Code for Procurement request approval process using Google Apps Script

1. The procurement request approval process using Google Apps Script.

Role :- Developer

Insight:- JavaScript,Google app script,Google form short cuts,Google spreadsheet short cuts.

1. Handled form submissions and generated unique request IDs.
2. Sent automated emails to approvers and requestors using Gmail's SMTP service.
3. Implemented approval and decline workflows for managers and HODs.
4. Updated status columns in Google Sheets based on approval decisions.
5. Integrated on-edit triggers and managed data migration.
6. Created methods for generating approval links.

```
//HOD EMAIL WITH SAME ADDRESS


function onFormSubmit(e) {
  var userEmail = e.values[2].toLowerCase().trim();
  var sheet = SpreadsheetApp.openById('Spreadsheet
ID').getSheetByName('Sheet1');
  var row = findRowByEmail(sheet, userEmail);
```

```
  if (row !== -1) {
    var requestor = sheet.getRange(row, 1).getValue();
    var approver = getEmailforapprover(userEmail); // Get the
approver's email
    var requestID = generateUniqueRequestID();
```

```
    // Update the request ID in the form responses sheet
    var formResponsesSheet = SpreadsheetApp.openById('Spreadsheet
ID').getSheetByName('Form Responses 1');
    formResponsesSheet.getRange(formResponsesSheet.getLastRow(),
13).setValue(requestID);
```

```
    // Send detailed email to the approver
    notifyApprover(requestID, requestor, approver);
    updateApproverEmail(requestID, approver);
     // Send confirmation email to the employee
    sendConfirmationEmail(userEmail, requestID);




}
}
```

```
// Function to send confirmation email to the employee
```

```
function sendConfirmationEmail(userEmail, requestID) {
  var subject = "Procurement Request Confirmation";
  var formDetails = getFormDetails(requestID);
```

```
  var message = "Dear Employee, your procurement request with Request
ID: " + requestID + " has been submitted successfully." +
    "<p>Form Details:</p>" + "<p>" + formDetails + "</p>";
```

```
    // Use Gmail's SMTP service to send email from the developer's
email address
  var senderEmail = "sender@email_id"; // Replace with the developer's
email address
  var recipientEmail = userEmail;
```

```
  // Configure the SMTP parameters
  var smtpUsername = "sender@email_id"; // Replace with the developer's
Gmail username
  var smtpPassword = "passwordofyouremail"; // Replace with the
developer's Gmail password
  var smtpServer = "smtp.gmail.com";
  var smtpPort = 4650;
```

```
  // Send the email using Gmail's SMTP service
  MailApp.sendEmail({
    to: recipientEmail,
    subject: subject,
    body: message,
    htmlBody: message,
    from: senderEmail,
    replyTo: senderEmail,
    name: "sender@gmail.com", // Replace with the developer's name
    htmlOptions: {
      name: "sender@gmail.com", // Replace with the developer's name
      htmlBody: message,
    },
    noReply: true,
    // Configure SMTP settings
    // Uncomment the lines below and replace with appropriate values
    // smtpServer: smtpServer,
    // smtpPort: smtpPort,
    // smtpUsername: smtpUsername,
    // smtpPassword: smtpPassword,
    // tls: true,
  });
}
```

```
// function generateUniqueRequestID() {
//   var scriptProperties = PropertiesService.getScriptProperties();
//   var lastRequestID = scriptProperties.getProperty("lastRequestID");
//   if (!lastRequestID || parseInt(lastRequestID) > 0) {
//     lastRequestID = "0";
//   }
//   var newRequestID = parseInt(lastRequestID) + 1;
//   scriptProperties.setProperty("lastRequestID",
newRequestID.toString());
//   return "#" + newRequestID;
// }
```

```
function generateUniqueRequestID() {
  var scriptProperties = PropertiesService.getScriptProperties();
  var lastRequestID = scriptProperties.getProperty("lastRequestID");
```

```
  if (!lastRequestID || parseInt(lastRequestID) >
10000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000) {
    lastRequestID = 0;
  } else {
