherwise
function handleSettingsMenu() {
    const settingsWindow = document.createElement("div");

    if (!isMenuOpen) {
        isMenuOpen = true
        settingsWindow.id = "settings";
        settingsWindow.style.position = "fixed";
        settingsWindow.style.top = '-200px'; // Adjusted for visibility
        settingsWindow.style.left = '60px';
        settingsWindow.style.backgroundColor = settingsColor;
        settingsWindow.style.padding = "10px";
        settingsWindow.style.boxShadow = "0 4px 8px rgba(0, 0, 0, 0.2)";
        settingsWindow.style.zIndex = "9999";
        settingsWindow.style.width = "50vw"; // Adjust if necessary
        settingsWindow.style.maxHeight = "95vh"; // Max height is 80% of the viewport height
        settingsWindow.style.overflowY = "auto"; // Enable scrolling if the content exceeds max-height
        settingsWindow.style.border = "1px solid #ccc";
        settingsWindow.style.position = "relative";
        settingsWindow.style.color = fontColor; // Apply dynamic font color
        const versionTag = document.createElement('p')
        versionTag.innerText = `this is version ${GM_info.script.version}`
        settingsWindow.appendChild(versionTag)

        // Create the color input row for color settings
        const colorSettings = document.createElement("div");
        colorSettings.style.marginBottom = "20px"; // Space between rows
        colorSettings.style.float = 'left'

        function createColorSetting(labelText, settingName, defaultValue) {
            const settingDiv = document.createElement("div");
            settingDiv.style.marginBottom = "10px";

            const label = document.createElement("label");
            label.textContent = labelText;

            const input = document.createElement("input");
            input.type = "color";
```

```
            input.value = defaultValue; // Default color value

            input.addEventListener("input", (e) => {
                GM_setValue(settingName, e.target.value); // Save the new value
            });

            settingDiv.appendChild(label);
            settingDiv.appendChild(input);
            return settingDiv;
        }
        let inputDiv = document.createElement('div')
        let inputTitle = document.createElement('label')
        inputTitle.innerText = 'Button Amount'
        let buttonAmountInput = document.createElement('input')
        buttonAmountInput.type = 'number'
        buttonAmountInput.min = '0'
        buttonAmountInput.max = '100'
        buttonAmountInput.value = GM_getValue('buttonAmount', 6)
        buttonAmountInput.id = 'buttonAmountInput'
        buttonAmountInput.style.width = '40px'
        buttonAmountInput.addEventListener("input", (e) => { GM_setValue('buttonAmount', buttonAmountInput.value) });
        inputDiv.appendChild(inputTitle)
        inputDiv.appendChild(buttonAmountInput)
        // Color settings for background, font, drag bars, etc.
        colorSettings.appendChild(createColorSetting("Background Color:", "backgroundColor", GM_getValue('backgroundColor', "#056b00")));
        colorSettings.appendChild(createColorSetting("Font Color:", "fontColor", GM_getValue('fontColor', "white")));
        colorSettings.appendChild(createColorSetting("Drag Bar Color:", "dragBarColor", GM_getValue('dragBarColor', "#193818")));
        colorSettings.appendChild(createColorSetting("Options Background Color:", "optionsBackgroundColor", GM_getValue('optionsBackgroundColor',
"white")));
        colorSettings.appendChild(createColorSetting("Input Background Color:", "inputBackgroundColor", GM_getValue('inputBackgroundColor', "white")));
        colorSettings.appendChild(createColorSetting("Close Button Background Color:", "closeButtonBackgroundColor",
GM_getValue('closeButtonBackgroundColor', "red")));
        colorSettings.appendChild(createColorSetting('Settings Button Color:', 'settingsButtonColor', GM_getValue('settingsButtonColor', "#D8DCDE")))
        colorSettings.appendChild(inputDiv);
        settingsWindow.appendChild(colorSettings);

