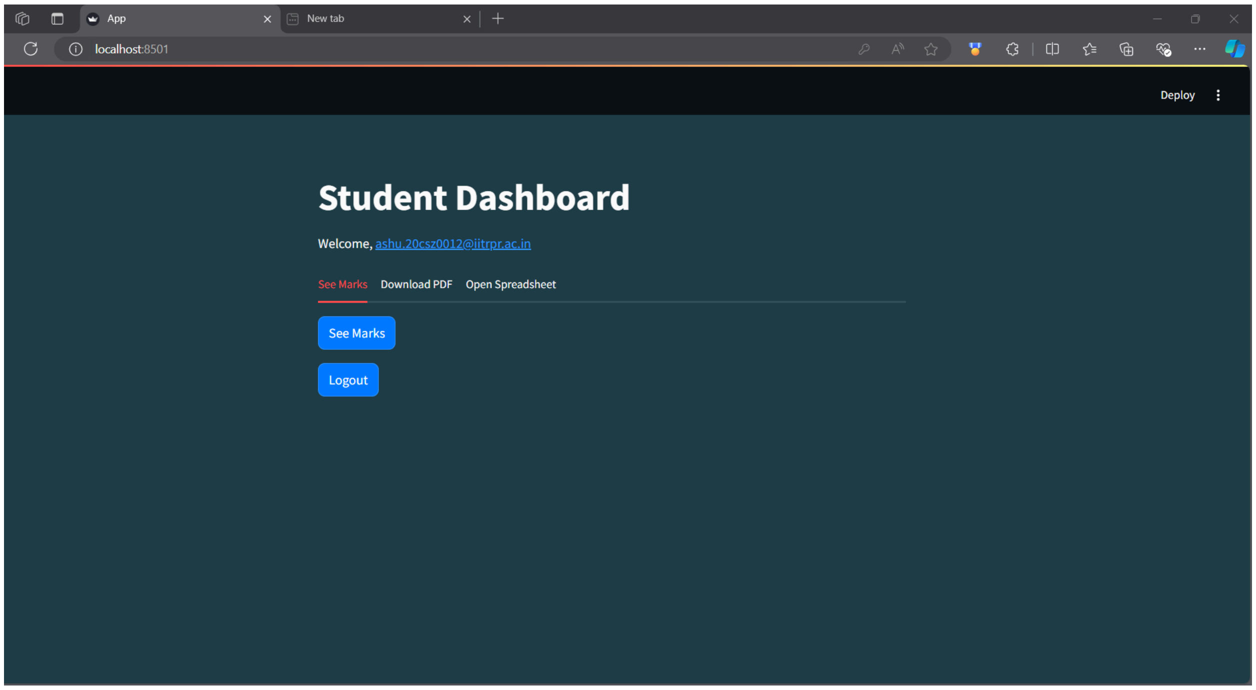
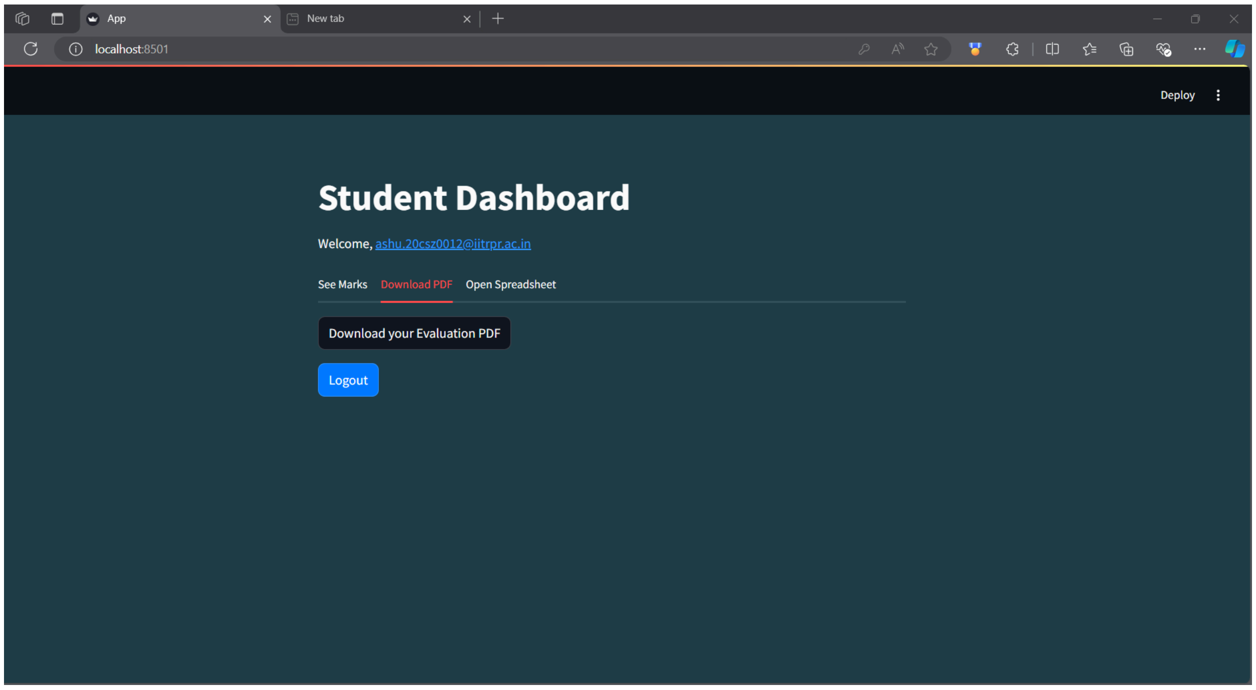
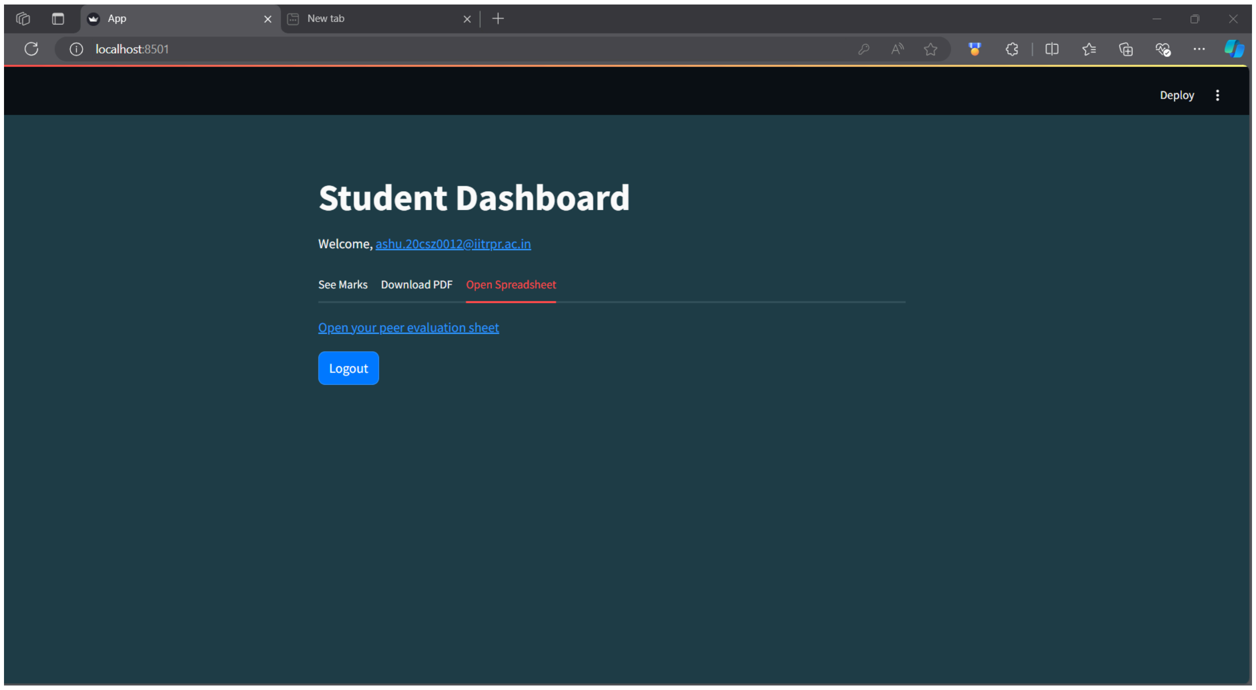
Peer Evaluation System UI/UX

Sample Screenshots of the UI/UX design: -

- The changes from the today's code are reflected below: -





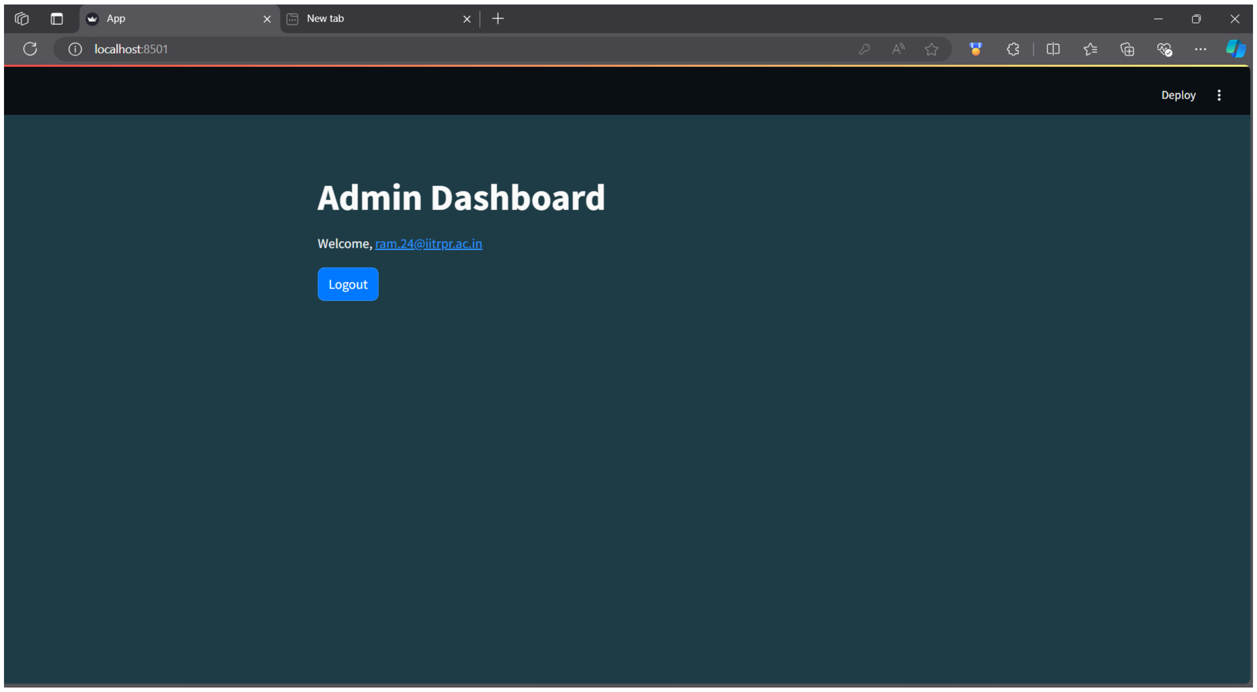


Student Dashboard

Welcome, ashu.20csz0012@iitrpr.ac.in

See Marks Download PDF Open Spreadsheet

Open your peer evaluation sheet



Admin Dashboard

Welcome, ram.24@iitrpr.ac.in

Code: -

1. **Python: -**

```
import io
import gspread
import requests
import streamlit as st
from googleapiclient.discovery import build
from googleapiclient.http import MediaIoBaseUpload
from googleapiclient.http import MediaIoBaseDownload
from oauth2client.service_account import ServiceAccountCredentials

# Google Sheets and Google Drive setup
SCOPE = [
  "https://spreadsheets.google.com/feeds",
  "https://www.googleapis.com/auth/drive"
]
CREDENTIALS_FILE = "D:/ROHIT IIT/Peer
Evaluation/peer-evaluation-sem1-e2fcf8b5fc27.json"
SHEET_NAME = "UserRoles"


# Initialize connection to Google Sheets
def connect_to_google_sheets():
  creds = ServiceAccountCredentials.from_json_keyfile_name(CREDENTIALS_FILE,
SCOPE)
  client = gspread.authorize(creds)
  sheet = client.open(SHEET_NAME).sheet1