    lastRequestID = parseInt(lastRequestID);
  }
```

```
  // Increment the last request ID and store it in script properties
  var newRequestID = lastRequestID + 1;
  scriptProperties.setProperty("lastRequestID",
newRequestID.toString());
```

```
  return "#" + newRequestID;
}
```

```
// email content for approver1 with approve and decline button with
form details
function notifyApprover(requestID, requestor, approver) {
  var subject = "Procurement Request Approval Required";
  var formDetails = getFormDetails(requestID);
  var message = "Dear " + approver + ", an procurement request with
Request ID: " + requestID + " is awaiting your approval." +
    "<p>Form Details:</p>" + "<p>" + formDetails + "</p>" +
    '<form action="' + getApprovalLink("approve", requestID, requestor,
approver) + '" method="post" style="margin-bottom: 10px;">' +
    '<input type="submit" name="action" value="Approve">' +
    '</form>' +
    '<form action="' + getApprovalLink("decline", requestID, requestor,
approver) + '" method="post">' +
    '<input type="submit" name="action" value="Decline">' +
    '</form>';
```

```
 var senderEmail = "sender@gmail.com"; // Replace with the developer's
email address
  var recipientEmail = approver;
```

```
  // Configure the SMTP parameters
  var smtpUsername = "sender@gmail.com"; // Replace with the
developer's Gmail username
  var smtpPassword = "emailpassword"; // Replace with the developer's
Gmail password
  var smtpServer = "smtp.gmail.com";
  var smtpPort = 4650;
```

```
  // Send the email using Gmail's SMTP service
  MailApp.sendEmail({
    to: recipientEmail,
    subject: subject,
    body: message,
    htmlBody: message,
    from: senderEmail,
    replyTo: senderEmail,
    name: "sender@gmail.com", // Replace with the developer's name
    htmlOptions: {
      name: "sender@gmail.com", // Replace with the developer's name
      htmlBody: message,
    },
    noReply: true,
    // Configure SMTP settings
    // Uncomment the lines below and replace with appropriate values
    // smtpServer: smtpServer,
    // smtpPort: smtpPort,
    // smtpUsername: smtpUsername,
    // smtpPassword: smtpPassword,
    // tls: true,
  });
}
```

```
function getFormDetails(requestID) {
  var sheet =
SpreadsheetApp.openById('Spreadsheet_ID').getSheetByName('Form
Responses 1');
  var data = sheet.getDataRange().getValues();
  var formDetails = "";
```

```
// Find the column index for "Request ID"
  var headers = data[0];
  var requestIDColumn = headers.indexOf("Request ID");
```

```
  for (var i = 1; i < data.length; i++) {
    if (data[i][requestIDColumn] === requestID) {
```

```
      formDetails += "EMAIL: " + data[i][2] + "<br>"; //email is in the
4th column
      formDetails += "EMPLOYEE ID : " + data[i][3] + "<br>"; //Module
is in the 2nd column
      formDetails += "DEPARTMENT: " + data[i][4] + "<br>"; // Reason
for request is in the 3rd column
      formDetails += "PROJECT NAME: " + data[i][5] + "<br>";
      formDetails += "PRODUCTS BE PROCURED: " + data[i][6] + "<br>";
      formDetails += "QUANTITY: " + data[i][7] + "<br>";
      formDetails += "REQUIRED BY DATE : " + data[i][8] + "<br>";
      formDetails += "PRODUCT LINK (share any shopping link): " +
data[i][9] + "<br>";
      formDetails += "Reason :" + data[i][10] + "<br>";
      formDetails += "Request ID: " + data[i][requestIDColumn] + "<br>";
      break;
    }
  }
  return formDetails;
}
```

```
// for approver1 email
function getEmailforapprover(requestor){
  var sheet =
SpreadsheetApp.openById('Spreadsheet_ID').getSheetByName('Sheet1'); //
Replace with your spreadsheet ID and sheet name
  var data = sheet.getDataRange().getValues();
  Logger.log(data);
  requestor = requestor.toLowerCase();
  for (var i = 1; i < data.length; i++) { // from row 2,headers in row
1
    if (data[i][0].toLowerCase() === requestor) { // Checks if the
requestor email matches
      return data[i][1]; // Return to approver email
    }
  }
  return 'sender@gmail.com';
}
// to generate approval links deploy u will get this link
