/\*\*

\* Function that is called during the JavaScript Task execution.

\* @param {IntegrationEvent} event

\*/

function executeScript(event) {

let input = event.getParameter('output');

let jsonOutput = {

message: {

Other\_Carrier\_Insurance: "No",

Member\_is\_calpers: "Yes",

claim\_details: {

"Claim Type": null,

"Claim id": null,

"Service From Date": null,

"Service Thru Date": null,

"total allowed amount": null,

"total charge amount": null,

"Member Name": null,

"Member DOB": null

},

Action: "Routed to claims ops"

}

};

if (typeof input === 'string') {

const claimTypeMatch = input.match(/Claim Type:\s\*([\w]+)/i);

const claimIdMatch = input.match(/Claim (?:ID|id):\s\*([\w\d]+)/i); // allow alphanumeric

const serviceFromMatch = input.match(/Service From Date:\s\*([\d.Ee+-]+)/i);

const serviceThruMatch = input.match(/Service Thru Date:\s\*([\d.Ee+-]+)/i);

const allowedAmountMatch = input.match(/total allowed amount:\s\*([\d.]+)/i);

const chargeAmountMatch = input.match(/total charge amount:\s\*([\d.]+)/i);

const nameMatch = input.match(/Name:\s\*([\w\s]+)/i);

const dobMatch = input.match(/DOB:\s\*([\d/]+)/i);

if (claimTypeMatch) jsonOutput.message.claim\_details["Claim Type"] = claimTypeMatch[1];

if (claimIdMatch) jsonOutput.message.claim\_details["Claim id"] = claimIdMatch[1];

if (serviceFromMatch) {

let serviceFromDate = Number(serviceFromMatch[1]);

jsonOutput.message.claim\_details["Service From Date"] = isNaN(serviceFromDate) ? serviceFromMatch[1] : serviceFromDate.toString();

}

if (serviceThruMatch) {

let serviceThruDate = Number(serviceThruMatch[1]);

jsonOutput.message.claim\_details["Service Thru Date"] = isNaN(serviceThruDate) ? serviceThruMatch[1] : serviceThruDate.toString();

}

if (allowedAmountMatch) {

let allowedAmount = parseFloat(allowedAmountMatch[1]);

jsonOutput.message.claim\_details["total allowed amount"] = allowedAmount.toFixed(2);

}

if (chargeAmountMatch) {

let chargeAmount = parseFloat(chargeAmountMatch[1]);

jsonOutput.message.claim\_details["total charge amount"] = chargeAmount.toFixed(2);

}

if (nameMatch) {

jsonOutput.message.claim\_details["Member Name"] = nameMatch[1].trim();

}

if (dobMatch) {

const dob = dobMatch[1].trim();

if (isValidDateFormat(dob)) {

jsonOutput.message.claim\_details["Member DOB"] = dob;

} else {

throw new Error('Invalid DOB format. Expected MM/DD/YY, got: ' + dob);

}

}

} else {

throw new Error('Input is not a valid string.');

}

// Final output - MUST BE STRING

event.setParameter('output', JSON.stringify(jsonOutput));

}

// Helper function to validate DOB

function isValidDateFormat(dob) {

const regex = /^(0[1-9]|1[0-2])\/(0[1-9]|[12]\d|3[01])\/\d{2}$/;

return regex.test(dob);

}