Problem Statement & Requirements:

Based on the teams formed, each team is supposed to create a web application of the proposed functionalities in the previous assignments. Each team can be divided into Group G1 and Group G2; G1 is supposed to work on the front-end and G2 on the back-end sections, respectively. You can use Flask + MySQL for developing your WebApp.

The web app should support the dynamic execution (The changes done by the user through the web app shall be reflected in the main database and web page also) of the following functions on your database:

- INSERT
- UPDATE
- DELETE
- RENAME
- WHERE clause

Push it to GitHub and share the link for submission.

Please note that all the constraints mentioned in Assignment 1 and 2 should be implemented and functioning in your application.

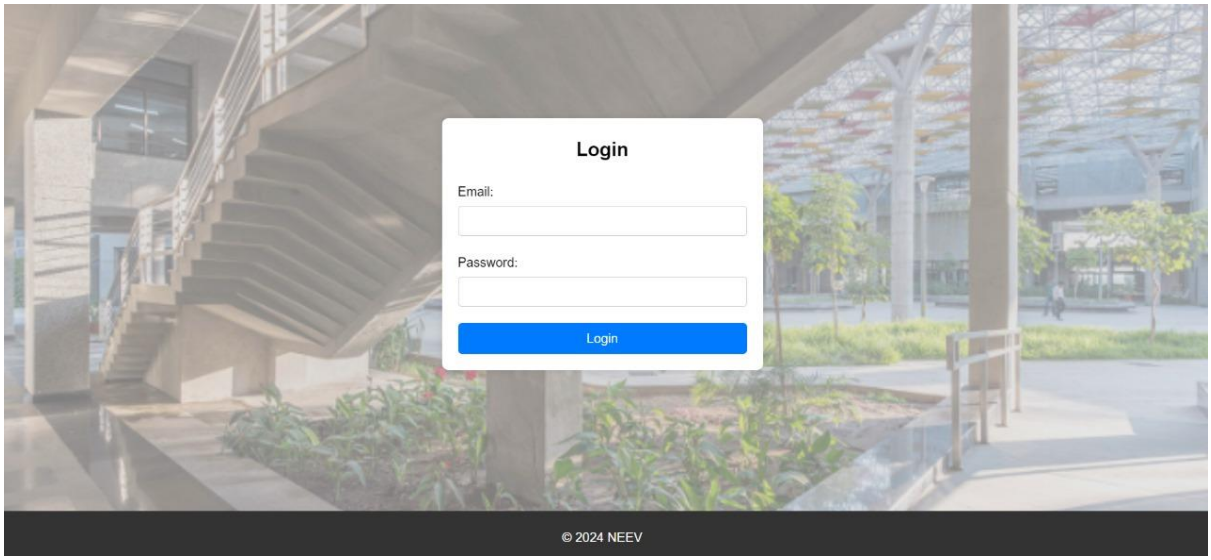
Login page with authentication of users and stakeholders.

3.1 Responsibility of G1:

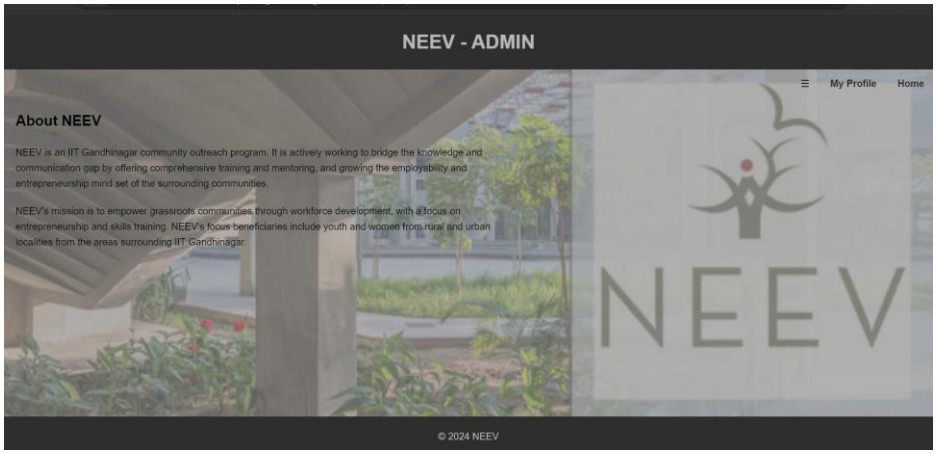
G1 is supposed to design the front end and beautify accordingly using HTML/CSS/JS, you may use any other libraries for the front end, such as Bootstrap or jQuery. But flask is mandatory.

