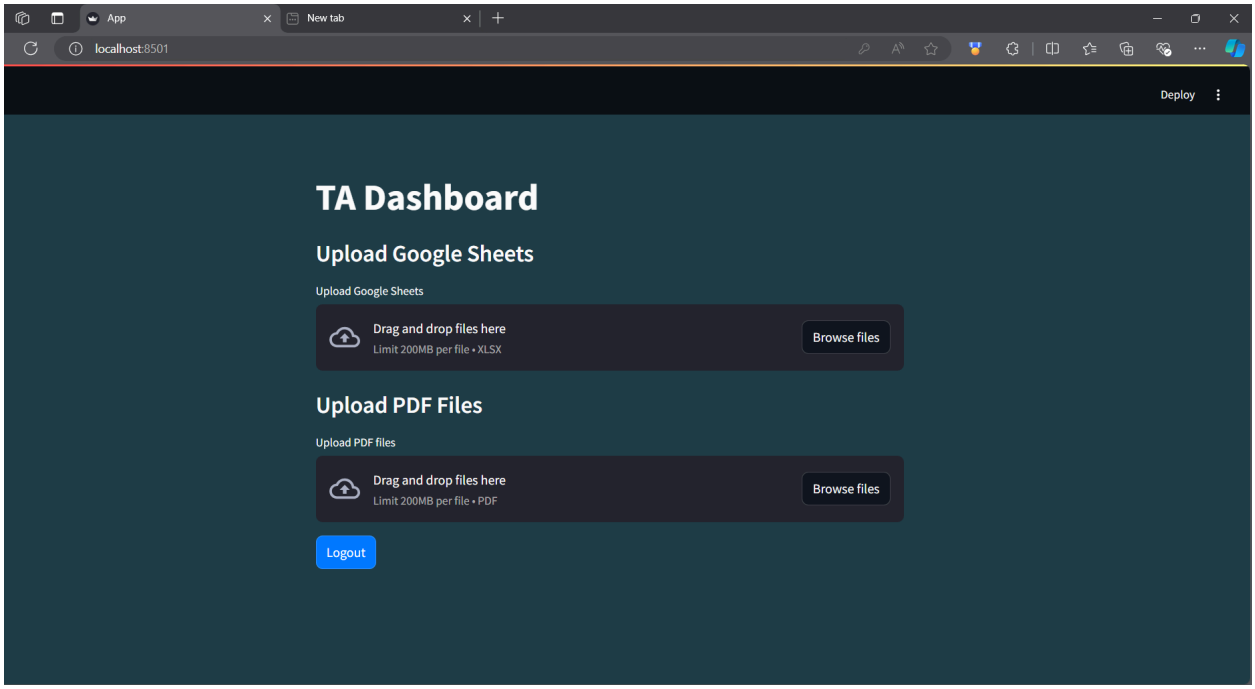
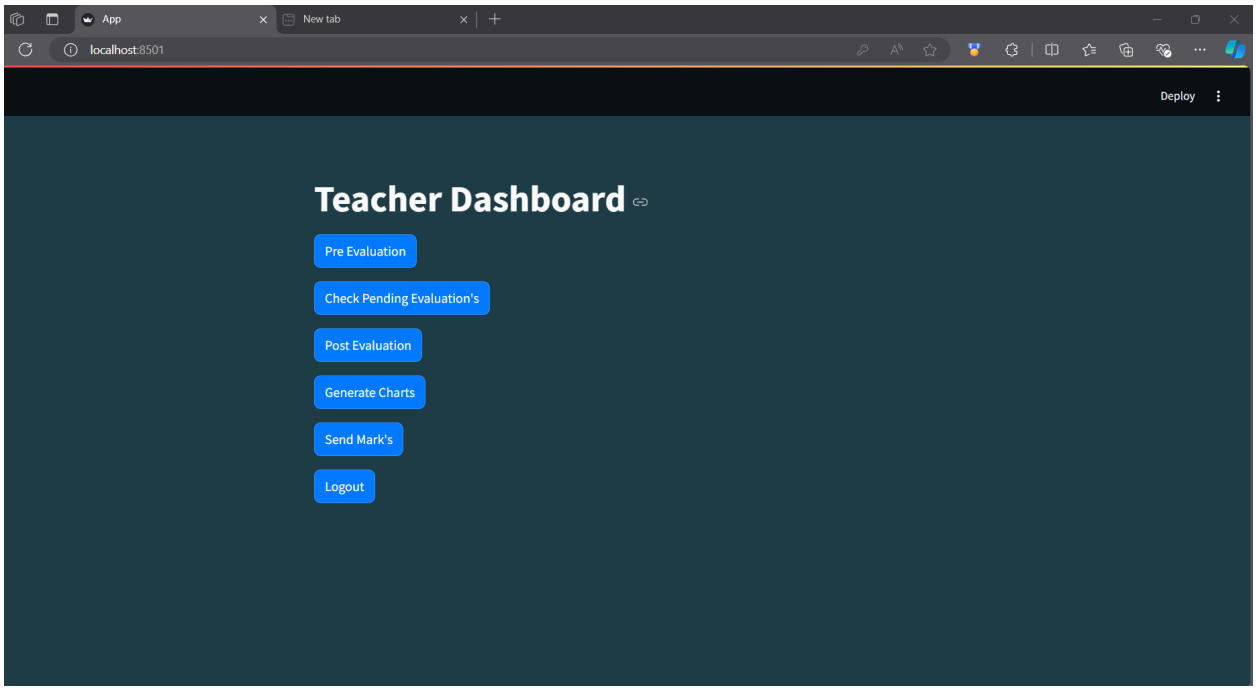
Peer Evaluation System UI/UX

Sample Screenshots of the proposed UI/UX design: -

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





Code: -

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


# 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("Admins can manage everything.")

def teacher_dashboard():
    st.title("Teacher Dashboard")
    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")

    # 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)


def student_dashboard():
    st.title("Student Dashboard")
    st.write("Students can evaluate peers and view feedback.")
```

```
# 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()
```