Peer Evaluation System UI/UX

This document contains information regarding the proposed format for the UI/UX of the Peer Evaluation System. We have discussed various formats for creating the UI/UX and have decided on the following format for development. The interface will include multiple roles for the login process, such as Admin, Teacher, Student/Peer, and Teaching Assistant. Admin users will have maximum access, allowing them to perform all operations, including those of a teacher or other roles.

Problem Statement: -

The Peer Evaluation System needs a user-friendly interface that supports multiple roles: Admin, Teacher, Student/Peer, and Teaching Assistant. Each role requires specific functionalities, with Admins having full control over evaluations, Teachers and Assistants managing assessments, and Students/Peers submitting and viewing evaluations.

The challenge is to design a secure, intuitive interface with seamless navigation, enabling efficient task performance and minimal training. The system must handle complex workflows like submissions/uploading exam sheets and feedback while being responsive across devices to ensure accessibility for all users.

Objective: -

The Peer Evaluation System's UI/UX design aims to create a simple, intuitive, and responsive interface for multiple user roles—Admin, Teacher, Student/Peer, and Teaching Assistant. The design will ensure seamless navigation, enabling users to easily perform tasks based on their access levels.

Admins will have full control over evaluations, users, and results, while Teachers and Teaching Assistants can manage assessments and feedback. Students/Peers will have an easy-to-use platform for peer evaluations and feedback. The system will be accessible across devices, reduce complexity, and enhance user satisfaction with minimal training.

Scope: -

The scope of the system is well-suited for addressing the challenges of flip teaching, where students study independently, ask questions, and are evaluated daily through quizzes. Peer evaluation helps distribute the grading workload, reducing the burden on Teaching Assistants and teachers, as students assess each other's quizzes. This system ensures an efficient and balanced evaluation process while supporting active learning.

Methodology: -

The following are the steps for creation of the above desired system: -

- **Role-Based Access:** Define user roles (Admin, Teacher, Student/Peer) with specific permissions for uploading, reviewing, and evaluating scanned quiz/exam sheets.
- **Design and Upload:** Develop a user-friendly interface for Teachers/Admins to upload scanned documents, and provide students with an intuitive dashboard to view and evaluate these documents.
- **Evaluation Process:** Implement a peer evaluation workflow where students can review and grade quizzes, with feedback compiled for Teacher/Assistant review.
- **Security and Privacy:** Ensure secure access controls and data protection for all uploaded and evaluated documents.
- **Responsiveness:** Design a responsive interface accessible across devices, supporting both desktop and mobile users.
- **Testing and Iteration:** Conduct user testing, gather feedback, and refine the design for optimal usability.

System Architecture: -

- **Frontend:** Interface for uploading quiz sheets and Excel files based on user type; dashboard for students to evaluate assigned quizzes.
- **Backend:** Processes uploaded files, extracts unique IDs, forms evaluation groups, and distributes sheets to students, ensuring no student evaluates their own sheet.
- **Database:** Stores user data, document metadata, and evaluation records.
- **Security:** Ensures secure access and protects data confidentiality.
- **Evaluation Process:** Multiple students evaluate each sheet; average marks are calculated and sent to the original student.
- **Integration:** Includes notifications and reporting tools.
- **Testing and Maintenance:** Regular testing and updates for functionality and support.

User Interface (UI) Design: -

- The UI will be designed with a focus on simplicity and functionality, aiming to provide educators with a seamless and intuitive experience through a Streamlit application. It will facilitate effortless navigation, allowing educators to easily upload answer keys and access detailed reports.
- The UI will enhance efficiency by incorporating interactive charts to clearly present and process information, making student performance analysis straightforward. The system will ensure smooth data upload and extraction into the database, improving the overall user experience.

Data Security: -

- Our system will prioritize student data protection through robust security measures, including end-to-end encryption for both data transfer and storage, to safeguard sensitive information from unauthorized access.
- Teachers' login credentials will be essential for effective student data management, and the use of a Sheets will ensure a structured environment that allows for easy access, updates, and versatile data storage.