Screenshots:

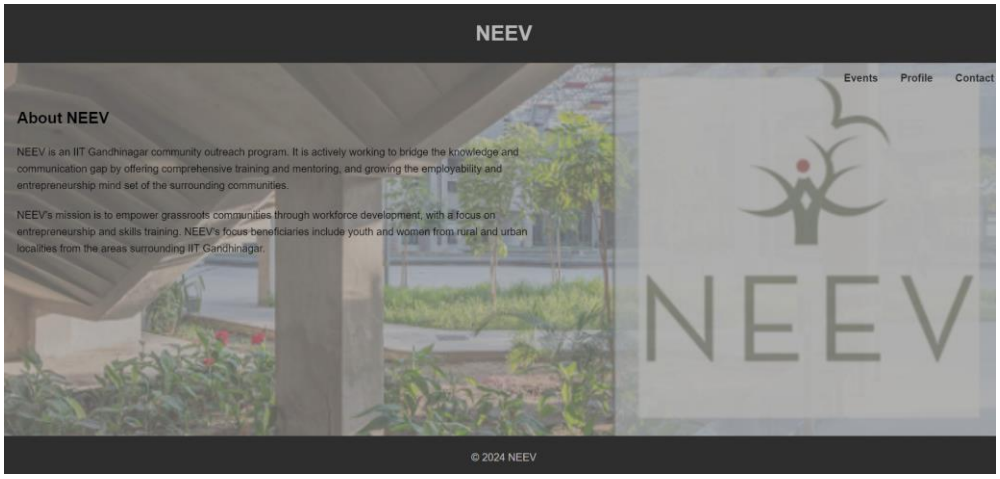
- **Login**: Login intreface



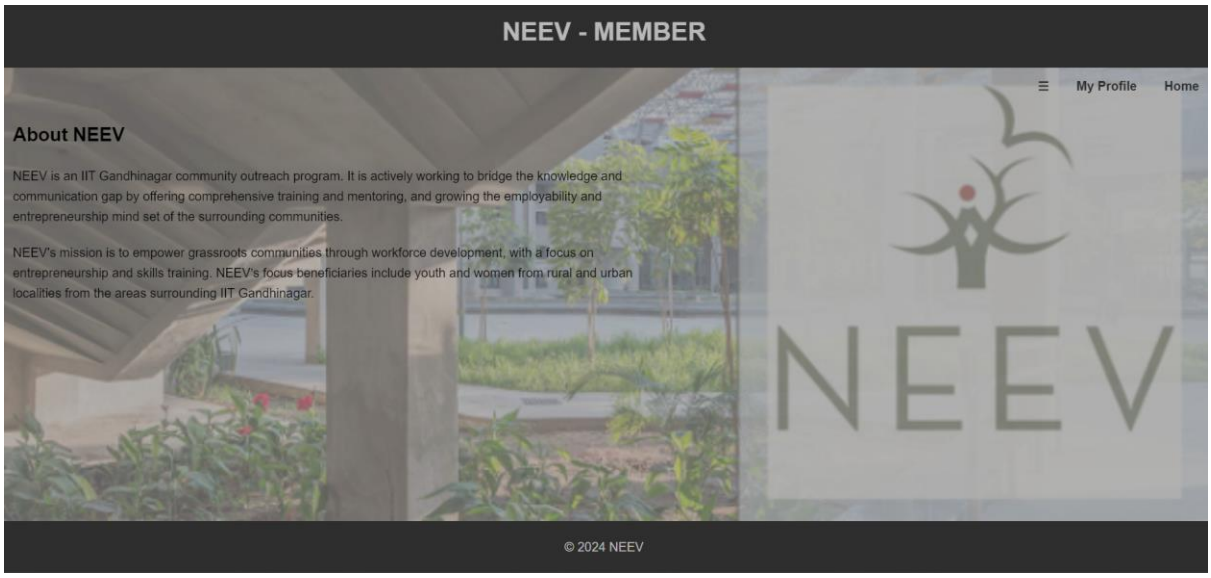
- **Admin Dashboard:** The front page for the Admin.