  return sheet


# Google Drive authentication
def authenticate_drive():
  creds = ServiceAccountCredentials.from_json_keyfile_name(CREDENTIALS_FILE,
SCOPE)
  service = build('drive', 'v3', credentials=creds)
  return service
```

```
# Fetch users from Google Sheets
def get_users_from_sheets():
  sheet = connect_to_google_sheets()
  records = sheet.get_all_records()
  return records


# Add new user to Google Sheets
def register_user(username, password, role):
  sheet = connect_to_google_sheets()
  new_user = [username, password, role]
  sheet.append_row(new_user)


# Verify user credentials
def login(username, password, users):
  for user in users:
     if user['username'] == username and user['password'] == password:
        st.session_state["login_status"] = True
        st.session_state["role"] = user["role"]
        st.session_state["username"] = username
        st.session_state["page"] = "dashboard"
        st.session_state["message"] = None
        return
  st.session_state["message"] = "Incorrect username or password"


# Logout function
def logout():
  st.session_state["login_status"] = False
  st.session_state["role"] = None
  st.session_state["username"] = None
  st.session_state["page"] = "login"
  st.session_state["message"] = "Logged out successfully"


def trigger_google_apps_script(function_name):
  web_app_url =
"https://script.google.com/macros/s/AKfycbwlBil062YhNYcbIqmP9obfLBKgoeIdTdRD
```

```
Q_BOB4rF1S6JhTxvVFH8MhW2x84bgyAVag/exec"  # Replace with your web app
URL
   url = f"{web_app_url}?action={function_name}"  # Append the function name as the
'action' parameter
   try:
       response = requests.get(url)
       if response.status_code == 200:
           st.success(f"{function_name} executed successfully!")
       else:
           st.error(f"Failed to execute {function_name}. Status code:
{response.status_code}")
   except Exception as e:
       st.error(f"An error occurred: {str(e)}")


def admin_dashboard():
  st.title("Admin Dashboard")
  st.write(f"Welcome, {st.session_state['username']}")

def teacher_dashboard():
  st.title("Teacher Dashboard")
  st.write(f"Welcome, {st.session_state['username']}")

  if st.button("Pre Evaluation"):
      trigger_google_apps_script("PreEval")

  if st.button("Check Pending Evaluation's"):
      trigger_google_apps_script("CheckEval")

  if st.button("Post Evaluation"):
      trigger_google_apps_script("PostEval")

      # Button to trigger Function 2
  if st.button("Generate Charts"):
      trigger_google_apps_script("GenChart")

  if st.button("Send Mark's"):
      trigger_google_apps_script("SendMail")
```

```
# Function to check if a file already exists in Google Drive folder
def file_exists(drive_service, folder_id, file_name):
  query = f"'{folder_id}' in parents and name='{file_name}'"
  results = drive_service.files().list(q=query, spaces='drive', fields='files(id,
name)').execute()
  files = results.get('files', [])
  return any(file['name'] == file_name for file in files)


# Function to upload PDF files to Google Drive
def upload_pdfs(uploaded_files, folder_id):
  drive_service = authenticate_drive()
  count = 0

  for uploaded_file in uploaded_files:
     if file_exists(drive_service, folder_id, uploaded_file.name):
        #st.warning(f"PDF file '{uploaded_file.name}' already exists in the folder.")
        continue

     file_metadata = {
        'name': uploaded_file.name,
        'parents': [folder_id]
     }
     media = MediaIoBaseUpload(uploaded_file, mimetype='application/pdf')
     drive_service.files().create(body=file_metadata, media_body=media,
fields='id').execute()
     count = count + 1
     #st.session_state["success_message"] = f"Uploaded PDF file '{uploaded_file.name}'
to Google Drive"

  st.success(f" The {count} files are uploaded to the Google Drive.")


