

Problem Statement & Requirements:

3.1 Responsibility of G1:

The G1 takes two feedbacks from the stakeholders, one initial feedback (on or before 8th April 11:59 PM), and then makes relevant changes as suggested per the first feedback, then final feedback (on or before 11th April 11:59 PM) post changes. The write-up/documentation should have screenshots before the first feedback, after the first feedback, and after the second feedback. If a team discusses with multiple stakeholders, please fill out the forms again.

We have taken feedbacks from stakeholders to improve this system. All relevant changes recommended by the stakeholders are implemented in G2: 2.

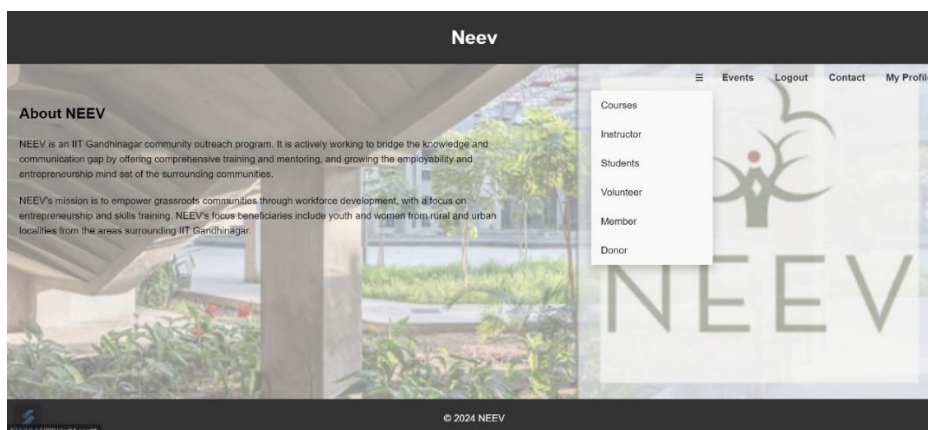
First Feedback

1. To change the order of the tabs at the home screen page of the admin and the member.
2. To add the information of past events and the link to images uploaded on social media.
3. They suggested to add the home/back button on the interface while adding member or student.

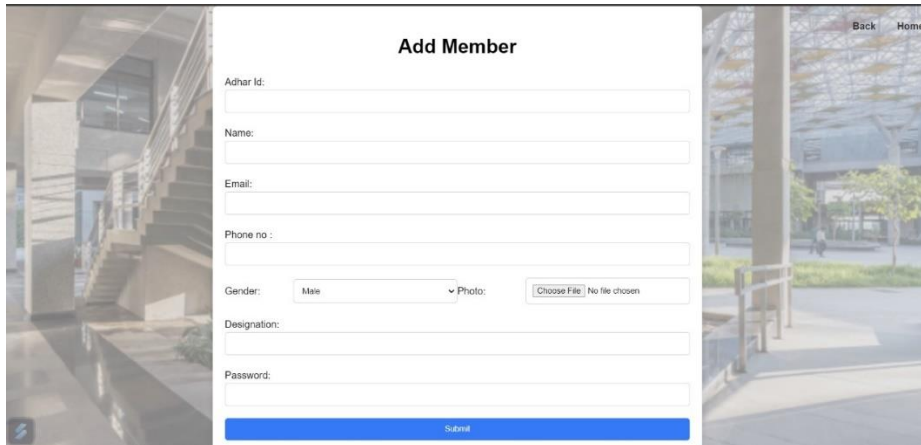
Feedbacks are given by **Soumya Harish**.

2. Attach screenshots of different views [along with a write-up on their privileges] of the database as seen by different classes of users.

➤ Admin view:



- Only Admin can make changes in the donor and the members.
- Admin has the total access whom to add and whom to remove and update.