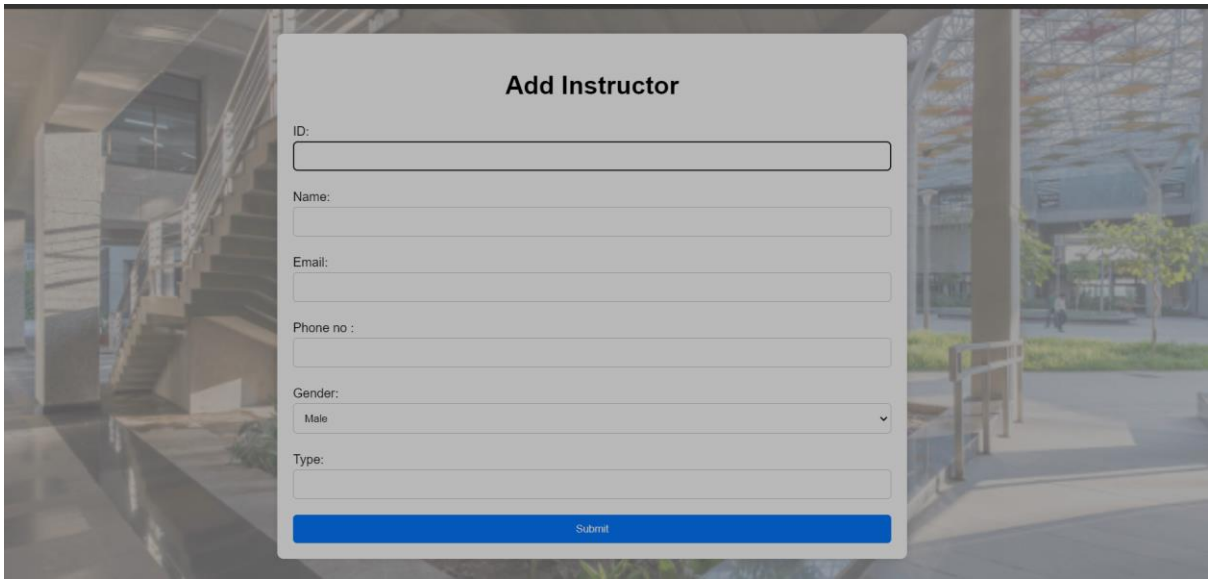
Main Dashboard: The main page for NEEV members.



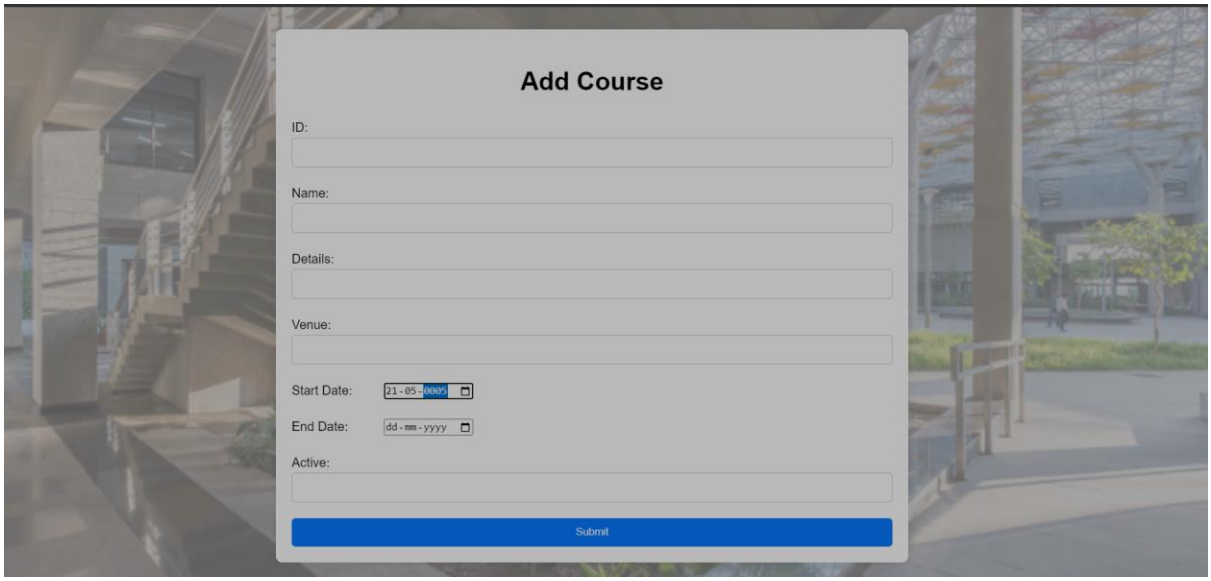
- **Neev_members Dashboard: This page is displayed for the neev members**



- **Add Instructor page**: Admin can add Instructor using the information below.