        // Create the title-input and content-input rows for each button
        const buttonSettings = document.createElement("div");
        buttonSettings.style.marginTop = "20px"; // Space before the title-content inputs
        let buttonAmount = GM_getValue('buttonAmount', 6)
        for (let i = 1; i <= buttonAmount; i++) {
            buttonSettings.appendChild(createTitleContentPair(i, GM_getValue(`button${i}Title`, "input title"), GM_getValue(`button${i}Content`, "input
contents")));
        }
        function createTitleContentPair(index, title, content) {

            const pairContainer = document.createElement("div");
            pairContainer.id = `pairContainer${index}`
            pairContainer.style.marginBottom = "5px";
            pairContainer.style.marginRight = '5px'
            pairContainer.style.float = 'left'
            pairContainer.classList.add('button-pair'); // Add class for selecting later
            pairContainer.appendChild(createColorSetting(`Button ${index} Background Color:`, `button${index}Color`, GM_getValue(`button${index}Color`,
"#056b00")));
            pairContainer.appendChild(createColorSetting(`Button ${index} Font Color:`, `button${index}fontColor`, GM_getValue(`button${index}fontColor`,
"white")));

            const titleLabel = document.createElement("label");
            titleLabel.textContent = `Button ${index} Title: `;
            const titleInput = document.createElement("input");
            titleInput.type = "text";
            titleInput.value = title;
            titleInput.classList.add('button-title'); // Add class for selecting later

            const contentLabel = document.createElement("label");
            contentLabel.textContent = `Button ${index} Content: `;
            const contentInput = document.createElement("textarea");
            contentInput.value = content;
            contentInput.classList.add('button-content'); // Add class for selecting later

            titleInput.addEventListener("input", (e) => {
                GM_setValue(`button${index}Title`, e.target.value); // Save the new title
            });

            contentInput.addEventListener("input", (e) => {
```

```
            GM_setValue(`button${index}Content`, e.target.value); // Save the new content
          });
          pairContainer.appendChild(titleLabel);
          pairContainer.appendChild(titleInput);
          pairContainer.appendChild(contentLabel);
          pairContainer.appendChild(contentInput);
          return pairContainer;
        }


        // Button titles and contents for button1 to button6
        settingsWindow.appendChild(buttonSettings);

        // Append the settings window to the DOM
        document.querySelector('[data-test-id="support_nav"]').appendChild(settingsWindow);

      }
      else { document.getElementById("settings").remove(); isMenuOpen = false }
    }
.this handles the showing of the macros window and the sidebar
    function showDatalistPrompt(message, options) {
      const promptStartX = GM_getValue("promptX", false)
      const promptStartY = GM_getValue("promptY", false)
      const buttonList = GM_getValue("buttonList", undefined)
      macroArray.sort((a, b) => b.relevancePoints - a.relevancePoints)
      let displayList = options
      isPromptBoxActive = true;
      const promptContainer = document.createElement("div");
      promptContainer.id = "macro-prompt";
      promptContainer.style.position = "fixed";
      promptContainer.style.top = "5px";
      if (promptStartY) { promptContainer.style.top = promptStartY }
      promptContainer.style.left = "450px"//"83%";
      if (promptStartX) { promptContainer.style.left = promptStartX }
      //if (promptContainer.style.top == '-500px' && promptContainer.style.left == '-500px') { console.log('amogus2') }
      promptContainer.style.transform = "translateX(-50%)";
      promptContainer.style.backgroundColor = backgroundColor;
      promptContainer.style.padding = "5px 10px";
      promptContainer.style.boxShadow = "0 4px 8px rgba(0, 0, 0, 0.2)";
      promptContainer.style.zIndex = "9999";
      promptContainer.style.color = fontColor;
      promptContainer.style.width = "220px";
      promptContainer.style.border = "1px solid #ccc";
      promptContainer.style.position = "relative";
      const sidebar = document.createElement("div");
      const settingsButton = document.createElement("button")
      const userButtonsContainer = document.createElement("div")
      if (!isMacroContainerPresent) {
        isMacroContainerPresent = true
        sidebar.id = 'sidebar'
        sidebar.textContent = "M A C R O S";
        const smallTextContent = document.createElement('div')
        smallTextContent.textContent = 'Drag from this bar to retrieve the macro box, drag it back here to hide it again'
        smallTextContent.style.position = 'fixed'
        smallTextContent.style.top = '558px'
        smallTextContent.style.fontSize = '11px'
        sidebar.appendChild(smallTextContent)
        //sidebar.style.float = 'left'
        sidebar.style.padding = "2px 10px";
        sidebar.style.fontSize = "26px";
        sidebar.style.backgroundColor = backgroundColor;
        sidebar.style.border = "none";
        sidebar.style.color = fontColor;
        sidebar.style.fontWeight = "bold";
        sidebar.style.cursor = "pointer";
        sidebar.style.position = "fixed";
        sidebar.style.top = "355px";
        sidebar.style.width = "40px";
        sidebar.style.height = '310px'
        sidebar.style.display = 'flex';
        sidebar.style.justifyContent = 'center';
        sidebar.style.textAlign = 'center';
        sidebar.style.zIndex = "999";