# Function to upload Google Sheets files to Google Drive
def upload_sheets(uploaded_files, folder_id):
  drive_service = authenticate_drive()

  for uploaded_file in uploaded_files:
     if file_exists(drive_service, folder_id, uploaded_file.name):
            #st.warning(f"Google Sheet file '{uploaded_file.name}' already exists in the
folder.")
            continue

        file_metadata = {
            'name': uploaded_file.name,
            'parents': [folder_id],
            'mimeType': 'application/vnd.google-apps.spreadsheet'
        }
        media = MediaIoBaseUpload(uploaded_file, mimetype='application/vnd.ms-excel')
        drive_service.files().create(body=file_metadata, media_body=media,
fields='id').execute()

    st.success("The Excel sheet has been uploaded to the Google Drive.")


# Role-based content: Teacher Dashboard with multiple file uploads
def ta_dashboard():
    st.title("TA Dashboard")
    st.write(f"Welcome, {st.session_state['username']}")

    # Folder ID for the Google Drive folder where the files will be saved
    folder_id = "1fT-inciLQut85BGEQrjMSWbVRcTsdWfQ"  # Replace this with your
folder ID

    # Allow file upload for multiple Google Sheets
    st.subheader("Upload Google Sheets")
    sheet_files = st.file_uploader("Upload Google Sheets", type=["xlsx"],
accept_multiple_files=True,
                                    key="sheet_uploader")

    if sheet_files:
        upload_sheets(sheet_files, folder_id)

    # Allow file upload for multiple PDFs
    st.subheader("Upload PDF Files")
    pdf_files = st.file_uploader("Upload PDF files", type=["pdf"],
accept_multiple_files=True, key="pdf_uploader")

    if pdf_files:
```

```
    upload_pdfs(pdf_files, folder_id)


# Helper function to connect to a specific Google Sheet
def connect_to_google_sheets_with_name(sheet_name):
  creds = ServiceAccountCredentials.from_json_keyfile_name(CREDENTIALS_FILE,
SCOPE)
  client = gspread.authorize(creds)
  sheet = client.open(sheet_name)
  return sheet



def get_student_details(username):
  # Connect to the specific Google Sheet containing marks
  sheet_name = "UI/UX Copy of Peer Evaluation2"
  sheet = connect_to_google_sheets_with_name(sheet_name)  # Modify to accept a sheet
name
  peer_eval_sheet = sheet.worksheet('PeerEval')  # Open the "PeerEval" sheet

  # Fetch all the data from the "PeerEval" sheet
  records = peer_eval_sheet.get_all_records()

  # Find marks for the current user
  for record in records:
    if record['EMail ID'] == username:   # Ensure this matches your column name
      return record['Average Marks'], record['Unique ID'], record['Spreadsheet Link']  #
Returning the Average Mark's and Unique id

  return None, None, None  # If no marks found for the user

# Fetch the student's PDF from Google Drive using unique ID
def get_student_pdf(unique_id):
  drive_service = authenticate_drive()
  folder_id = "1fT-inciLQut85BGEQrjMSWbVRcTsdWfQ"
  query = f"'{folder_id}' in parents and name contains '{unique_id}'"
  results = drive_service.files().list(q=query, fields="files(id, name)").execute()
  files = results.get('files', [])

  if files:
    file_id = files[0]['id']
```

```
        file_name = files[0]['name']

        # Download the PDF
        request = drive_service.files().get_media(fileId=file_id)
        fh = io.BytesIO()
        downloader = MediaIoBaseDownload(fh, request)
        done = False
        while not done:
            status, done = downloader.next_chunk()

        fh.seek(0)
        return fh, file_name

    return None, None


def student_dashboard():
    st.title("Student Dashboard")
    st.write(f"Welcome, {st.session_state['username']}")

    if st.session_state["username"]:
        # Fetch marks, unique ID, and spreadsheet link using the session's username
        marks, unique_id, sheet_link = get_student_details(st.session_state["username"])
    else:
        st.error("Username is Incorrect!")