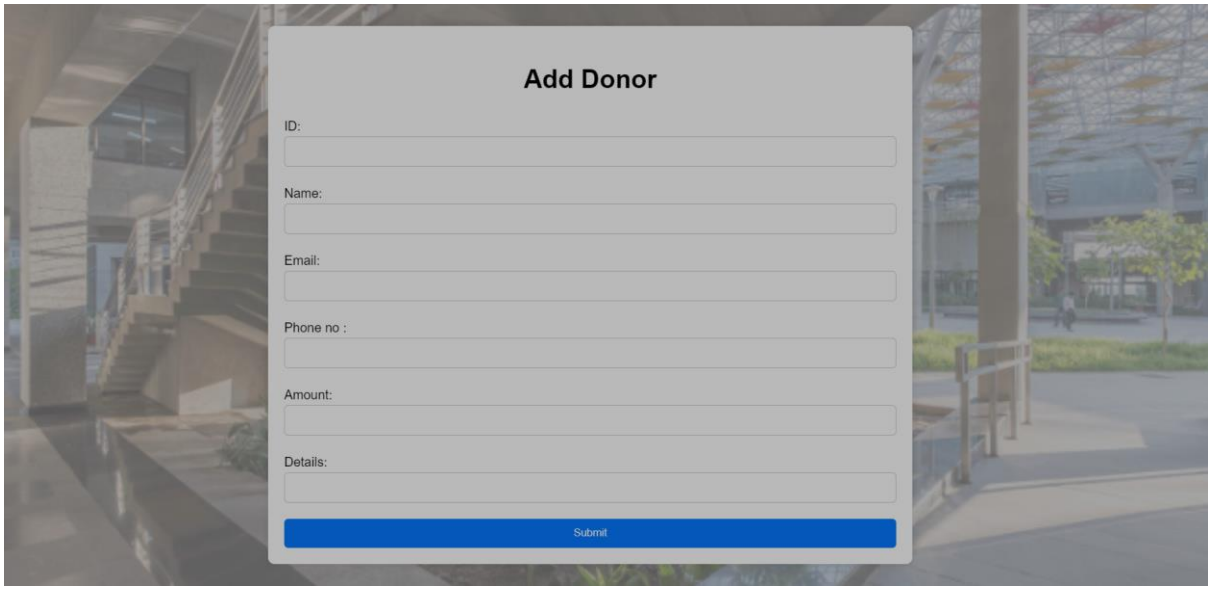


- **Add Course page**: Admin can add the course using following information below.

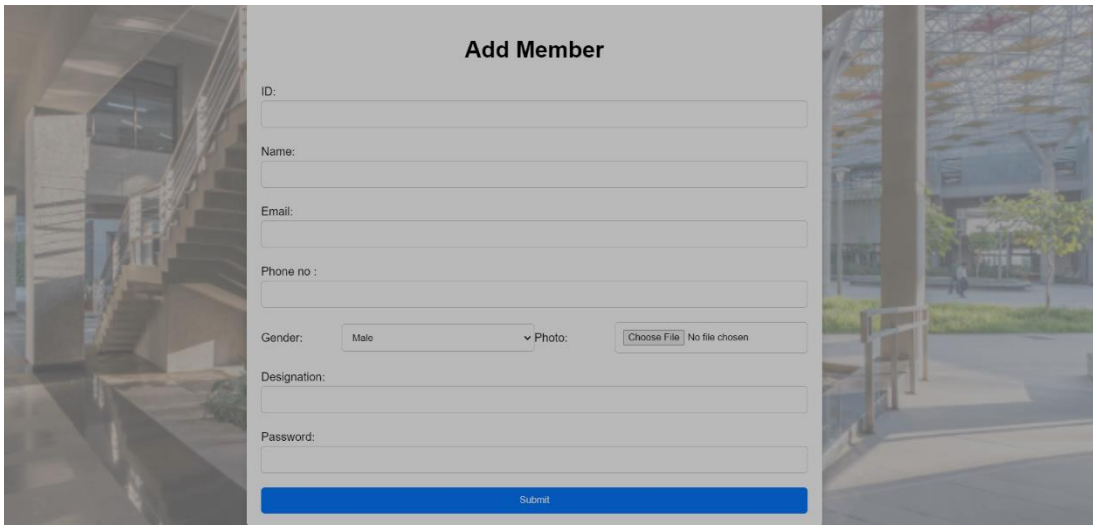


- **Add Voluntteer**: Admin can add volunteer using below information.

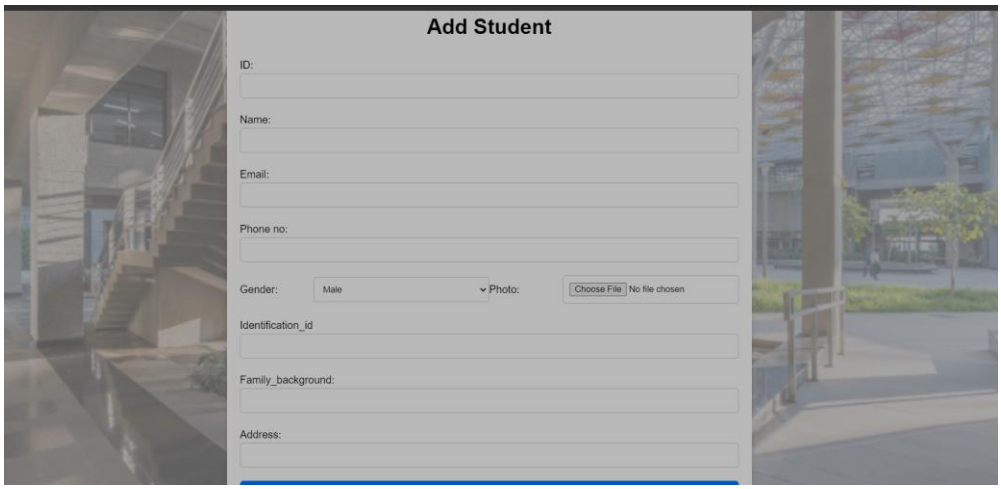
- **Add Voluntteer**: Admin can add donor using below information.