        sidebar.addEventListener("mousedown", (e) => {
```

```
            const promptBox = document.getElementById('macro-prompt')
            lastX = e.clientX
            lastY = e.clientY
            promptBox.style.top = `${e.clientY}px`
            promptBox.style.left = `${e.clientX}px`
            isDragging = true;
            document.body.style.userSelect = "none"; // Prevent text selection while dragging
        });
        sidebar.addEventListener("mouseup", (e) => {
            isDragging = false
        });
        document.querySelector('[data-test-id="support_nav"]').appendChild(sidebar)
    }
    settingsButton.style.backgroundImage = 'url(https://raw.githubusercontent.com/EEEGuba/Ryanair-Zendesk-Tampermonkey-
addons/refs/heads/main/Custom%20synced%20macros/settings%20button%20icon.png)';
    settingsButton.style.backgroundSize = 'cover';
    settingsButton.style.backgroundRepeat = 'repeat-x';
    settingsButton.style.backgroundSize = '40%'
    settingsButton.style.backgroundPosition = 'center';
    settingsButton.style.position = 'fixed'
    settingsButton.style.top = '325px'
    settingsButton.style.height = '30px'
    settingsButton.style.width = '60px'
    settingsButton.style.zIndex = "999";
    settingsButton.style.backgroundColor = settingsButtonColor
    settingsButton.addEventListener("mousedown", (e) => {
        e.stopPropagation(); // Prevent this event from bubbling up to the sidebar
        handleSettingsMenu();
    })
    userButtonsContainer.style.backgroundColor = "rgba(255, 0, 0, 0)"
    userButtonsContainer.style.position = 'fixed'
    userButtonsContainer.style.top = '670px'
    userButtonsContainer.style.left = '-1px'
    userButtonsContainer.style.height = `${window.innerHeight - 665}px`
    userButtonsContainer.style.width = '80px'
    userButtonsContainer.style.overflowY = 'auto'
    userButtonsContainer.style.overflowX = 'hidden'
    userButtonsContainer.style.zIndex = "999";
    sidebar.appendChild(settingsButton)
    //document.appendChild(userButtonsContainer)
    document.querySelector('[data-test-id="support_nav"]').appendChild(userButtonsContainer)
    userButtonsContainer.addEventListener('click', (e) => {
        // this is here to prevent click event from bubbling up to sidebar

    });
    class UserButton {
        constructor(title, content, color, fontColor) {
            this.button = document.createElement("button");
            this.button.textContent = title;
            this.button.style.backgroundColor = color;
            this.button.style.color = fontColor;
            this.button.style.textAllign = 'center'
            this.content = content;
            this.button.style.float = 'left'
            this.button.style.border = "2px solid #000";
            this.button.style.fontSize = '12px'
            this.button.style.lineHeight = "1";
            this.button.style.marginTop = '2px'; // Adds 2px space below each button
            this.button.style.maxWidth = '60px'
            this.button.style.width = '100%'
            this.button.style.maxHeight = '160px'
            this.button.style.minHeight = '40px'
        }
.This is bad, but didn't work in any better way I tried
        // Method to set an arbitrary click handler
        setClickHandler(callback) {
            this.button.addEventListener("click", (e) => {
                // Prevent the event from bubbling up to the parent
                callback(); // Call the passed-in callback function
            });
        }
    }

    // Simulate the GM_setValue data (assuming you have it elsewhere in code)
    //GM_setValue("buttonList", [{ title: "test1", content: "this is test1, you may not like it but I don't either", color: "pink", fontColor: "green" }]);
```

```
    function createButtons() {
      for (let i = 1; i <= GM_getValue('buttonAmount', 0); i++) {
        const userBtn = new UserButton(GM_getValue(`button${i}Title`, 'input title'), GM_getValue(`button${i}Content`, "placeholder contents"),
GM_getValue(`button${i}Color`, 'green'), GM_getValue(`button${i}fontColor`, 'white'));
        // Define an arbitrary function to be called when the button is clicked
        const clickFunction = () => {
          const originalBackgroundColor = userBtn.button.style.backgroundColor
          const originalFontColor = userBtn.button.style.color
          setTimeout(() => {
            userBtn.button.style.border = "2px solid #000000";
            //userBtn.button.style.backgroundColor = originalBackgroundColor
            userBtn.button.style.color = originalFontColor
          }, 100);
          userBtn.button.style.border = "2px solid #ffffff";
          //userBtn.button.style.backgroundColor = 'black'
          userBtn.button.style.color = 'white'
          navigator.clipboard.writeText(userBtn.content)
          createMessageBox('Copied button\n' + userBtn.button.textContent.toString() + '\n to clipboard', 2000)
        };