    # Creating tabs
    tab1, tab2, tab3 = st.tabs(["See Marks", "Download PDF", "Open Spreadsheet"])

    # Tab for viewing marks
    with tab1:
        if st.button("See Marks"):
            if marks and unique_id:
                st.write(f"Your evaluation marks are =  {marks}")
            else:
                st.error("No marks are available.")

    # Tab for downloading PDF
    with tab2:
        pdf_file, file_name = get_student_pdf(unique_id)
        if pdf_file:
            st.download_button(
                label="Download your Evaluation PDF",
                data=pdf_file,
                file_name=file_name,
                mime='application/pdf'
            )
        else:
            st.error("PDF not found.")

    # Tab for opening the peer evaluation spreadsheet
    with tab3:
        if sheet_link:
            st.markdown(f"[Open your peer evaluation sheet]({sheet_link})",
unsafe_allow_html=True)
        else:
            st.error("Spreadsheet link not found.")



# Main Streamlit app
def main():
    # Initialize session state variables if not present
    if "login_status" not in st.session_state:
        st.session_state["login_status"] = False
    if "role" not in st.session_state:
        st.session_state["role"] = None
    if "username" not in st.session_state:
        st.session_state["username"] = None
    if "page" not in st.session_state:
        st.session_state["page"] = "login"
    if "message" not in st.session_state:
        st.session_state["message"] = None
    if "success_message" not in st.session_state:
        st.session_state["success_message"] = None

    # Set background color and input field styling using HTML
    st.markdown(
        """
        <style>
        .stApp {
            background-color: #1f3f49;  /* Light blue background */
        }
        .stTextInput>div>input, .stPasswordInput>div>input {
            background-color: white;  /* White background for text and password inputs */
            color: black;  /* Text color for input fields */
        }
        .stButton>button {
            background-color: #007bff;  /* Optional: Style buttons with a color */
            color: white;
        }
        </style>
        """,
        unsafe_allow_html=True
    )

    # Page routing based on session state
    if st.session_state["page"] == "login":
        st.title("Peer Evaluation System")

        # Tabs for Login and Registration
        tab1, tab2 = st.tabs(["Login", "Register"])

        with tab1:
            st.header("Login")

            with st.form(key='login_form'):
                username = st.text_input("Username")
                password = st.text_input("Password", type="password")
                submit_button = st.form_submit_button("Login")

                if submit_button:
                    users = get_users_from_sheets()
                    login(username, password, users)
                    if st.session_state["login_status"]:
                        st.rerun()

        with tab2:
            st.header("Register")
```

```
            with st.form(key='register_form'):
                reg_username = st.text_input("Username", key='reg_username')
                reg_password = st.text_input("Password", type="password",
key='reg_password')
                role = st.selectbox("Role", ["Admin", "Teacher", "TA", "Student"])
                register_button = st.form_submit_button("Register")

                if register_button:
                    if not reg_username.endswith("@iitrpr.ac.in"):
                        st.error("Username must end with @iitrpr.ac.in")
                    else:
                        users = get_users_from_sheets()
                        if any(user['username'] == reg_username for user in users):
                            st.error("Username already exists")
                        else:
                            register_user(reg_username, reg_password, role)