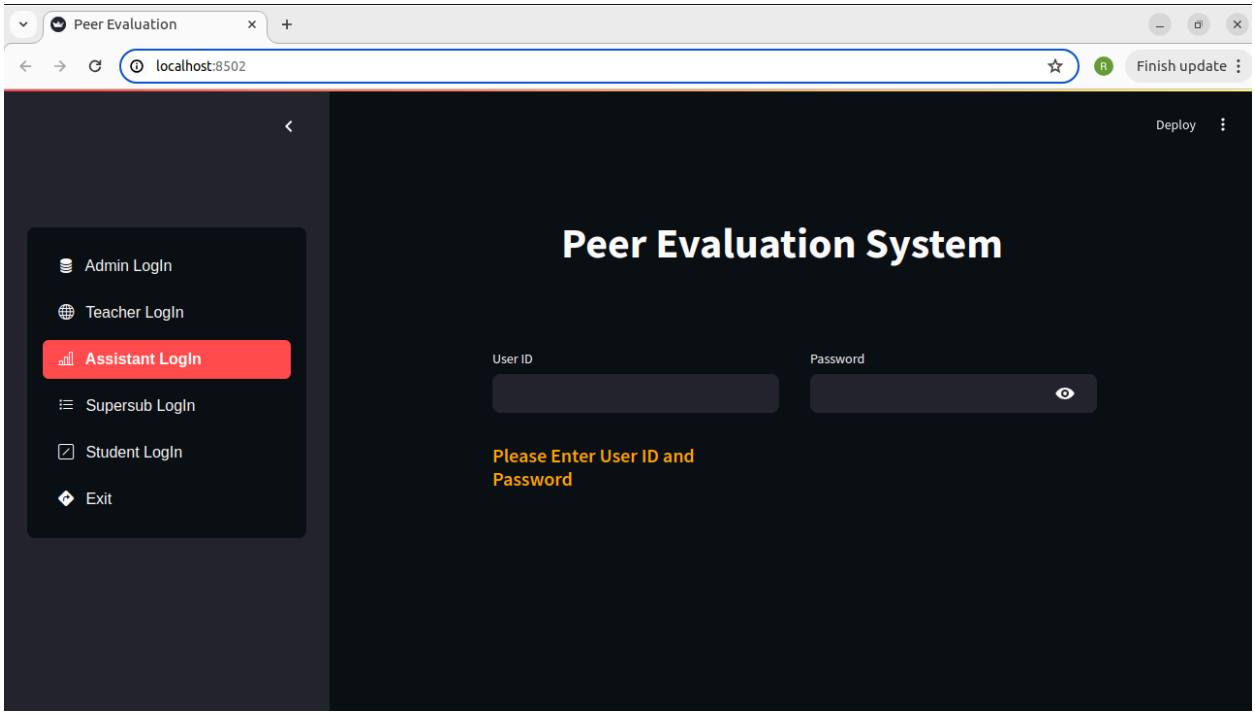
Conclusion: -

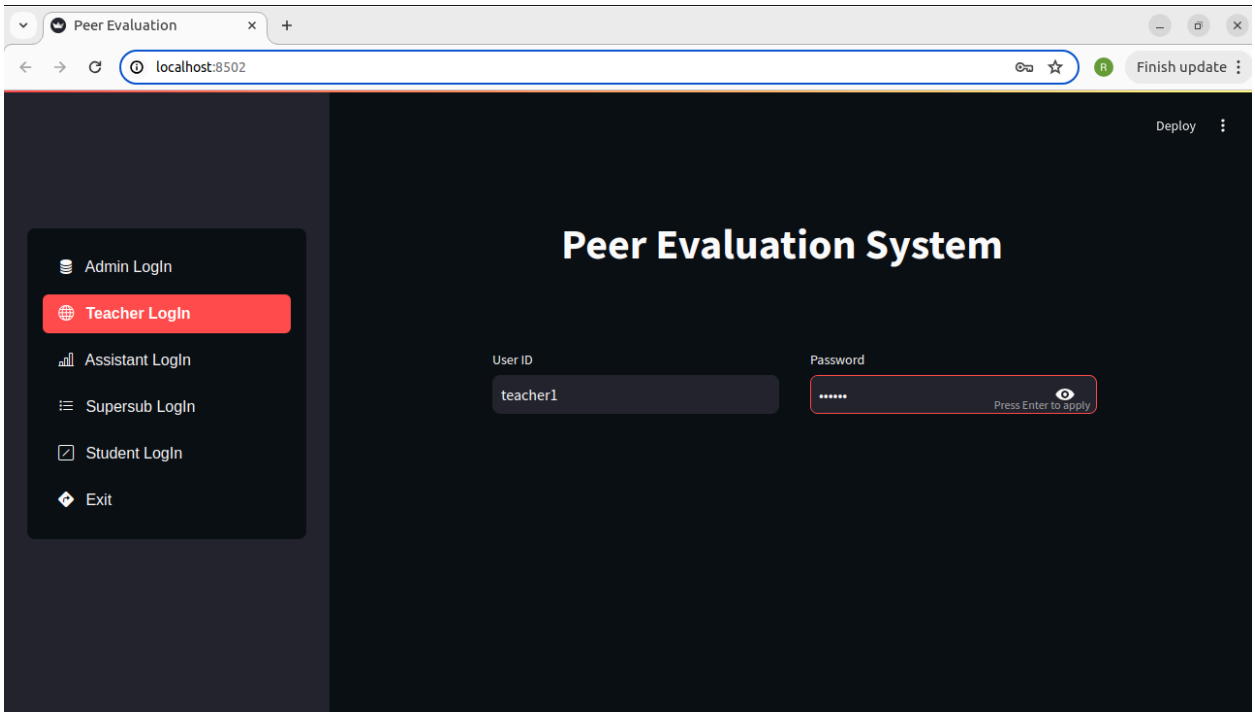
The proposed UI/UX design for the Peer Evaluation System will deliver a user-friendly and responsive platform for managing evaluations across multiple roles, including Admin, Teacher, Student/Peer, and Teaching Assistant. With a focus on simplicity and functionality, the design will enable smooth uploads of quiz sheets, seamless peer evaluations, and easy access to detailed reports. Interactive charts will aid in performance analysis, while robust security measures, including end-to-end encryption, will ensure the protection of sensitive data.