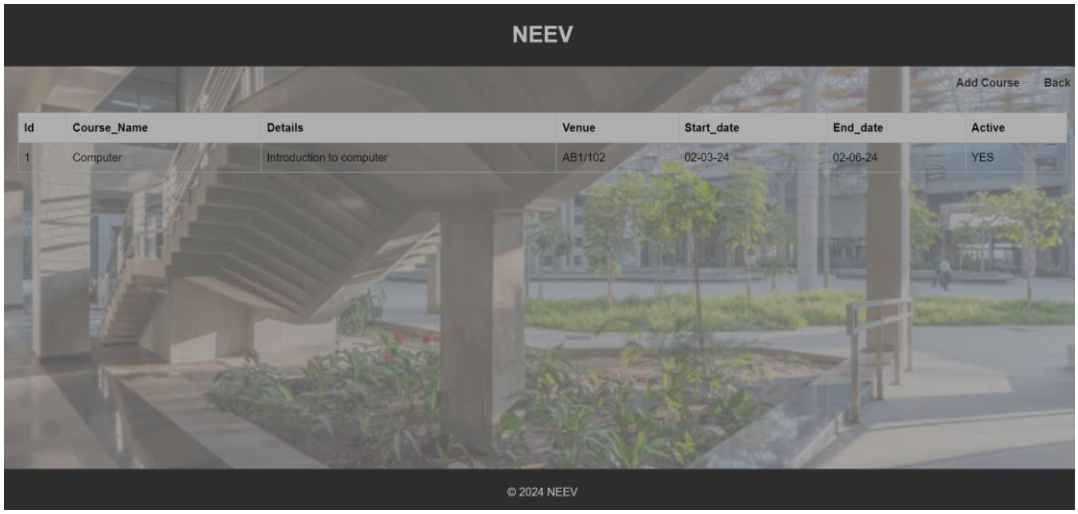
- **Add Neev member**: Admin can add the course using following information below.



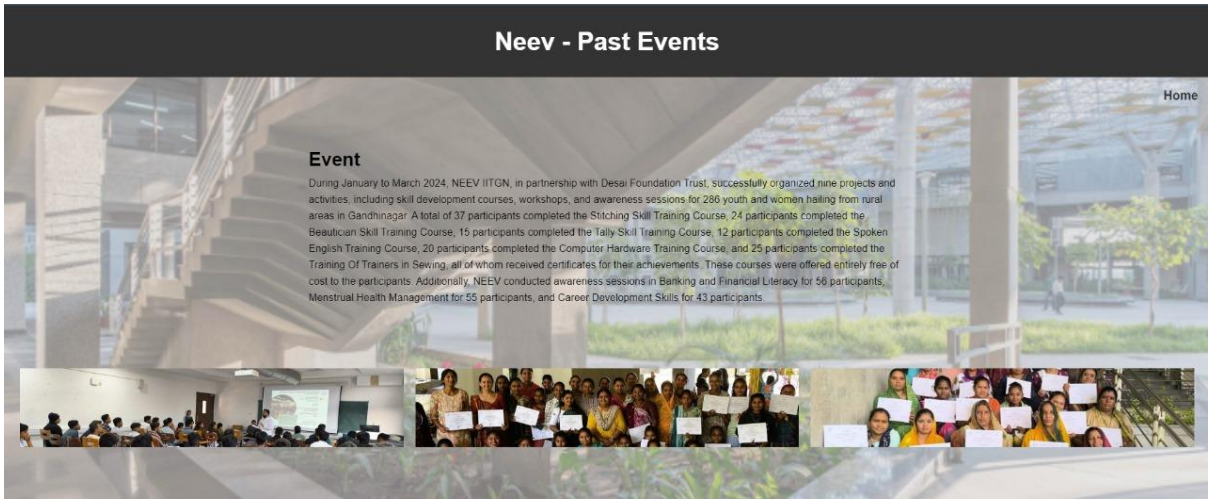
- **Add Student:** members of Neev can add the student using following information below.