                            st.success("User registered successfully")
                            # Redirect to the login page
                            st.session_state["page"] = "login"
                            st.rerun()

    elif st.session_state["page"] == "dashboard":
        if st.session_state["role"] == "Admin":
            admin_dashboard()
        elif st.session_state["role"] == "Teacher":
            teacher_dashboard()  # Updated function for Teacher Dashboard
        elif st.session_state["role"] == "TA":
            ta_dashboard()
        elif st.session_state["role"] == "Student":
            student_dashboard()

        # Logout button
        if st.button("Logout"):
            logout()
            st.rerun()


if __name__ == "__main__":
    main()
```

2. **Appscript: -**

```
function onOpen()
{

  var ui = SpreadsheetApp.getUi();
  ui.createMenu('Peer Evaluation')
    .addItem('Pre Evaluation', 'runMainPreEval')
    .addItem('Check Evaluation\'s Pending', 'runCheckEval')
    .addItem('Post Evaluation', 'runMainPostEval')
    .addItem('Generate Charts', 'generateCharts')
    .addItem('Send Marks','runSendMail')
    .addToUi();
}

var source_folder = "1fT-inciLQut85BGEQrjMSWbVRcTsdWfQ";
var target_folder = "1l4z7x3Twah6Qd8LQUepZHvmR0tYlY5cj";
var students_per_batch = countStudentsPerBatch();
var num_Questions = 1;

function countStudentsPerBatch()
{
  var folderId = source_folder;  // Replace with your folder's ID
  var folder = DriveApp.getFolderById(folderId);  // Get the folder by ID
  var files = folder.getFiles();  // Get all files in the folder

  var fileCount = 0;
  var students_per_batch = 0;

  while (files.hasNext())
{
    files.next();
    fileCount++;
  }

  students_per_batch = Math.floor(Math.sqrt(fileCount));

  // Log or return the count of files
  Logger.log('Students per Batch are: ' + students_per_batch);
  return students_per_batch;

}


function doGet(e)
{
  var action = e.parameter.action;  // Get the 'action' parameter from the URL
  if (action == "PreEval")
   {
    return runMainPreEval();
   }
  else if (action == "CheckEval")
   {
    return runCheckEval();
   }
  else if (action == "PostEval")
   {
    return runMainPostEval();
   }
  else if (action == "GenChart")
   {
    return generateCharts();
   }
  else if (action == "SendMail")
   {
    return runSendMail();
   }
  else
   {
    return ContentService.createTextOutput("Invalid function call.");
   }
}

function runMainPreEval()
{
  // Calling the mainPreEval to run all the necessary functions

  mainPreEval(source_folder, target_folder, students_per_batch, num_Questions);
}
```

```
function runCheckEval()
{
  //Call the function from Eval Check.gs to check for the peer's who don't evaluated the
sheets yet
  evalMarksInSheets();
  emailPeerPendingEval();
}

function runMainPostEval()
{
  // Calling the mainPreEval to run all the necessary functions
  mainPostEval(num_Questions);
}


function generateCharts()
{
  // Call the function from Graph.gs to generate charts
  runAllChartFunctions();
}

function runSendMail()
{
  //Call the function from Mail.gs to send the final mark's of each student
  sendMailToAllStudents();
}

function sendMailToAllStudents() {
  mapPeerAverageMarks()
  sendEmailByMarks()
}

function mapPeerAverageMarks() {

  var sourceSheetName = "Evaluation Results";
  var targetSheetName = "PeerEval";

  var sourceSheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName(sourceSheetName);
```

```
 var targetSheet =
SpreadsheetApp.getActiveSpreadsheet().getSheetByName(targetSheetName);

 var sourceData = sourceSheet.getDataRange().getValues();

 var peerAverageRow = -1;
 for (var i = 0; i < sourceData.length; i++) {
   if (sourceData[i][0] === "Peer Average") {
     peerAverageRow = i;
     break;
   }
 }

 if (peerAverageRow === -1) {
   Logger.log('Peer Average row not found.');
   return;
 }

 var peerIDs = sourceData[0];
 var peerAverageMarks = sourceData[peerAverageRow];