Add Member

Adhar Id:

Name:

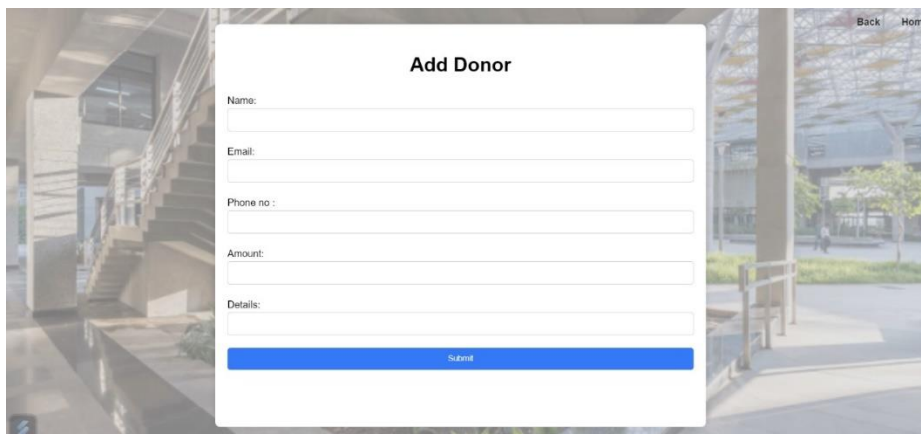
Email:

Phone no :

Gender: Photo: No file chosen

Designation:

Password:



Add Donor

Name:

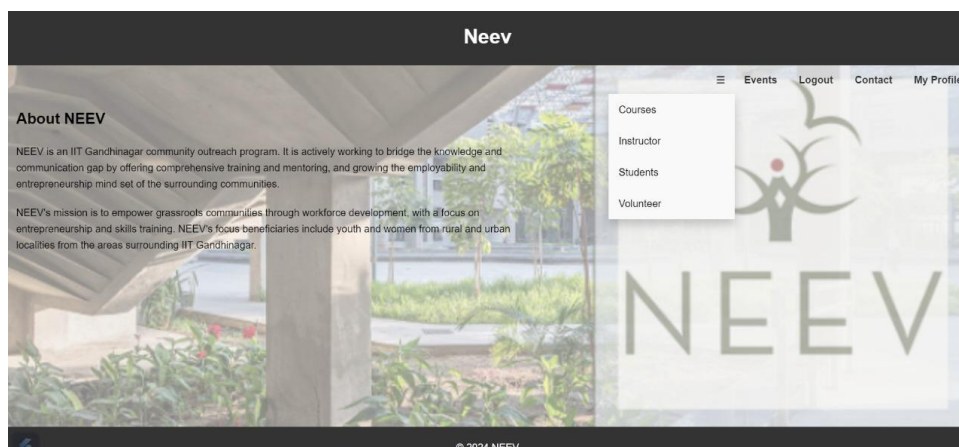
Email:

Phone no :

Amount:

Details:

➤ Neev members view:



- Neev members can add only the courses Instructor Students and Admin.

3.2 Responsibility of G2:

1. Concurrent multi-user access: Multiple users with different roles can access and update the database concurrently. In such a scenario, the same item cannot be updated by two different users. For example, locks can be applied to tables in MySQL.

To acquire a lock in Flask in MySQL, we have two locks:

Share lock for reading:

```
@app.route('/students')
def students():
    if isSessionSet():
        con=mysql_app.connect()
        cur=con.cursor()
        cur.execute("select * from students LOCK IN SHARE MODE")
        data=cur.fetchall()

        cur.close()
        con.close()
        return render_template('students.html',data=data,path=app.config['UPLOAD_FOLDER'])
    else:
        return redirect(url_for('home'))
```

- By adding "LOCK IN SHARE MODE" in select query we can acquire a shared lock and to release a lock we execute "Commit" statement that releases lock acquired by the current transaction.
- Exclusive lock for reading and writing:

```
1 @app.route('/volunteers')
2 def volunteers():
3     if isSessionSet():
4         con = mysql_app.connect()
5         id_1 = session['v_id']
6         cur = con.cursor()
7         cur.execute("BEGIN")
8         cur.execute("select * from volunteer WHERE 1=0 FOR UPDATE")
9         cur.execute("commit")
10        data = cur.fetchall()
11
12        cur.close()
13        con.close()
14        return render_template('volunteer.html', data=data, path=app.config['UPLOAD_FOLDER'])
15    else:
16        return redirect(url_for('home'))
17
```