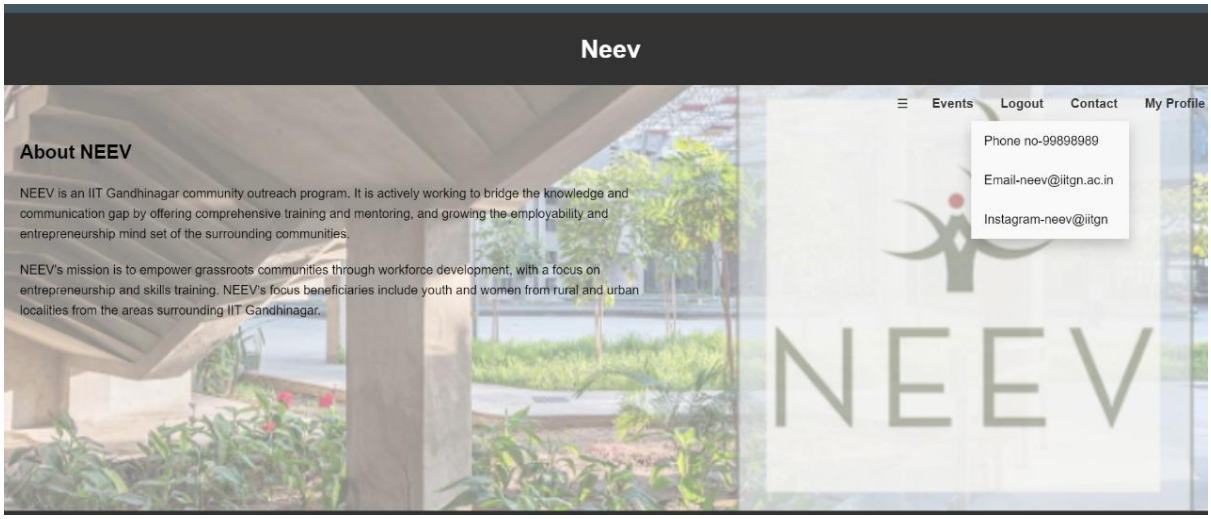
- Courses: Below interfaces shows the list of courses in NEEV. The Column ‘Active’ shows the active course is the course available in specific time period



- The page too see the past events.



- To know the contact details



Similarly, the page for Neev members is present. But the neev members can add Volunteer, Students, Courses, Instructor. Also, neev members cannot add Donor and Any other neev member.

Responsibility of G2:

G2 has to work on integrating MySQL in WebApp. This group works on the proper working of the backend.

- **Query for Login Authentication, for both admin side and member side**.

```
@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()

        if len(data_member) == 0 and len(data_admin) == 0:
            return render_template('login.html',error=1)
        elif len(data_admin) == 1:
            session['member_type']='admin'
        else:
            session['member_type']='member'

        global session_Set
        session_Set=1
        session['set']=1
        session['email']=request.form['email']
        session['password']=request.form['password']
        return redirect(url_for('dashboard',session=session))  #1=> member and 2=>admin
    else:
        return render_template('home.html')
```

- **Query to display profile**

```
@app.route('/profile')
def display_profile():
    if isSessionSet():
        con=mysql_app.connect()
        cur=con.cursor()
        if  session['member_type'] == 'admin':
            cur.execute("select * from admin where Email=%s and Password=%s",(session['email'],session['password']))
        else:
            cur.execute("select * from neev_members where Email=%s and Password=%s",(session['email'],session['password']))
        data=cur.fetchone()
        cur.close()
        con.close()
    else :
        return redirect(url_for('home'))

    return render_template('profile.html',data=data,type=session['member_type'])
```

- **Query to activate course:**

```
def activate_course(id):
    if isSessionSet():
        con=mysql_app.connect()
        cur=con.cursor()
        cur.execute("select * from course where C_ID=%s",id)
        data=cur.fetchall()

        cur.execute("select * from instructors")
        instructors=cur.fetchall()

        cur.close()
        con.close()
        current_date = date.today().isoformat()

        return render_template('activate_course.html',data=data[0],ins=instructors,date=current_date)
    else:
        return redirect(url_for('home'))

@app.route('/activate_course/done_activate',methods=['POST'])
def done_activate():
    if request.method == 'POST':
        con=mysql_app.connect()
        cur=con.cursor()

        cur.execute("INSERT INTO active_courses (I_ID, C_ID, start_date, end_date) VALUES (%s,%s,%s,%s)",(request.form['instructor'],request.form['courseId'],reque
        con.commit()

        cur.close()
        con.close()
        return redirect(url_for('courses'))
```

```
            con.close()
            return redirect(url_for('courses'))
        else:
            return redirect(url_for('home'))

@app.route('/update/<id>')
def update_link(id):
    if isSessionSet():
        con=mysql_app.connect()
        cur=con.cursor()
        cur.execute("select * from course where C_ID=%s",(id))
        course_data=cur.fetchall()
        cur.close()
        con.close()
        return render_template('update_course.html',data=course_data)
    else:
        return redirect(url_for('home'))

@app.route('/update_course',methods=['POST'])
```