function getApprovalLink(action, requestID, requestor, approver) {
  var baseUrl = "Deployment url";
```

```
  // Construct the approval link with the provided parameters
  var link = baseUrl + "?action=" + action +
             "&requestID=" + encodeURIComponent(requestID) +
             "&requestor=" + encodeURIComponent(requestor) +
             "&approver=" + encodeURIComponent(approver);
```

```
  return link;
}
```

```
// this function is used to match the employee email with approver
email id
function findRowByEmail(sheet, email) {
  // the email is in the first column
  var data = sheet.getRange(1, 1, sheet.getLastRow(), 1).getValues();
  for (var i = 0; i < data.length; i++) {
    if (data[i][0].toLowerCase().trim() === email) {
      //it is used to convert the uppercase email id to lower case
email id
      return i + 1;
    }
  }
  return -1; // Return -1 if email is not found
}
```

```
// Function to send confirmation email to the employee based on the
approver's action
function sendEmployeeConfirmationEmail(userEmail, requestID, action,
comments) {

  var subject = "Manager Request Confirmation";
  var formDetails = getFormDetails(requestID);
```

```


  var message = "Dear Employee, your procurement request with Request
ID: " + requestID + " has been " + action + " By Manager." +
    "<p>Form Details:</p>" + "<p>" + formDetails + "</p>" +
    '<p>Manager Status: '+ action +'</p>';
    var senderEmail = "sender@gmail.com"; // Replace with the
developer's email address
    var recipientEmail = userEmail;
```

```
  // Configure the SMTP parameters
  var smtpUsername = "sender@gmail.com"; // Replace with the
developer's Gmail username
  var smtpPassword = "email password"; // Replace with the developer's
Gmail password
  var smtpServer = "smtp.gmail.com";
  var smtpPort = 4650;
```

```
  // Send the email using Gmail's SMTP service
  MailApp.sendEmail({
    to: recipientEmail,
    subject: subject,
    body: message,
    htmlBody: message,
    from: senderEmail,
    replyTo: senderEmail,
    name: "sender@gmail.com", // Replace with the developer's name
    htmlOptions: {
      name: "sender@gmail.com", // Replace with the developer's name
      htmlBody: message,
    },
    noReply: true,
    // Configure SMTP settings
    // Uncomment the lines below and replace with appropriate values
    // smtpServer: smtpServer,
    // smtpPort: smtpPort,
    // smtpUsername: smtpUsername,
    // smtpPassword: smtpPassword,
    // tls: true,
  });
}
```

```
// Function to get the existing status
```

```
function getStatus(requestID) {
  // Get the spreadsheet containing form responses
  var formResponsesSheet =
SpreadsheetApp.openById('Spreadsheet_ID').getSheetByName('Form
Responses 1');
```

```
  // Find the row in the spreadsheet based on the request ID
  var data = formResponsesSheet.getDataRange().getValues();
  var headers = data[0];
  var requestIDColumn = headers.indexOf("Request ID");
  var statusColumn = headers.indexOf("approver1 status");
```

```
  for (var i = 1; i < data.length; i++) {
    if (data[i][requestIDColumn] === requestID) {
      return data[i][statusColumn];
    }
  }
```

```
  // If the request ID is not found, return an empty status
  return "";
}
```

```
// Function to handle approval logic
function handleApproval(requestID, requestor, approver, comments ) {
  // Update the status and approver email columns
  updateStatus(requestID, "Approved By Manager", 14); // Assuming 7 is
the index of the desired status column
  updateHODEmail(requestID);
  updateUID(requestID, 1);
  // Send confirmation email to the employee
  sendEmployeeConfirmationEmail(requestor, requestID, "Approved",
comments);
}
```

```
// Function to update the uid column
function updateUID(requestID, uidValue) {
  // Get the spreadsheet containing form responses
  var formResponsesSheet =
SpreadsheetApp.openById('Spreadsheet_ID').getSheetByName('Form
Responses 1');
```

```
  // Find the row in the spreadsheet based on the request ID