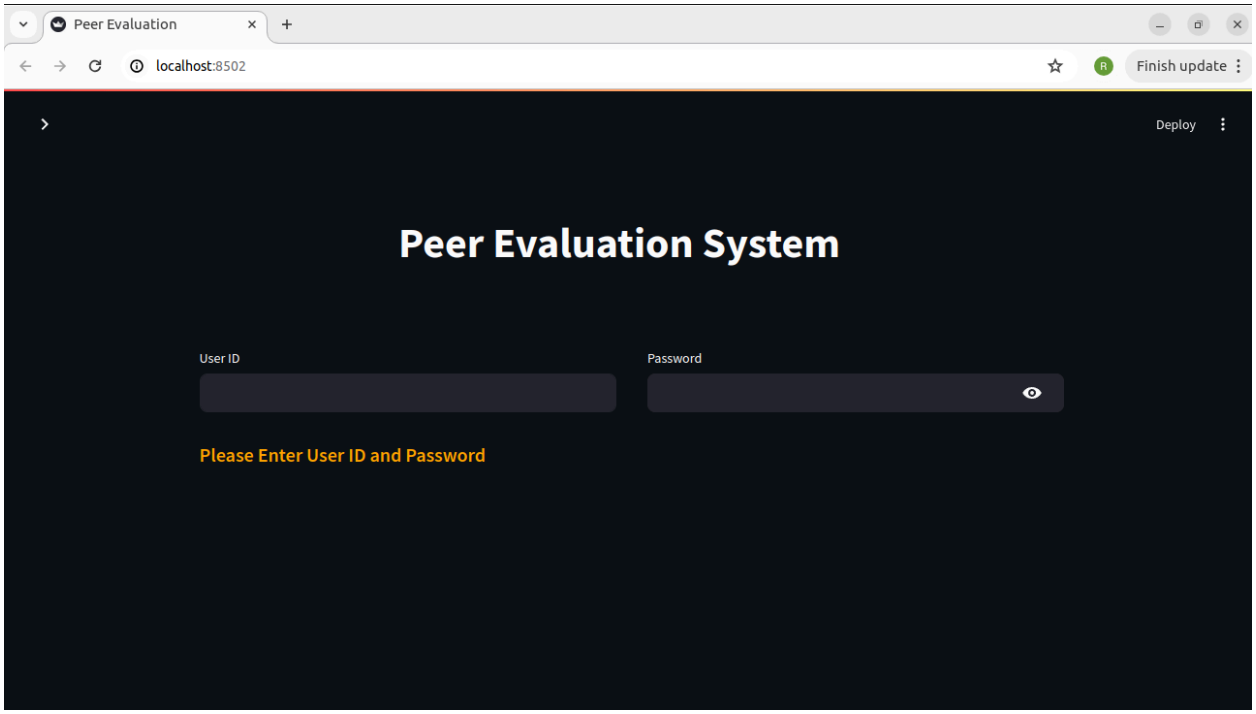
The system will utilize a structured data storage approach to support effective management and updates. Methodology includes defining role-based access, creating an intuitive interface, and ensuring security and responsiveness. Through ongoing testing and refinement, the system will enhance usability, effectively balance evaluation workloads, and provide a secure and accessible platform for all users.

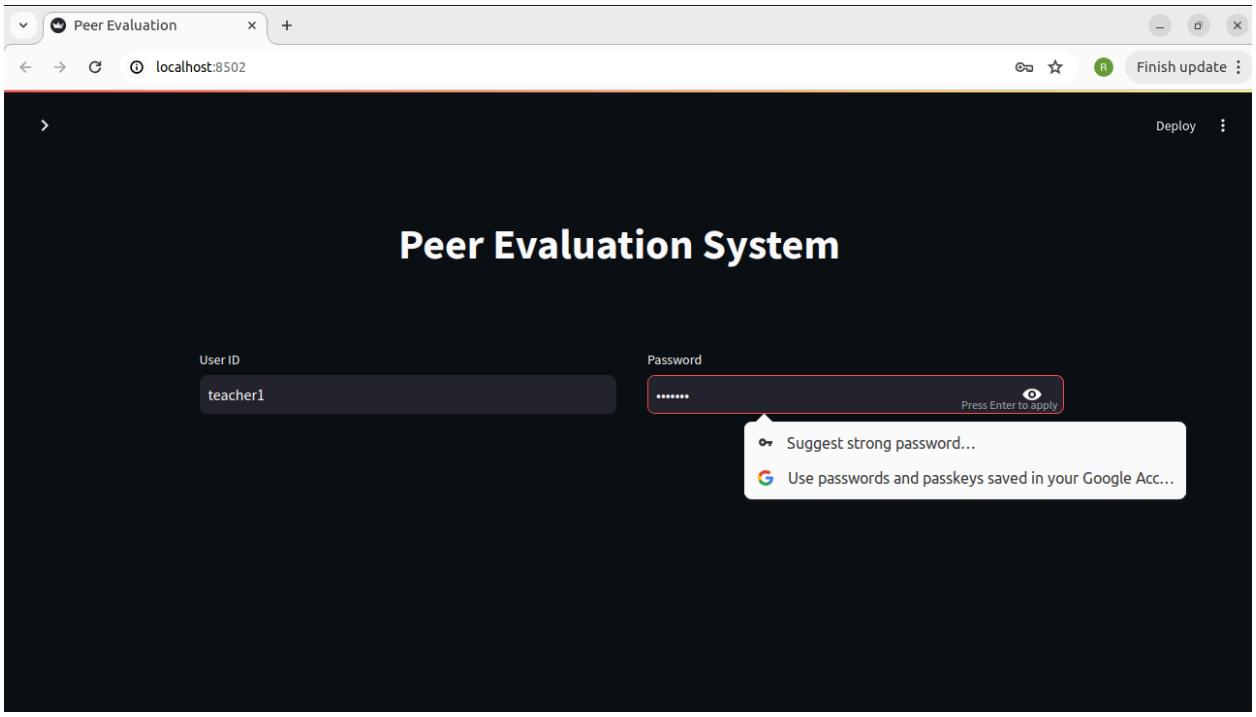
Sample Screenshots of the proposed UI/UX design: -

