// Import necessary libraries

import fetch from 'node-fetch';

import \* as JSZip from 'jszip';

// Define constants

const SCHEMA\_LINK = "http://schemas.openxmlformats.org/wordprocessingml/2006/main";

// Define interfaces for authentication and document handling

interface AuthOptions {

accessToken: string;

}

interface DocumentOptions {

fileUrl: string;

valuesToAdd: string[][];

contentControlsToEdit: { title: string; newText: string }[];

}

// Function to fetch and parse the Word document

async function fetchAndParseDocument(fileUrl: string, accessToken: string): Promise<Document> {

const response = await fetch(fileUrl, {

method: 'GET',

headers: {

'Authorization': `Bearer ${accessToken}`,

'Accept': 'application/vnd.openxmlformats-officedocument.wordprocessingml.document'

}

});

const blob = await response.blob();

const arrayBuffer = await blob.arrayBuffer();

const zip = new JSZip();

await zip.loadAsync(arrayBuffer);

const documentXml = await zip.file("word/document.xml").async("string");

const parser = new DOMParser();

return parser.parseFromString(documentXml, "application/xml");

}

// Function to add rows to the first table

function addRowsToTable(xmlDoc: Document, values: string[][]) {

const tables = xmlDoc.getElementsByTagName("w:tbl");

if (tables.length > 0) {

const firstTable = tables[0];

values.forEach(rowValues => {

const newRow = xmlDoc.createElementNS(SCHEMA\_LINK, "w:tr");

rowValues.forEach(value => {

const newCell = xmlDoc.createElementNS(SCHEMA\_LINK, "w:tc");

const newParagraph = xmlDoc.createElementNS(SCHEMA\_LINK, "w:p");

const newRun = xmlDoc.createElementNS(SCHEMA\_LINK, "w:r");

const newText = xmlDoc.createElementNS(SCHEMA\_LINK, "w:t");

newText.textContent = value;

// Set text format here if needed

newRun.appendChild(newText);

newParagraph.appendChild(newRun);

newCell.appendChild(newParagraph);

newRow.appendChild(newCell);

});

firstTable.appendChild(newRow);

});

}

}

// Function to edit content controls by title

function editContentControls(xmlDoc: Document, contentControlsToEdit: { title: string; newText: string }[]) {

const contentControls = xmlDoc.getElementsByTagName("w:sdt");

contentControlsToEdit.forEach(controlToEdit => {

for (let i = 0; i < contentControls.length; i++) {

const alias = contentControls[i].getElementsByTagName("w:alias")[0];

if (alias && alias.getAttribute("w:val") === controlToEdit.title) {

const textElement = contentControls[i].getElementsByTagName("w:t")[0];

if (textElement) {

textElement.textContent = controlToEdit.newText;

}

}

}

});

}

// Function to upload the modified document back to OneDrive

async function uploadDocument(fileUrl: string, accessToken: string, modifiedBlob: Blob) {

await fetch(fileUrl, {

method: 'PUT',

headers: {

'Authorization': `Bearer ${accessToken}`,

'Content-Type': 'application/vnd.openxmlformats-officedocument.wordprocessingml.document'

},

body: modifiedBlob

});

}

// Main function to fetch, edit, and upload the document

async function fetchEditAndUploadDocument(authOptions: AuthOptions, documentOptions: DocumentOptions) {

const { accessToken } = authOptions;

const { fileUrl, valuesToAdd, contentControlsToEdit } = documentOptions;

const xmlDoc = await fetchAndParseDocument(fileUrl, accessToken);

addRowsToTable(xmlDoc, valuesToAdd);

editContentControls(xmlDoc, contentControlsToEdit);

const serializer = new XMLSerializer();

const modifiedDocumentXml = serializer.serializeToString(xmlDoc);

const zip = new JSZip();

zip.file("word/document.xml", modifiedDocumentXml);

const modifiedArrayBuffer = await zip.generateAsync({ type: "arraybuffer" });

const modifiedBlob = new Blob([modifiedArrayBuffer], { type: 'application/vnd.openxmlformats-officedocument.wordprocessingml.document' });

await uploadDocument(fileUrl, accessToken, modifiedBlob);

}

// Example usage

const authOptions: AuthOptions = {

accessToken: 'YOUR\_ACCESS\_TOKEN'

};

const documentOptions: DocumentOptions = {

fileUrl: 'https://your-onedrive-url-to-the-document',

valuesToAdd: [

['Cell 1', 'Cell 2', 'Cell 3'],

['Cell 4', 'Cell 5', 'Cell 6']

],

contentControlsToEdit: [

{ title: 'ContentControlTitle1', newText: 'New Text 1' },

{ title: 'ContentControlTitle2', newText: 'New Text 2' }

]

};

fetchEditAndUploadDocument(authOptions, documentOptions).then(() => {

console.log("Document modified and uploaded successfully.");

}).catch(error => {

console.error("Error: ", error);

});