        // Set the click handler to the arbitrary function
        userBtn.setClickHandler(clickFunction);

        userButtonsContainer.appendChild(userBtn.button); // Append the button element to the container
      };
    }

    createButtons();

    // Create the left bar
    const leftBar = document.createElement("div");
    leftBar.style.cursor = "move"
    leftBar.style.width = "10px";
    leftBar.style.backgroundColor = dragBarColor;
    leftBar.style.border = "1px solid #ccc";
    leftBar.style.height = "104px";
    leftBar.style.position = "absolute";
    leftBar.style.left = "-12px";
    leftBar.style.top = "-1px";
    leftBar.style.backgroundImage = 'url(https://raw.githubusercontent.com/EEEGuba/Ryanair-Zendesk-Tampermonkey-
addons/refs/heads/main/Custom%20synced%20macros/three%20dots.png)';
    leftBar.style.backgroundSize = 'cover';
    leftBar.style.backgroundRepeat = 'no-repeat';
    leftBar.style.backgroundSize = '300%'
    leftBar.style.backgroundPosition = 'center';

    leftBar.addEventListener("mousedown", (e) => {
      isDragging = true;
      lastX = e.clientX
      lastY = e.clientY
      document.body.style.userSelect = "none"; // Prevent text selection while dragging
    });

    // Add the left bar to the promptContainer
    promptContainer.appendChild(leftBar);
    const rightBar = document.createElement("div");
    rightBar.style.cursor = "move"
    rightBar.style.width = "10px";
    rightBar.style.backgroundColor = dragBarColor;
    rightBar.style.border = "1px solid #ccc";
    rightBar.style.height = "104px";
    rightBar.style.position = "absolute";
    rightBar.style.right = "-12px";
    rightBar.style.top = "-1px";
    rightBar.style.backgroundImage = 'url(https://raw.githubusercontent.com/EEEGuba/Ryanair-Zendesk-Tampermonkey-
addons/refs/heads/main/Custom%20synced%20macros/three%20dots.png)';
    rightBar.style.backgroundSize = 'cover';
    rightBar.style.backgroundRepeat = 'no-repeat';
    rightBar.style.backgroundSize = '300%'
    rightBar.style.backgroundPosition = 'center';

    rightBar.addEventListener("mousedown", (e) => {
      lastX = e.clientX
```

```
    lastY = e.clientY
    isDragging = true;
    document.body.style.userSelect = "none"; // Prevent text selection while dragging
});

rightBar.addEventListener("mouseup", () => {
    GM_setValue("promptX", promptContainer.style.left)
    GM_setValue("promptY", promptContainer.style.top)
})
leftBar.addEventListener("mouseup", () => {
    GM_setValue("promptX", promptContainer.style.left)
    GM_setValue("promptY", promptContainer.style.top)
})
// Add the left bar to the promptContainer
promptContainer.appendChild(rightBar);

// Create the header container (text + close button)
const headerContainer = document.createElement("div");
headerContainer.style.display = "flex";
headerContainer.style.justifyContent = "space-between";
headerContainer.style.alignItems = "center";
headerContainer.style.whiteSpace = "nowrap"; // Prevent the text from wrapping

// Create a prompt message element
const messageElement = document.createElement("p");
messageElement.textContent = message;
messageElement.style.margin = "0"; // Remove default margin to avoid spacing issues
messageElement.style.flexGrow = "1"; // Allow the message to take up available space
messageElement.style.fontSize = "20px"
headerContainer.appendChild(messageElement);

// Create the "x" button (close button) and position it in the top right
const closeButton = document.createElement("div");
closeButton.textContent = "x";
closeButton.style.padding = "2px 10px";
closeButton.style.fontSize = "24px";
closeButton.style.backgroundColor = closeButtonBackgroundColor
closeButton.style.border = "none";
closeButton.style.color = fontColor;
closeButton.style.fontWeight = "bold";
closeButton.style.cursor = "pointer";

closeButton.style.position = "relative";
closeButton.style.top = "-5px";
closeButton.style.left = "10px";
closeButton.onclick = function () {
    createMessageBox("Copying nothing, like you wanted!", 3000)
        ; promptContainer.style.left = '-500px'; promptContainer.style.top = '-500px'; setTimeout(() => {
            GM_setValue("promptX", promptContainer.style.left)
            GM_setValue("promptY", promptContainer.style.top)
        }, 100);
};
headerContainer.appendChild(closeButton);

promptContainer.appendChild(headerContainer); // Add the header to the prompt container