 var targetData = targetSheet.getDataRange().getValues();

 var headers = targetData[0];
 var averageMarksColIndex = headers.indexOf("Average Marks");

 if (averageMarksColIndex === -1) {
   averageMarksColIndex = headers.length;
   targetSheet.getRange(1, averageMarksColIndex + 1).setValue("Average Marks");
 }

 for (var i = 1; i < targetData.length; i++) {
   var targetPeerID = targetData[i][2];
   var peerIndex = peerIDs.indexOf(targetPeerID);

   if (peerIndex !== -1) {
     var averageMark = peerAverageMarks[peerIndex];
     targetSheet.getRange(i + 1, averageMarksColIndex + 1).setValue(averageMark);
   } else {
     targetSheet.getRange(i + 1, averageMarksColIndex + 1).setValue("Not Found");
   }
 }
}

/*
function sendEmailByMarks() {

 var sheetName = "PeerEval";

 var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName(sheetName);
 var data = sheet.getDataRange().getValues();

 var headers = data[0];

 var nameColIndex = headers.indexOf("Name");
 var emailColIndex = headers.indexOf("EMail ID");
 var marksColIndex = headers.indexOf("Average Marks");

 if (emailColIndex === -1 || marksColIndex === -1) {
   Logger.log('Required columns not found.');
   return;
 }

 for (var i = 1; i < data.length; i++) {
   var name = data[i][nameColIndex];
   var email = data[i][emailColIndex];
   var averageMarks = data[i][marksColIndex];

   if (name && email && averageMarks !== "") {
     var subject = "Your Average Evaluation Marks based on Peer Evaluation";
     var body = "Dear " + name + ",<br><br>" +
                "Your average marks are     <b><span style='color: red; font-size: 20px;'>" + 
averageMarks +
                "</span></b>      , based on Peer Evaluation.<br><br>Best regards,<br>CSE, 
IIT Ropar";

     MailApp.sendEmail({to: email, subject: subject, htmlBody: body});
   }
 }
```

```
}

*/
function sendEmailByMarks() {

  var sheetName = "PeerEval";
  var folderName = "Source Folder";

  var folder = getFolderByName(folderName);
  if (!folder) {
    Logger.log("Folder not found for path: " + folderPath);
    return;
  }

  var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName(sheetName);
  var data = sheet.getDataRange().getValues();

  var headers = data[0];

  var nameColIndex = headers.indexOf("Name");
  var emailColIndex = headers.indexOf("EMail ID");
  var uidColIndex = headers.indexOf("Unique ID");
  var marksColIndex = headers.indexOf("Average Marks");

  if (emailColIndex === -1 || marksColIndex === -1 || uidColIndex === -1 ||
marksColIndex === -1) {
    Logger.log('Required columns not found.');
    return;
  }

  for (var i = 1; i < data.length; i++) {
    var name = data[i][nameColIndex];
    var email = data[i][emailColIndex];
    var uid = data[i][uidColIndex];
    var averageMarks = data[i][marksColIndex];

    if (name && email && uid && averageMarks !== "") {
      var subject = "Your Average Evaluation Marks";
      var body = "Dear " + name + ",<br><br>" +
            "Your average marks are <b><span style='color: red; font-size: 20px;'>" +
averageMarks +
            "</span></b>.<br><br> Please find below the attached PDF file of your
Quiz/Exam.<br><br>Best regards,<br>CSE, IIT Ropar";

    var files = folder.getFilesByName(uid + ".pdf");

    if (files.hasNext()) {
      var file = files.next();
      var attachment = file.getAs(MimeType.PDF);

      MailApp.sendEmail({
        to: email,
        subject: subject,
        htmlBody: body,
        attachments: [attachment]
      });

      Logger.log("Email sent to " + email + " with attachment " + file.getName());
    } else {
      Logger.log("No file found for " + uid);

      MailApp.sendEmail({
        to: email,
        subject: subject,
        htmlBody: body
      });
      Logger.log("Email sent to " + email + " without attachment.");
    }
   }
  }
}

function getFolderByName(folderName) {
  var folders = DriveApp.getFoldersByName(folderName);
  if (folders.hasNext()) {
    return folders.next();
  }
  return null;
}
```