  var data = formResponsesSheet.getDataRange().getValues();
  var headers = data[0];
  var requestIDColumn = headers.indexOf("Request ID");
  var uidColumn = headers.indexOf("uid");
  var approver1StatusColumn = headers.indexOf("approver1 status");
```

```
  for (var i = 1; i < data.length; i++) {
    if (data[i][requestIDColumn] === requestID) {
      // Check the value in the 'approver1 status' column
      var approver1Status = data[i][approver1StatusColumn];
```

```
      // If the status is 'Approved', update the uid column
      if (approver1Status === 'Approved') {
        formResponsesSheet.getRange(i + 1, uidColumn +
1).setValue(uidValue);
      }
      // If the status is 'Declined', update the uid column to 0
      if (approver1Status === 'Declined') {
        formResponsesSheet.getRange(i + 1, uidColumn +
1).setValue(uidValue);
      }
```

```
      // You can add additional logic here if needed
      break;
    }
  }
}
```

```
// Function to handle decline logic
function handleDecline(requestID, requestor, approver , comments) {
  // Update the status and approver email columns
  updateStatus(requestID, "Declined By Manager", 14); // Assuming 7 is
the index of the desired status column
  updateUID(requestID, 0); // Update the uid column to 0
```

```
  // Send confirmation email to the employee
  sendEmployeeConfirmationEmail(requestor, requestID, "Declined",
comments);
}
```

```
// Function to update the approver email column
function updateApproverEmail(requestID, approver) {
  // Get the spreadsheet containing form responses
  var formResponsesSheet =
SpreadsheetApp.openById('Spreadsheet_ID').getSheetByName('Form
Responses 1');
```

```
  // Find the row in the spreadsheet based on the request ID
  var data = formResponsesSheet.getDataRange().getValues();
  var headers = data[0];
  var requestIDColumn = headers.indexOf("Request ID");
  var approverColumn = headers.indexOf("approver1"); // Assuming 6th
column (JavaScript uses 0-based indexing)
```

```
  for (var i = 1; i < data.length; i++) {
    if (data[i][requestIDColumn] === requestID) {
      // Update the approver email in the 6th column
      formResponsesSheet.getRange(i + 1, approverColumn +
1).setValue(approver);
      break;
    }
  }
}
```

```
// Add an onEdit trigger function
function onFormEdit(e) {
```

```
  var formResponsesSheet =
SpreadsheetApp.openById('Spreadsheet_ID').getSheetByName('Form
Responses 1');
```

```
  var editedRange = e.range;
  var editedColumn = editedRange.getColumn();
  var headers = formResponsesSheet.getRange(1, 1, 1,
formResponsesSheet.getLastColumn()).getValues()[0];
```

```
  // Check if the edited column corresponds to the 'hodemail' column
  var hodEmailColumn = headers.indexOf("hodemail") + 1; // Adjust the
index as needed
```

```
  if (editedColumn === hodEmailColumn) {
    // Get the edited row and request ID
    var editedRow = editedRange.getRow();
    var requestID = formResponsesSheet.getRange(editedRow,
headers.indexOf("Request ID") + 1).getValue();
```

```
   // Get the new HOD email
    var newHODEmail = formResponsesSheet.getRange(editedRow,
hodEmailColumn).getValue();
```

```
    // Trigger email to HOD with the new email
    sendEmailToHODAfterUpdate(newHODEmail, requestID);
  }
}
function sendEmailToHODAfterUpdate(hodEmail, requestID,approverStatus)
{
  var subject = "HOD Procurement Email - Request ID: " + requestID;
  var formDetails = getFormDetails(requestID);
  var message = "Dear HOD, an procurement request with Request ID: " +
requestID + " is awaiting your approval." +
    "<p>Form Details:</p>" + "<p>" + formDetails + "</p>" +
    '<p> Manager Status: '+ 'Approved' +'</p>' +
    '<form action="' + getApprovalLink("approveHOD", requestID,
hodEmail) + '" method="post" style="margin-bottom: 10px;">' +
    '<input type="submit" name="action" value="Approve">' +
    '</form>' +
    '<form action="' + getApprovalLink("declineHOD", requestID,
hodEmail) + '" method="post">' +