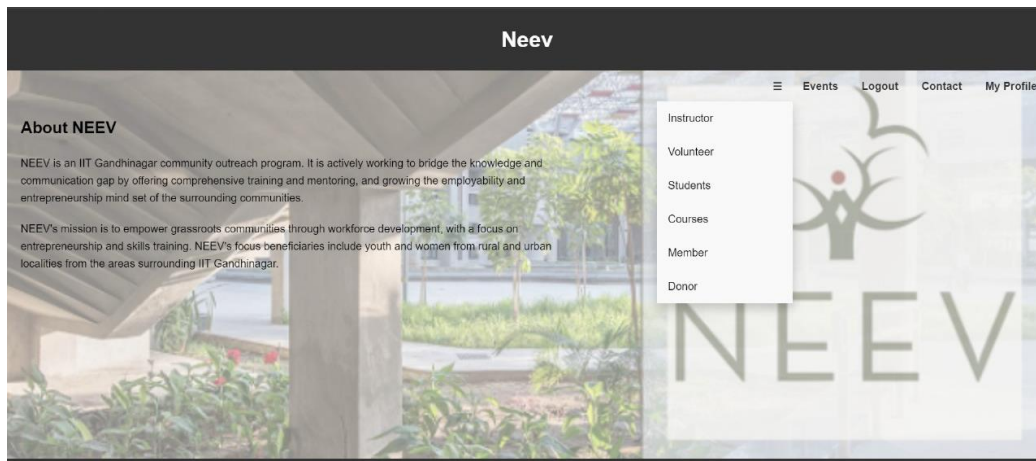
- By using "FOR UPDATE" in select query we can acquire the Exclusive lock and can read and write to that table and to release a lock we execute "Commit" statement that releases lock acquired by the current transaction.

2. Implement the changes in the database as per the feedback received from stakeholders.

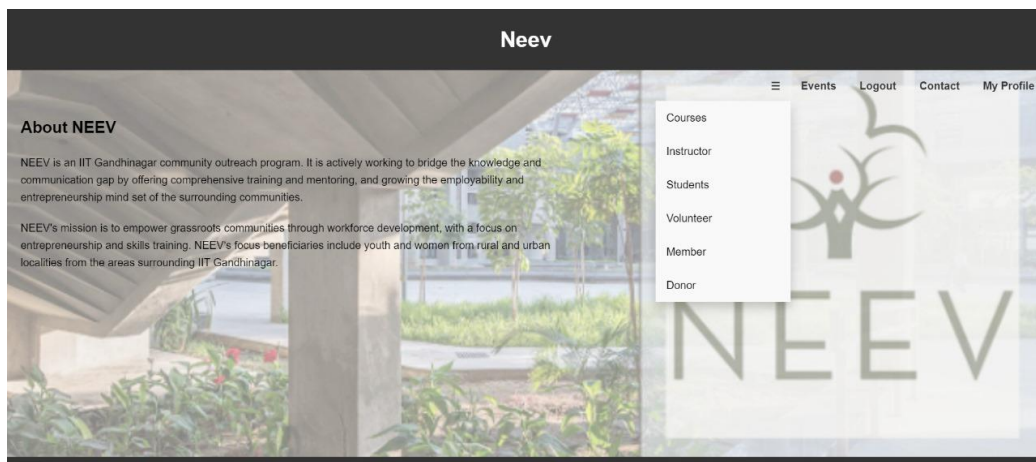
Initial feedback: Recommendation 1

To change the order of the tabs at the home screen page of the admin and the member.

Before:



After:



As we can see the option on the left side of the events helps to switch between the tabs of different users.

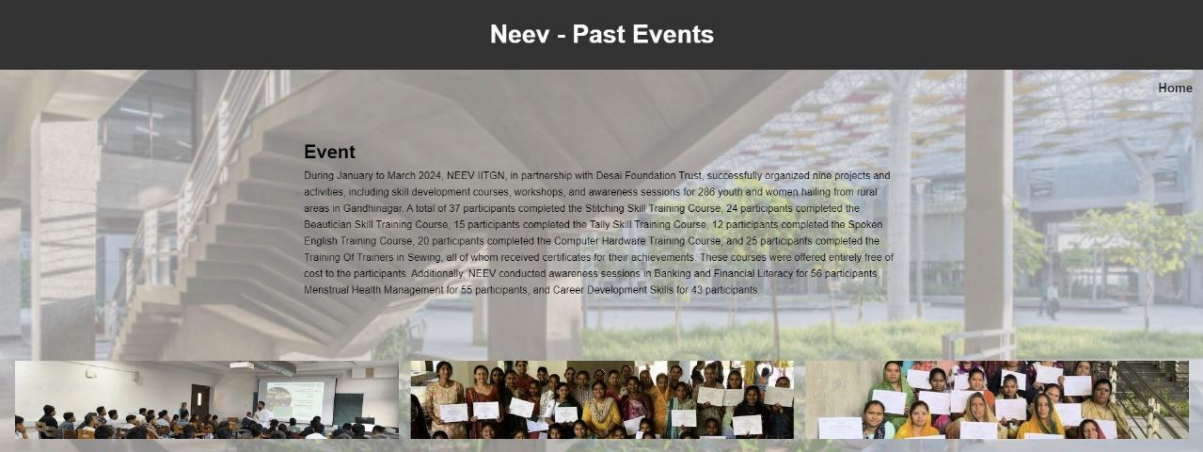
Stakeholder gave us the serial order of that so made changes according to the stakeholder.