- The below images shows the sample Login window for the proposed system: -






Peer Evaluation
localhost:8502
Finish update
Deploy
Peer Evaluation System
User ID
Password
Please Enter User ID and Password


Peer Evaluation
localhost:8502
Finish update
Deploy
Peer Evaluation System
User ID
teacher1
Password
Press Enter to apply
Suggest strong password...
Use passwords and passkeys saved in your Google Acc...

Code: -

```
import os
import time
import numpy as np
import pandas as pd
import psycopg2
import bcrypt
import streamlit as st
from streamlit_option_menu import option_menu
from streamlit_extras.add_vertical_space import add_vertical_space
from dotenv import load_dotenv
import warnings

warnings.filterwarnings('ignore')
load_dotenv()

def streamlit_config():
    # page configuration
    st.set_page_config(page_title='Peer Evaluation', layout="wide")

    # page header transparent color
    page_background_color = """
    <style>

    [data-testid="stHeader"]
    {
    background: rgba(0,0,0,0);
    }

    </style>
    """
    st.markdown(page_background_color, unsafe_allow_html=True)

    # title and position
    st.markdown(f'<h1 style="text-align: center;">Peer Evaluation System</h1>',
                unsafe_allow_html=True)
    add_vertical_space(4)


class sql:

    def create_database():
```

```
        try:
            connection = psycopg2.connect(host=os.getenv('HOST'),
                                          user=os.getenv('USER'),
                                          password=os.getenv('PASSWORD'),
                                          database=os.getenv('DEFAULT_DATABASE'))

            connection.autocommit = True
            cursor = connection.cursor()

            # create a new database
            cursor.execute(f"create database {os.getenv('DATABASE')};")

        # If database already exist means skip the process
        except psycopg2.errors.DuplicateDatabase:
            pass

        except Exception as e:
            add_vertical_space(2)
                        st.markdown(f'<h5  style="text-position:center;color:orange;">{e}</h5>',
unsafe_allow_html=True)

        finally:
            # close the connection
            if connection:
                cursor.close()
                connection.close()

    def main():
          sql.create_database()

streamlit_config()

with st.sidebar:
    add_vertical_space(4)
    option = option_menu(menu_title='',
                         options=['Admin LogIn', 'Teacher LogIn', 'Assistant LogIn', 'Supersub
LogIn', 'Student LogIn', 'Exit'],
                         icons=['database-fill', 'globe', 'bar-chart-line', 'list-task', 'slash-square',
'sign-turn-right-fill'])


if option == 'Admin LogIn':
    sql.main()
```

```
elif option == 'Teacher LogIn':

    pass

elif option == 'Student LogIn':

    pass

elif option == 'Supersub LogIn':

    pass

elif option == 'Assistant LogIn':

    pass

elif option == 'Exit':

    add_vertical_space(1)
    pass
```