    '<input type="submit" name="action" value="Decline">' +
    '</form>';
```

```
  // Use Gmail's SMTP service to send email from the developer's email
address
  var senderEmail = "sender@gmail.com"; // Replace with the developer's
email address
  var recipientEmail = hodEmail;
```

```
  // Configure the SMTP parameters
  var smtpUsername = "sender@gmail.com"; // Replace with the
developer's Gmail username
  var smtpPassword = "password"; // Replace with the developer's Gmail
password
  var smtpServer = "smtp.gmail.com";
  var smtpPort = 4650;
```

```
  // Send the email using Gmail's SMTP service
  MailApp.sendEmail({
    to: recipientEmail,
    subject: subject,
    body: message,
    htmlBody: message,
    from: senderEmail,
    replyTo: senderEmail,
    name: "sender@gmail.com", // Replace with the developer's name
    htmlOptions: {
      name: "sender@gmail.com", // Replace with the developer's name
      htmlBody: message,
    },
    noReply: true,
    // Configure SMTP settings
    // Uncomment the lines below and replace with appropriate values
    // smtpServer: smtpServer,
    // smtpPort: smtpPort,
    // smtpUsername: smtpUsername,
    // smtpPassword: smtpPassword,
    // tls: true,
  });
}
```

```
// Function to get HOD email based on the requestor email
function getHODEmail(requestorEmail) {
  // Get the sheet containing HOD details
  var hodSheet =
SpreadsheetApp.openById('Spreadsheet_ID').getSheetByName('Sheet1');
  var hodData = hodSheet.getDataRange().getValues();
```

```
  requestorEmail = requestorEmail.toLowerCase();
```

```
  for (var i = 1; i < hodData.length; i++) {
    var sheetRequestor = hodData[i][0].toLowerCase();
```

```
    if (sheetRequestor === requestorEmail) {
      var hodEmail = hodData[i][2];
```

```
      Logger.log('Match found. Requestor: ' + requestorEmail + ', HOD
Email: ' + hodEmail);
```

```
      return hodEmail;
    }
  }
  Logger.log('No match found for requestor: ' + requestorEmail);
  return 'sender@gmail.com'; // Replace with the default HOD email or
appropriate logic
}

// Function to generate HOD link
function getHODLink(action, requestID, requestor, hodEmail) {
  var baseUrl = "Deployement_ID";

  // Construct the HOD link with the provided parameters
  var link = baseUrl + "?action=" + action +
             "&requestID=" + encodeURIComponent(requestID) +
             "&requestor=" + encodeURIComponent(requestor) +
             "&hodEmail=" + encodeURIComponent(hodEmail);

  return link;
}
// Function to update the HOD email column
function updateHODEmail(requestID) {
  // Get the spreadsheet containing form responses
  var formResponsesSheet =
SpreadsheetApp.openById('Spreadsheet_ID').getSheetByName('Form
Responses 1');

  // Find the row in the spreadsheet based on the request ID
  var data = formResponsesSheet.getDataRange().getValues();
  var headers = data[0];
  var requestIDColumn = headers.indexOf("Request ID");
  var requestorColumn = headers.indexOf("EMAIL");
  var hodEmailColumn = headers.indexOf("hodemail");

  for (var i = 1; i < data.length; i++) {
    if (data[i][requestIDColumn] === requestID) {
      var requestorEmail = data[i][requestorColumn].toLowerCase();

      // Assuming you have a function to get HOD email based on the
requestor or some other criteria
      var hodEmail = getHODEmail(requestorEmail);

      // Update the HOD email in the 'hodemail' column
      formResponsesSheet.getRange(i + 1, hodEmailColumn +
1).setValue(hodEmail);
```

```
      // Send email to HOD after updating HOD email
      sendEmailToHODAfterUpdate(hodEmail, requestID);
```

```
      // You can add additional logic here if needed
      break;
    }
  }
}
// Function to handle HOD approval logic
function handleHODApproval(requestID, hodEmail) {
  // Update the status column to 'Approved' for HOD
  updateStatus(requestID, "Approved by HOD", 18); // Assuming 9 is the
index of the desired status column
  sendEmployeeConfirmationEmailFromHOD(getEmployeeEmail(requestID),
requestID, "Approved by HOD");