// Create a datalist
const datalist = document.createElement("datalist");
datalist.id = "prompt-datalist";

// Create a container to simulate the datalist
const datalistContainer = document.createElement("div");
datalistContainer.style.maxHeight = "150px";
datalistContainer.style.overflowY = "auto";
datalistContainer.style.backgroundColor = optionsBackgroundColor;
datalistContainer.style.border = "1px solid #ccc";
datalistContainer.style.position = "absolute";
datalistContainer.style.top = "99px";
datalistContainer.style.width = "218px";
datalistContainer.style.zIndex = "9999";
//datalistContainer.style.borderRadius = "4px";
datalistContainer.style.boxShadow = "0 2px 10px rgba(0, 0, 0, 0.1)";

// Add options to the datalist
displayList.forEach((option) => {
```

```
    if (option.title != "CONTENT COLUMN" && option.title != "KEYWORDS COLUMN" && option.title != "TITLE COLUMN") {
        const optionElement = document.createElement("div");
        optionElement.value = option.title;
        optionElement.innerText = option.title;
        optionElement.style.color = "black";
        optionElement.style.padding = "5px";
        optionElement.style.cursor = "pointer";
        optionElement.style.borderBottom = "2px solid #f0f0f0";
        optionElement.id = `option${option.index}`
        optionElement.onmouseover = () => {
            optionElement.style.backgroundColor = "#f0f0f0"; // Optional: highlight on hover
            const messageBox = document.createElement("div");
            messageBox.id = options.indexOf(option);
            messageBox.style.position = "fixed";
            messageBox.style.top = "150px";
            messageBox.style.right = "10px";
            if (window.event.clientX > window.innerWidth / 2) { messageBox.style.left = "10px"; }
            messageBox.style.backgroundColor = backgroundColor;
            messageBox.style.color = fontColor;
            messageBox.style.padding = "10px 10px";
            messageBox.style.borderRadius = "15px";
            messageBox.style.fontSize = "16px";
            messageBox.style.boxShadow = "0 4px 8px rgba(0, 0, 0, 0.2)";
            messageBox.style.zIndex = "9999";
            messageBox.style.display = "none";
            messageBox.style.maxWidth = '45%';
            document.body.appendChild(messageBox);
            decode(option.content).then((result) => {
                messageBox.innerText = result;
                messageBox.style.display = "block";
            });
        };

        optionElement.onmouseleave = () => {
            document.getElementById(options.indexOf(option)).remove();
        };

        optionElement.onmouseout = () => {
            optionElement.style.backgroundColor = ""; // Reset highlight
        };

        optionElement.onmousedown = () => {
            document.getElementById(options.indexOf(option)).remove();
            select(option, false);
        };

        datalistContainer.appendChild(optionElement);
        promptContainer.appendChild(datalistContainer);
    }
});
document.body.appendChild(datalist);
const checkboxDiv = document.createElement("div")
const checkboxText = document.createElement("small")
checkboxText.innerText = "search content?   "
const searchContentCheckbox = document.createElement("input")
searchContentCheckbox.setAttribute("type", "checkbox");
checkboxDiv.style.position = "relative";
checkboxDiv.style.top = "0px";
checkboxDiv.style.left = "0px";
checkboxDiv.style.width = "200px"
searchContentCheckbox.addEventListener('input', search)
// Create an input field linked to the datalist
const inputElement = document.createElement("input");
inputElement.setAttribute("list", "prompt-datalist");
inputElement.setAttribute("placeholder", "Type here to search");
inputElement.style.backgroundColor = inputBackgroundColor
inputElement.style.fontSize = "20px"; // Adjust text size
inputElement.style.padding = "5px"; // Add padding
inputElement.style.height = "30px"; // Set height
inputElement.style.width = "210px";
checkboxDiv.appendChild(checkboxText)
checkboxDiv.appendChild(searchContentCheckbox)
promptContainer.appendChild(checkboxDiv)
inputElement.addEventListener('input', search)
```