- **Query to update the link:**

```
@app.route('/update/<id>')
def update_link(id):
    if isSessionSet():
        con=mysql_app.connect()
        cur=con.cursor()
        cur.execute("select * from course where C_ID=%s",(id))
        course_data=cur.fetchall()
        cur.close()
        con.close()
        return render_template('update_course.html',data=course_data)
    else:
        return redirect(url_for('home'))

```

- **Query to add course details:**

```
@app.route('/course_details/<id>')
def course_details(id):
    if isSessionSet():
        con=mysql_app.connect()
        cur=con.cursor()
        cur.execute("select * from course where C_ID=%s",(id))
        course_data=cur.fetchall()

        cur.execute("select name from active_courses natural join instructors where C_ID=%s",(id))
        inst=cur.fetchall()
        inst=tuple(item for subtuple in inst for item in subtuple)

        cur.execute("select * from active_courses where C_ID=%s",(id))
        active_part=cur.fetchall()

        stu=[]
        stu_len=[]
        vol=[]
        vol_len=[]
        all_stu=[]
        all_vol=[]
        for i in active_part:
            cur.execute("select s_id,name,email  from students where s_id in (select s_id from active_courses NATURAL join student_course where active_id=%s)",(i[4
            rel=cur.fetchall()
            stu.append(rel)
            stu_len.append(len(rel))

            cur.execute("select v_id,name  from volunteer where v_id in ( select v_id  from active_courses NATURAL join volunteer_course where active_id=%s)",(i[4]
            rel_vol=cur.fetchall()
            vol.append(rel_vol)
            vol_len.append(len(rel_vol))

            cur.execute("select S_ID,name from students where S_ID NOT IN (select S_ID from active_courses NATURAL join student_course where active_id=%s)",(i[4]
            all_stu.append(cur.fetchall())

            cur.execute("select v_id,name,email from volunteer where v_id not in ( select v_id  from active_courses NATURAL join volunteer_course where active_id
            all_vol.append(cur.fetchall())

        cur.close()
        con.close()

        return render_template('course_details.html',course=course_data, instructors=inst, stud_not_in_course=all_stu,
        vol_not_in_course=all_vol, active_course=active_part,students=stu,stu_count=stu_len,volunteers=vol,vol_count=vol_len)
    else:
        return redirect(url_for('home'))
```

- **Query to add volunteer:**

```
@app.route('/update_volunteer',methods=['POST','GET'])
def update_volunteer():
    if request.method == 'POST':
        con=mysql_app.connect()
        cur=con.cursor()

        sql="UPDATE volunteer SET Name=%s, Email=%s,Identification_ID=%s,Gender=%s,Address=%s,Photo=%s,Phone_No=%s WHERE v_id=%s"


        if request.files['photo'].filename == '':
            sql_2="UPDATE volunteer SET Name=%s, Email=%s,Identification_ID=%s,Gender=%s, Address=%s,Phone_No=%s WHERE M_id=%s"

            cur.execute(sql_2,(request.form['name'],request.form['email'],
            request.form['adhar_id'],request.form['gender'],request.form['designation'],
            request.form['phone'],request.form['v_id']))
            con.commit()
            app.logger.info("nothing")
        else:
            file = request.files['photo']
            filename= file.filename.rsplit('.', 1)[0].lower() + str(int(random.random() * 4500)) + '.' + \
            file.filename.rsplit('.', 1)[1].lower()
            filepath=os.path.join(app.config['UPLOAD_FOLDER'],filename)

            if file and allowed_file(file.filename):
                    file.save(filepath)
            else:
```

```
                    file.save(filepath)
            else:
                return render_template('update_member.html',error=2,path=app.config['UPLOAD_FOLDER'])

            cur.execute(sql,(request.form['name'],request.form['email'],request.form['adhar_id'],request.form['gender'],request.form['designation'],filename,reques
            con.commit()
            app.logger.info("else")

        cur.close()
        con.close()
        return redirect(url_for('volunteers'))
    else:
        return redirect(url_for('home'))
```