  sendEmployeeConfirmationEmailToProcurement(requestID, hodEmail);
```

```
}
```

```
// Function to handle HOD decline logic
function handleHODDecline(requestID, hodEmail) {
  // Update the status column to 'Declined' for HOD
  updateStatus(requestID, "Declined by HOD", 18); // Assuming 9 is the
index of the desired status column
  sendEmployeeConfirmationEmailFromHOD(getEmployeeEmail(requestID),
requestID, "Declined by HOD");

}
```

```
// Function to get the employee email based on the request ID
function getEmployeeEmail(requestID) {
  var sheet =
SpreadsheetApp.openById('Spreadsheet_ID').getSheetByName('Form
Responses 1');
  var data = sheet.getDataRange().getValues();
  var requestIDColumn = data[0].indexOf("Request ID");
```

```
  for (var i = 1; i < data.length; i++) {
    if (data[i][requestIDColumn] === requestID) {
      return data[i][2]; // Assuming email is in the third column,
adjust if needed
    }
  }
  return "";
}
```

```
// Function to send confirmation email to procurement
function sendEmployeeConfirmationEmailToProcurement(requestID, hodEmail
{
  var subject = "Procurement Request Approved by Manager and HOD -
Request ID: " + requestID;
  var formDetails = getFormDetails(requestID);
  var message = "Dear Procurement Team, an procurement request with
Request ID: " + requestID + " has been approved by the Manager and
HOD." +
    "<p>Form Details:</p>" + "<p>" + formDetails + "</p>" +
    '<p>Manager Status: '+ 'Approved' +'</p>' +
    '<p>HOD Status: '+ 'Approved' +'</p>';
  // Use Gmail's SMTP service to send email
  var senderEmail = "sender@gmail.com"; // Replace with the developer's
email address
  var recipientEmail = "sender@gmail.com"; // Replace with the
Procurement Team's email address
```

```
  // Configure the SMTP parameters
  var smtpUsername = "sender@gmail.com"; // Replace with the
developer's Gmail username
  var smtpPassword = "Password"; // Replace with the developer's Gmail
password
  var smtpServer = "smtp.gmail.com";
  var smtpPort = 4650;
```

```
  // Send the email using Gmail's SMTP service
  MailApp.sendEmail({
    to: recipientEmail,
    subject: subject,
    body: message,
    htmlBody: message,
    from: senderEmail,
    replyTo: senderEmail,
    name: "sender@gmail.com", // Replace with the developer's name
    htmlOptions: {
      name: "sender@gmail.com", // Replace with the developer's name
      htmlBody: message,
    },
    noReply: true,
    // Configure SMTP settings
    // Uncomment the lines below and replace with appropriate values
    // smtpServer: smtpServer,
    // smtpPort: smtpPort,
    // smtpUsername: smtpUsername,
    // smtpPassword: smtpPassword,
    // tls: true,
  });
}
```

```
// Function to update the status column
function updateStatus(requestID, status, columnIndex) {
  // Get the spreadsheet containing form responses
  var formResponsesSheet =
SpreadsheetApp.openById('Spreadsheet_ID').getSheetByName('Form
Responses 1');
```

```
  // Find the row in the spreadsheet based on the request ID
  var data = formResponsesSheet.getDataRange().getValues();
  var headers = data[0];
  var requestIDColumn = headers.indexOf("Request ID");
```

```
  for (var i = 1; i < data.length; i++) {
    if (data[i][requestIDColumn] === requestID) {
      // Validate the columnIndex to ensure it's within the valid range
      if (columnIndex >= 1 && columnIndex <= headers.length) {
        // Update the status column with the specified index
        formResponsesSheet.getRange(i + 1,
columnIndex).setValue(status);
      } else {
        Logger.log('Invalid columnIndex: ' + columnIndex);
        // Handle the error or log it as needed
      }
```

```

      break;
    }
  }
}
```

```
//Handle HTTP POST requests
function doPost(e) {
  var parameters = e.parameter;
  var action = parameters.action;
  var requestID = parameters.requestID;
  var requestor = parameters.requestor;
  var approver = parameters.approver;
  var hodEmail = parameters.hodEmail;

// Check if the request status is already set