```
    function search() {
      const searchTerm = inputElement.value.toLowerCase();
      const searchContent = searchContentCheckbox.checked
      displayList.forEach(item => {
        let itemText = undefined;
        if (searchContent) { itemText = item.content.toLowerCase() } else { itemText = item.title.toLowerCase() }
        if (itemText.includes(searchTerm)) {
          document.getElementById(`option${item.index}`).style.display = 'block';
        } else {
          document.getElementById(`option${item.index}`).style.display = 'none';
        }
      })
    }
    promptContainer.appendChild(inputElement);

    document.body.appendChild(promptContainer);
.this handles what happens when a macro is picked and clicked on, puts away the macro box and puts the macro into the clipboard, or returns changed macro for the preview
    function select(option, returnNotCopy = false) {
      promptContainer.style.left = '-500px'; promptContainer.style.top = '-500px'
      let resultOfFunction = undefined
      const userInput = option.title
        .replace(/%5cn/g, "\n")
        .replace(/\[RECENTDATE\]/g, recentConvoDate.toString())
      decode(option.content)
        .then((result) => {
          if (returnNotCopy) { result = resultOfFunction; return }
          navigator.clipboard
            .writeText(result)
            .catch((err) => {
              console.error("Error copying to clipboard: ", err) // Error handling
            })
        })
        .catch((error) => {
          console.error("Error decoding string: ", error) // Error handling for decode
        })
      if (returnNotCopy) { return resultOfFunction }
      createMessageBox(`Copied ${userInput}!`, 5000)
      //    document.body.removeChild(promptContainer)
      //     document.body.removeChild(datalist)
    }
  }
.this checks if you can write where you clicked
  function isContentEditable() {
    // Get the currently focused element
    const activeElement = document.activeElement
    // Check if the active element is a div and if it has the contenteditable attribute set to true
    if (activeElement && activeElement.tagName === "DIV") {
      return activeElement.getAttribute("contenteditable") === "true"
    }
    return false
  }
  function contenteditableCheck() {
    if (isContentEditable()) {
      //onEditableClick()
      boxCount = 0
      refreshBox()
    }
  }
This handles the refreshing when the user switches tabs or tickets, puts red on the box until the macros work again
  function refreshBox() {
    setTimeout(() => {
      if (recentConvoDate == undefined || recentConvoDate == 12345) {
        document.getElementById("macro-prompt").style.backgroundColor =
          "#C72222"
      } else {
        document.getElementById("macro-prompt").style.backgroundColor =
          backgroundColor
      }
      boxCount++
      if (boxCount < 50) {
        refreshBox()
      }
    }, 100)
```

```
  }
.this is a just in case thing
  function refreshBoxFallback() {
    setTimeout(() => {
      if (recentConvoDate == undefined || recentConvoDate == 12345) {
        document.getElementById("macro-prompt").style.backgroundColor =
          "#C72222"
      } else {
        document.getElementById("macro-prompt").style.backgroundColor =
          backgroundColor
      }
      refreshBoxFallback()
    }, 500)
  }
.this makes a message box, I know, shocking
  function createMessageBox(message, time) {
    const messageBox = document.createElement("div")
    messageBox.innerHTML = message
    messageBox.style.position = "fixed"
    messageBox.style.top = "150px"
    messageBox.style.left = "50%"
    messageBox.style.transform = "translateX(-50%)"
    messageBox.style.backgroundColor = backgroundColor
    messageBox.style.color = fontColor
    messageBox.style.padding = "10px 20px"
    messageBox.style.borderRadius = "5px"
    messageBox.style.fontSize = "16px"
    messageBox.style.boxShadow = "0 4px 8px rgba(0, 0, 0, 0.2)"
    messageBox.style.zIndex = "9999"
    messageBox.style.display = "none"

    document.body.appendChild(messageBox)

    setTimeout(() => {
      messageBox.style.display = "block"
    }, 100)
    setTimeout(() => {
      messageBox.remove()
    }, time)
  }
  document.addEventListener("mouseup", contenteditableCheck)
  // Function to extract the recent conversation date from the articles
.this is mainly to grab the recent date of the customers conversation to autofill
  function articleGrabber() {
    return new Promise((resolve, reject) => {
      function checkArticles() {
        if (articles.length < 1) {
          articles = document.querySelectorAll("article")
          setTimeout(checkArticles, 200)
        } else {
          const date = recentConversationDate()
          recentConvoDate = date // Set the date
        }
      }

      checkArticles() // Start checking for articles
    })
  }
.this actually selects that date
  // Function to extract the recent date
  function recentConversationDate() {
    if (
      document.querySelector('[data-test-id="tooltip-requester-name"]') == null
    ) { //can probably make it active with no requester by editing this, might crash tho
      return undefined
    }
    const articlesWithEndUserType = Array.from(articles).filter((article) => {
      const innerHTML = Array.from(article.querySelectorAll("span"))
      // Find the closest parent element that contains this article
      let parent = article.closest("div")