Initial Feedback: Recommendation 1 is fulfilled!

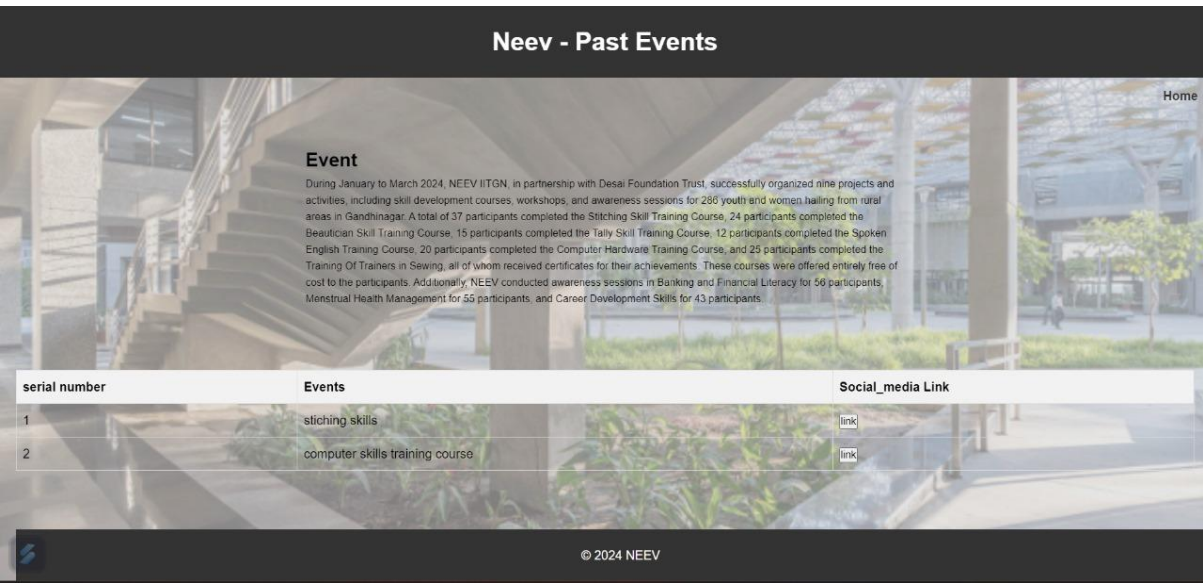
Initial feedback: Recommendation 2

To add the information of past events and the link to images uploaded on social media.

Before:



After:



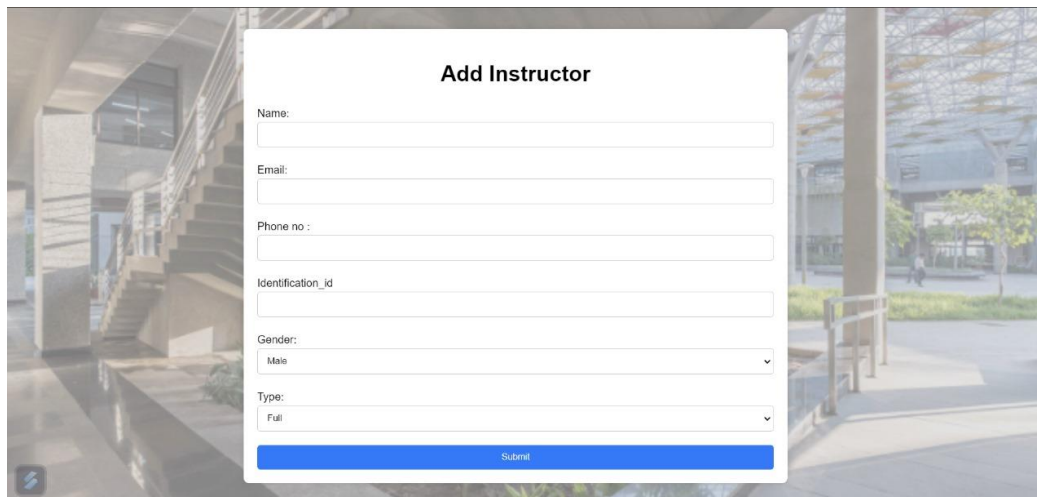
- Stakeholder asked us to remove the images from home instead of that add the table that shows the events name and asked to provide link that directs to the Social Media post of that particular event.
- These are the dummy variable that are added to demonstrate the interface.

Initial Feedback: Recommendation 2 is fulfilled!!

Initial feedback: Recommendation 3

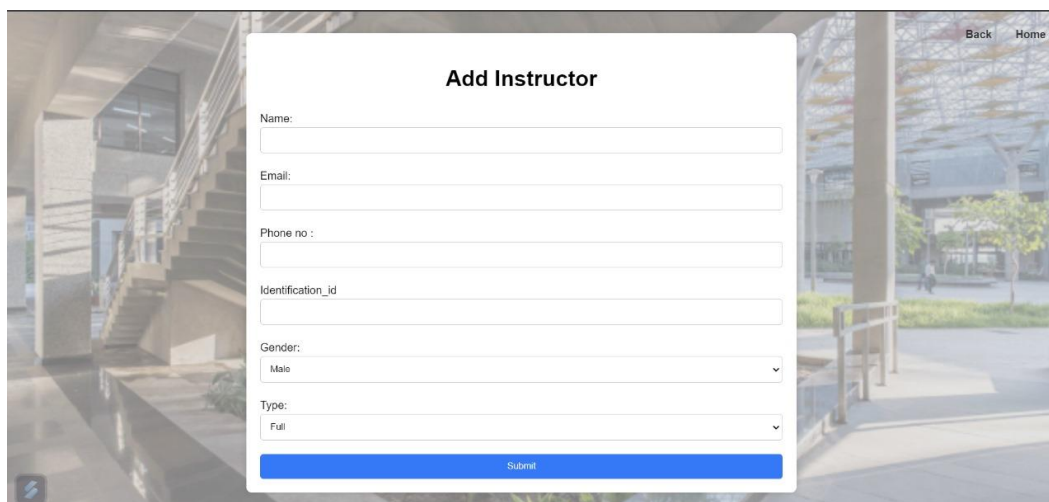
They suggested to add the home/back button on the interface while adding member or student.

Before:



The screenshot shows a web form titled "Add Instructor" overlaid on a background image of a modern building interior with a staircase. The form contains the following fields: "Name:" (text input), "Email:" (text input), "Phone no : " (text input), "Identification_id" (text input), "Gender:" (dropdown menu with "Male" selected), and "Type:" (dropdown menu with "Full" selected). A blue "Submit" button is at the bottom of the form. There are no navigation buttons in the top right corner.

After:

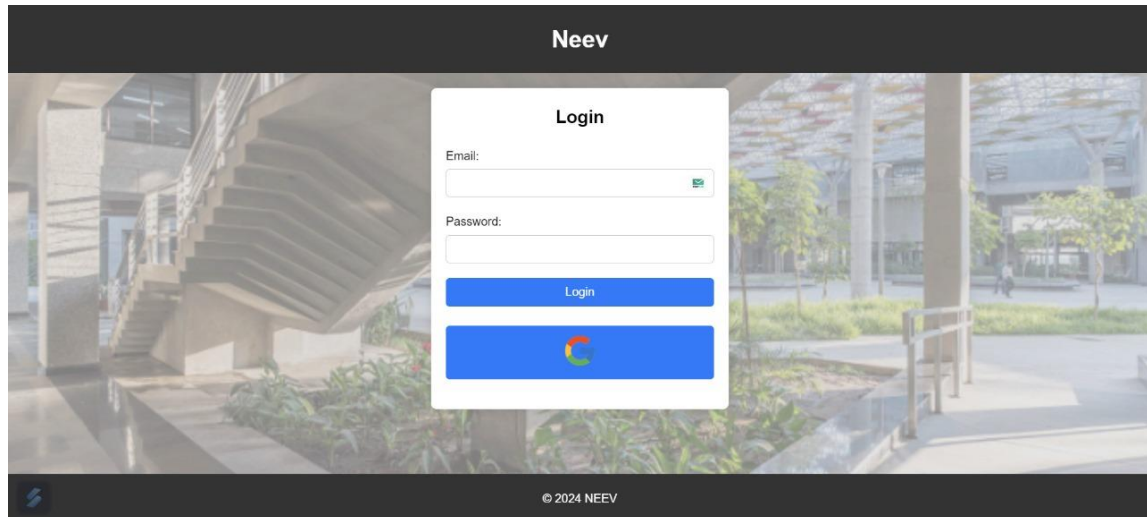


The screenshot shows the same "Add Instructor" form as before, but with two additional buttons in the top right corner: "Back" and "Home". The form fields and the "Submit" button remain the same.

- Stakeholder was getting problem whenever she wanted to add student if she wanted to go back to home or back she didn't know that.
- So, we added a home and a back button on the upper right corner to cancel in between if she wanted to cancel the process in between.

Initial Feedback: Recommendation 3 is fulfilled!!

1. Add google authentication for login and registration. (only IITGN user can login and register) **(10 Points)**



Actually, this webapp is only for the NEEV members they have their own email so we need not add people who are from the IIT GN.

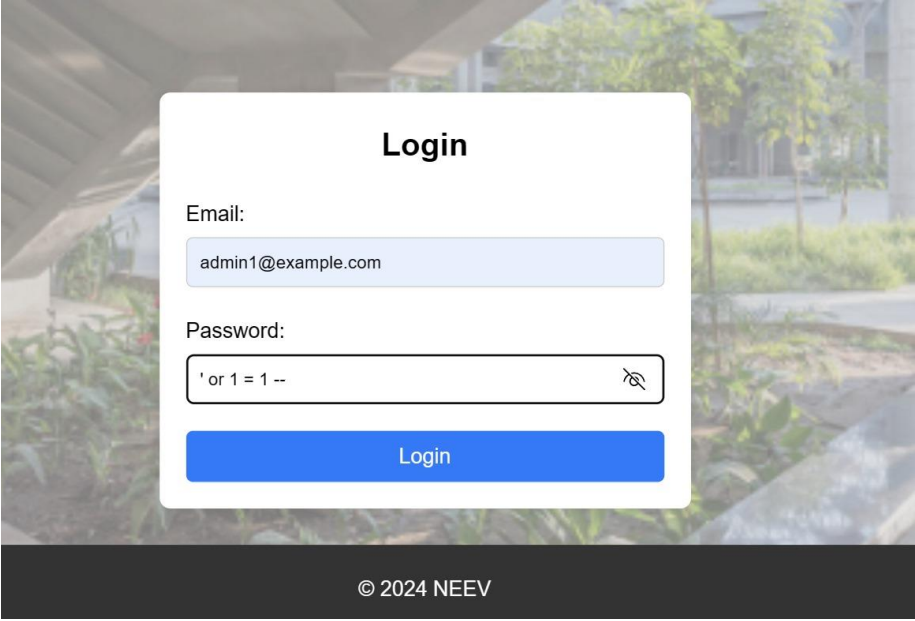
Because only NEEV Members and Admin can login into the webapp, no IIT GN student is involved in the login.

And students who come to learn these courses are from small villages out of IIT GN.

3.3 Responsibility of G1 & G2:

1. Documentation and screenshots of a total of 2 attacks [SQL Injection and XSS] performed and the defenses against those attacks.

- SQL Injection:



Login

Email:

admin1@example.com

Password:

' or 1 = 1 --

Login

© 2024 NEEV

```
@app.route('/login_auth', methods=['POST'])
def login_auth():
    if request.method == 'POST':
        con = mysql_app.connect()
        cur = con.cursor()
        cur.execute("select * from admin where Email=%s and Password=%s",
                    (request.form['email'], request.form['password']))
        data_admin = cur.fetchall()

        cur.execute("select * from neev_members where Email=%s and Password=%s",
                    (request.form['email'], request.form['password']))
        data_member = cur.fetchall()

        cur.close()
        con.close()
```

While, design the backend we already asked to given input of the password in string.