  if (action === "approve") {
    // Handle the regular approval logic
    handleApproval(requestID, requestor, approver);
  } else if (action === "decline") {
    // Handle the regular decline logic
    handleDecline(requestID, requestor, approver);
```

```

}
// Additional check for HOD status
```

```
else if (action === "approveHOD") {
  // Handle the HOD approval logic
  handleHODApproval(requestID, hodEmail);
} else if (action === "declineHOD") {
  // Handle the HOD decline logic
  handleHODDecline(requestID, hodEmail);

}
 // Return a response (optional)
  return ContentService.createTextOutput("Request handled
successfully");
}
```

```
// Function to get the HOD status
function getHODStatus(requestID) {
  // Get the spreadsheet containing form responses
  var formResponsesSheet =
SpreadsheetApp.openById('Spreadsheet_ID').getSheetByName('Form
Responses 1');

  // Find the row in the spreadsheet based on the request ID
  var data = formResponsesSheet.getDataRange().getValues();
  var headers = data[0];
  var requestIDColumn = headers.indexOf("Request ID");
  var hodStatusColumn = headers.indexOf("hod status");

  for (var i = 1; i < data.length; i++) {
    if (data[i][requestIDColumn] === requestID) {
      return data[i][hodStatusColumn];
    }
  }

  // If the request ID is not found, return an empty status
  return "";
}
```

```
function migrateDataToNewStructure() {
  var formResponsesSheet =
SpreadsheetApp.openById('Spreadsheet_id').getSheetByName('Form
Responses 1');
  var data = formResponsesSheet.getDataRange().getValues();
  var headers = data[0];
```

```
  // Map old index-based references to new column name-based references
  var columnMappings = {
    6: 'approver1', // Change the index and column name accordingly
    7: 'Request ID',
    8: 'approver1 status',
    9: 'uid',
    10: 'hodemail',
    11: 'hod status'

    // Add more mappings as needed
  };
```

```
  // Create a map of old column indexes to new column names
  var columnIndexMap = headers.reduce(function (map, header, index) {
    map[index + 1] = header; // Index is 1-based in columnMappings
    return map;
  }, {});
```

```
  // Iterate through each row and update the data
  for (var i = 0; i < data.length; i++) {
    Object.keys(columnMappings).forEach(function (oldIndex) {
      var newIndex = columnMappings[oldIndex];
      var oldValue = data[i][oldIndex - 1]; // Adjust for 0-based index
      var newIndexColumn = columnIndexMap[newIndex];
```

```
      // If the new column exists, update the value
      if (newIndexColumn !== undefined) {
        formResponsesSheet.getRange(i + 1, newIndexColumn +
1).setValue(oldValue);
      }
    });
  }

  Logger.log('Data migration completed.');
}

// // Add an email function for HOD confirmation
function sendEmployeeConfirmationEmailFromHOD(userEmail, requestID,
action) {
  var subject = "Procurement Request " + action + " by HOD";
  var formDetails = getFormDetails(requestID);

  var message = "Dear Employee, your procurement request with Request
ID: " + requestID + " has been " + action.toLowerCase() + " by the
HOD." +
    "<p>Form Details:</p>" + "<p>" + formDetails + "</p>" +
    '<p>HOD Status: '+ action +'</p>';

  var senderEmail = "sender@gmail.com"; // Replace with the developer's
email address
  var recipientEmail = userEmail;

  // Configure the SMTP parameters
  var smtpUsername = "sender@gmail.com"; // Replace with the
developer's Gmail username
  var smtpPassword = "Password"; // Replace with the developer's Gmail
password
  var smtpServer = "smtp.gmail.com";
  var smtpPort = 4650;

  // Send the email using Gmail's SMTP service
  MailApp.sendEmail({
    to: recipientEmail,
    subject: subject,
    body: message,
    htmlBody: message,
    from: senderEmail,
    replyTo: senderEmail,
    name: "sender@gmail.com", // Replace with the developer's name
    htmlOptions: {
      name: "sender@gmail.com", // Replace with the developer's name
      htmlBody: message,
    },
    noReply: true,
  });
}
```