      // Check if the parent contains a <div> with the desired elementtiming attribute
      let targetDiv = parent.querySelector(
        `div[elementtiming="omnilog/${currentTicketNr}"]`,
      )
```

```
        // If a target div is found within the same parent as the article
        const elements = Array.from(
          document.querySelectorAll('[data-test-id="tooltip-requester-name"]'),
        )
        const requesterCheck = elements.some((element) => {
          return element.textContent === innerHTML[0].textContent
        })
        return (
          article.querySelector('div[type="end-user"]') !== null &&
          requesterCheck &&
          Boolean(targetDiv)
        )
      })
      let nameCheckVariable = 'Customer'
      const dates = articlesWithEndUserType.map((article) => {
        nameCheckVariable = article.querySelector('[data-garden-id="typography.font"]').innerText
        const timeElement = article.querySelector("time")
        return timeElement ? timeElement.getAttribute("datetime") : null
      })
.this grabs the customers name to autofill
      const custNameArray = nameCheckVariable.split(' ')
      if (custNameArray.length>1){
      let treatedName = custNameArray[0].toLowerCase()
        treatedName = String(treatedName[0]).toUpperCase() + String(treatedName).slice(1);
        //treatedName[0].toUpperCase()
        customerName = treatedName
        }
      else{customerName='Customer'}
      if (dates.length === 0) {
        return undefined // Return undef if no date
      }
      const months = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December']
      const recentDate = dates[dates.length - 1]
      const dateArray = recentDate ? recentDate.slice(0, 10).split("-", 3).reverse(): undefined
      const correctTimeFormat = `${dateArray[0]}${getOrdinal(parseInt(dateArray[0]))} of ${months[parseInt(dateArray[1])-1]}`
.old date format, changed in July or so
    //OLD FORMAT //recentDate ? recentDate.slice(0, 10).split("-", 3).reverse().join("/"): undefined
      return correctTimeFormat
    }
    function getOrdinal(n) {
  let ord = 'th';

  if (n % 10 == 1 && n % 100 != 11)
  {
    ord = 'st';
  }
  else if (n % 10 == 2 && n % 100 != 12)
  {
    ord = 'nd';
  }
  else if (n % 10 == 3 && n % 100 != 13)
  {
    ord = 'rd';
  }

  return ord;
}
    // Function to replace [RECENTDATE] in a string with the actual date
    async function processString(inputString) {
      // Wait for the recentConvoDate to be set
      await articleGrabber()

      // Ensure recentConvoDate is available
      // Decode URL-encoded characters and replace [RECENTDATE] with the date
      let decodedString = inputString.replace(/%5cn/g, "\n").replace(/Customer/g,customerName).replace(/%27/,"'")

      if (recentConvoDate) {
      // Replace [RECENTDATE] with the actual date
        let finalString = decodedString.replace(/\[RECENTDATE\]/g,recentConvoDate)
        return finalString
        // Return the final string with replacements
      } else if(recentConvoDate = 12345){
        let finalString = decodedString.replace(/\[RECENTDATE\]/g,"XXX")
        return finalString
```

```
    }   }

  async function decode(inputString) {
    const result = await processString(inputString) // Get the result
    return result // Log or use the result
  }
.this is for calculating the relevance score with the keywords, it uses a data tree to be faster, which is why the code looks weird
  function calculateRelevance() { //W I P
    const returnedArticleData = returnArticleData()
    const tree = buildTree(returnedArticleData)
    for (let i = 0; i < titleArray.length; i++) {
      if (titleArray[i] != "TITLE COLUMN" && titleArray[i] != "KEYWORD COLUMN" && titleArray[i] != "CONTENT COLUMN") {
        const keywordArr = keywordArray[i].replaceAll(" ", "").split(",")
        let relevance = 0
        keywordArr.forEach((keyword) => {
          const weight = Number.parseInt(keyword.charAt(1))
          if (isNaN(weight) || weight == 0) { return }
          const word = keyword.substring(3)
          relevance += getWordCount(word) * weight
        })
        const lookup = macroArray.find(e => e.title === titleArray[i])
        if (Boolean(lookup)) { lookup.relevancePoints = relevance } else {
          macroArray.push(new macro(i, titleArray[i], contentArray[i], relevance))
        }
      }
      function macro(index, title, content, relevancePoints) {
        this.index = index
        this.title = title
        this.content = content
        this.relevancePoints = relevancePoints
      }
    }
    function getWordCount(word) {
      let currentNode = tree;

      for (let i = 0; i < word.length; i++) {
        const letter = word[i];