To perform SQL Injections we have to give “ ' or 1=1 ” as a user name so the or condition becomes true.

Because of string input adversary can't perform the SQL Injection on the website.

- XSS:

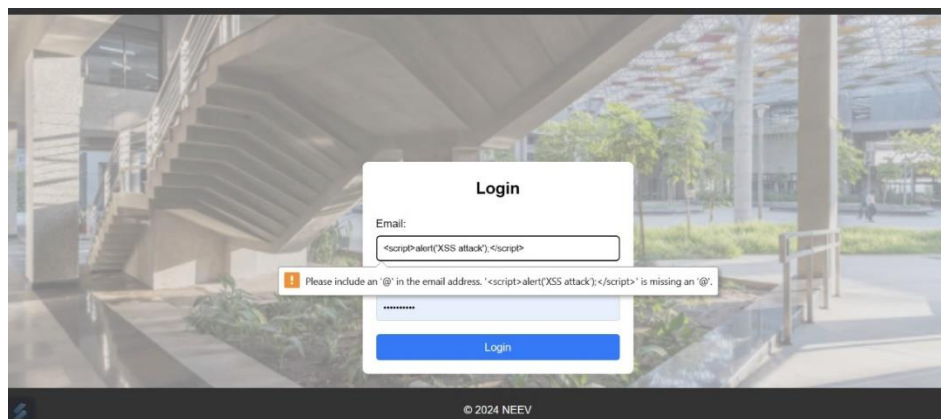

```

@app.route('/login_auth', methods=['POST'])
def login_auth():
    if request.method == 'POST':
        con = mysql_app.connect()
        cur = con.cursor()
        cur.execute("select * from admin where Email=%s and Password=%s",
                    (request.form['email'], request.form['password']))
        data_admin = cur.fetchall()

        cur.execute("select * from neev_members where Email=%s and Password=%s",
                    (request.form['email'], request.form['password']))
        data_member = cur.fetchall()

        cur.close()
        con.close()

```



Similarly, in the XSS attack we try to run the javascript file that contains malicious code.

Similarly, we take the input as string instead of directly checking the input on into the database. That causes the XSS attack.

Also, we provided the input username to contain @ while login and the input should be taken as string

- So, the system is secured from both the attacks XSS and SQL Injections.

2. Show that all the relations and their constraints, finalized after the second feedback, are present and valid as per the ER diagram constructed in Assignment 1.

Relationship	Cardinality	participation	Justification
Admin – Member	One - many	Partial – Total	As one admin can add multiple members. all the admin will not be adding the members but all members are only be added by the admin.
Member – Student	One - many	Partial – Total	As one member can add multiple students. all the members will not be adding the students but all students are only be added by the member.
Member – Volunteer	One - many	Partial – Total	As one member can add multiple volunteer. all the members will not be adding the volunteer but all volunteer are only be added by the member.

Relationship	Cardinality	participation	Justification
Student – Instructor	One - One	Partial – partial	As student can also become the instructor when type of instructor is set as s and sid will be stored in instructor relation.
Member – Instructor	One - One	Partial – partial	As neev member can also become the instructor when type of instructor is set as m and mid will be stored in instructor relation.

Admin – Donor	One - many	Partial – Total	Admin will add the details of the funds.
---------------	------------	-----------------	--

Relational Schemas derived from the relationship sets:

1. Student course: (S_ID, C_ID)

Student (Many - One) Course(One -many)

2. Volunteer course: (V_ID, C_ID)

Volunteer (Many- One) course(One -many)

3. Funds:(D_ID, AD_ID)

Donation (Many - One) Admin (One-One)

4. Active courses:(C_ID, I_ID)

Course (One - many) Instructor (One - many)

Hence, all the relations and their constraints are present in the are finalized.

Contribution:

G1: Himani Trivedi, Priya Darji

G2: Om Ambekar, Diya Parashar, Dhru Jain

G1 & G2: Himani Trivedi, Priya Darji, Om Ambekar, Diya Parashar, Dhru Jain