        // If the letter doesn't exist in the tree at this level, return 0
        if (!currentNode[letter]) {
          return 0;
        }

        // Move deeper into the tree
        currentNode = currentNode[letter];
      }

      // After traversing the word, check if 'count' exists, indicating the exact word was inserted
      return currentNode.count || 0; // Return the count, or 0 if the word isn't found
    }
    if (isPromptBoxActive) {
      const promptContainer = document.getElementById("macro-prompt")
      GM_setValue("promptX", promptContainer.style.left)
      GM_setValue("promptY", promptContainer.style.top)
      document.body.removeChild(promptContainer)
    }
    showDatalistPrompt("Please select macro", macroArray)
  }
.this just screams to the code when the url changes, so it runs again, the url changes when switching tickets too so this is a universal fix in this code for when
something was annoying to code in a persistent way, or broke whatever else was tried
  function handleUrlChange() {
    boxCount = 0
    refreshBox()
    currentTicketNr = window.location.href.toString().split("/").pop()
    recentConvoDate = undefined
    let count = 0
    if (window.location.href.indexOf("/agent/tickets/") != -1) {
      dateRefresh()
    }
    function dateRefresh() {
      if (count < 10) {
        count++
        articles = []
        setTimeout(() => {
          getResult().then((result) => {
```

```
                recentConvoDate = result
                //createMessageBox(recentConvoDate, 3000)//todelete
                if (recentConvoDate == undefined || recentConvoDate == 12345) {
                    dateRefresh()
                }
                else (calculateRelevance())
            })
        }, 500)
    }
}
function makeSure() {
    count = 8
    dateRefresh()
}
}
.this builds the tree of words, it makes a letter to letter tree, branching into the number of times it appears, its way faster than comparing 4000 different and repeating words
function buildTree(words) {
    let tree = {}
    words.forEach(word => {
        let currentNode = tree;
        for (let i = 0; i < word.length; i++) {
            const letter = word[i].toLowerCase();
            // If the letter doesn't exist as a property, create it
            if (currentNode == " ") { return }
            if (!currentNode[letter]) {
                currentNode[letter] = {};
            }
            // Move deeper into the tree
            currentNode = currentNode[letter];
        }
        // At the end of the word, increase the count or initialize it
        if (!currentNode.count) {
            currentNode.count = 0;
        }
        currentNode.count += 1;
    });
    return tree;
}
// Initial URL check

handleUrlChange()

// Listen for URL changes using popstate event
window.addEventListener("popstate", handleUrlChange)

// Also listen for pushState or replaceState method calls to detect changes in single-page apps
const originalPushState = history.pushState
history.pushState = function () {
    originalPushState.apply(this, arguments)
    handleUrlChange() // Handle URL change
}

const originalReplaceState = history.replaceState
history.replaceState = function () {
    originalReplaceState.apply(this, arguments)
    handleUrlChange()
}

.a quick keybind to pop up the macro window
function handleKeyPress(event) {
    if (event.altKey && event.key === 'b') {
        if (isPromptBoxActive) { document.getElementById('macro-prompt').style.left = '150px'; document.getElementById('macro-prompt').style.top = '10px' }
    }
}
document.addEventListener('keydown', handleKeyPress);
.this is for dragging the macro window.
document.addEventListener("mousemove", (e) => {
    if (isDragging) {
        const promptContainer = document.getElementById("macro-prompt")
        promptContainer.style.left = `${parseInt(promptContainer.style.left) - (lastX - e.clientX)}px`;
        lastX = e.clientX
        promptContainer.style.top = `${parseInt(promptContainer.style.top) - (lastY - e.clientY)}px`;
        lastY = e.clientY
    }
```

```
    });

    document.addEventListener("mouseup", (e) => {
        const promptContainer = document.getElementById("macro-prompt")
        const targetRect = document.getElementById('sidebar').getBoundingClientRect();
        if (e.clientX >= targetRect.left && e.clientX <= targetRect.right &&
            e.clientY >= targetRect.top && e.clientY <= targetRect.bottom && isDragging) {
            ; promptContainer.style.left = '-500px'; promptContainer.style.top = '-500px'; setTimeout(() => {
                GM_setValue("promptX", promptContainer.style.left)
                GM_setValue("promptY", promptContainer.style.top)
            }, 100);
        }
        isDragging = false;
        document.body.style.userSelect = "auto"; // Restore text selection
    });



}
```