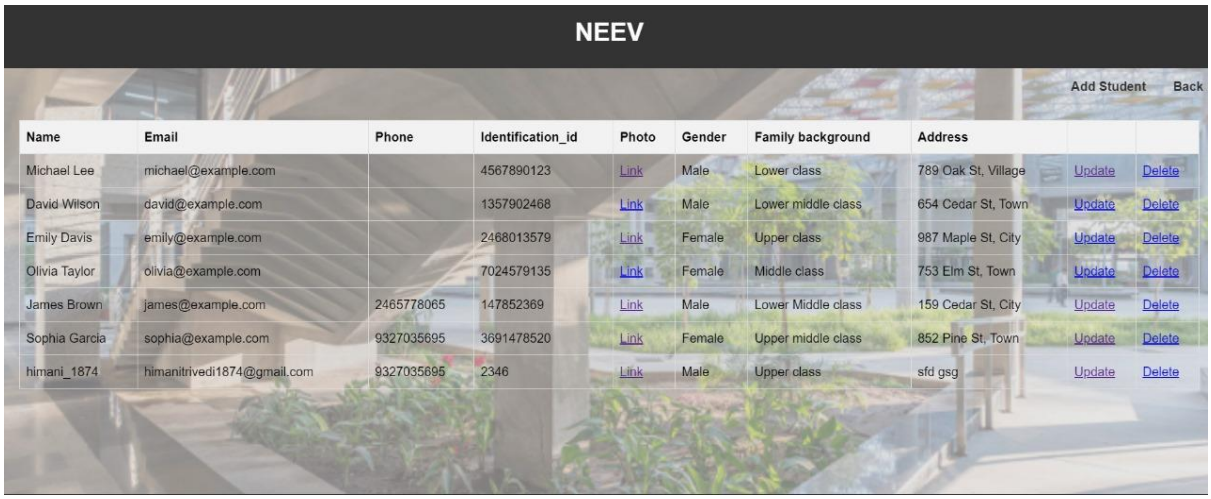
- **Query to remove volunteer Delete volunteer**

```
def delete_volunteer(s_id):
    if isSessionSet():
        sql="DELETE FROM volunteer WHERE v_id=%s"

        con=mysql_app.connect()
        cur=con.cursor()
        cur.execute(sql,s_id)
        con.commit()

        cur.close()
        con.close()

        return redirect(url_for('volunteers'))
    else:
        return redirect(url_for('home'))

@app.route('/add_volunteer_link')
```

For all students, members, volunteer, Instructor. All the queries and same to add and delete.

Responsibility of G1 & G2:

1. Create the database of the designed logical schema in your previous assignment number 2, enter dummy/imaginary values into it, and show these dummy entries on the web page.

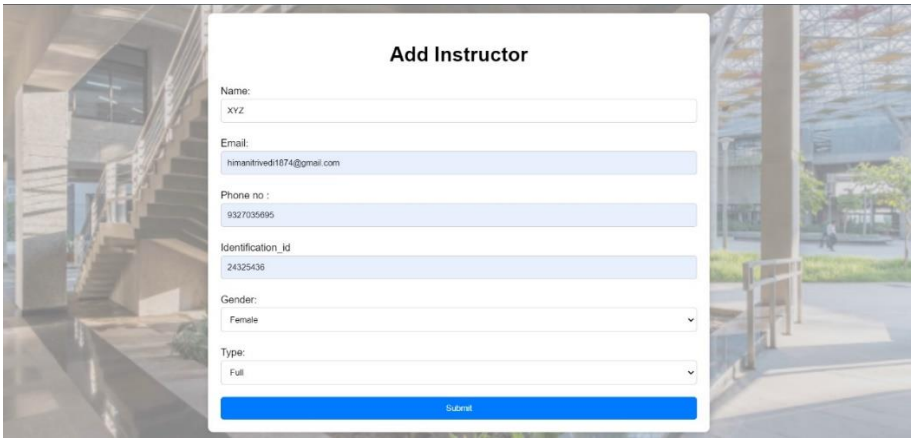


Name	Email	Phone	Identification_id	Photo	Gender	Family background	Address		
Michael Lee	michael@example.com		4567890123	Link	Male	Lower class	789 Oak St, Village	Update	Delete
David Wilson	david@example.com		1357902468	Link	Male	Lower middle class	654 Cedar St, Town	Update	Delete
Emily Davis	emily@example.com		2468013579	Link	Female	Upper class	987 Maple St, City	Update	Delete
Olivia Taylor	olivia@example.com		7024579135	Link	Female	Middle class	753 Elm St, Town	Update	Delete
James Brown	james@example.com	2465778065	147852369	Link	Male	Lower Middle class	159 Cedar St, City	Update	Delete
Sophia Garcia	sophia@example.com	9327035695	3691478520	Link	Female	Upper middle class	852 Pine St, Town	Update	Delete
himani_1874	himanitrivedi1874@gmail.com	9327035695	2346	Link	Male	Upper class	sfd gsg	Update	Delete

The web app should support the dynamic execution (The changes done by the user through the web app shall be reflected in the main database and web page also) of the following functions on your database:

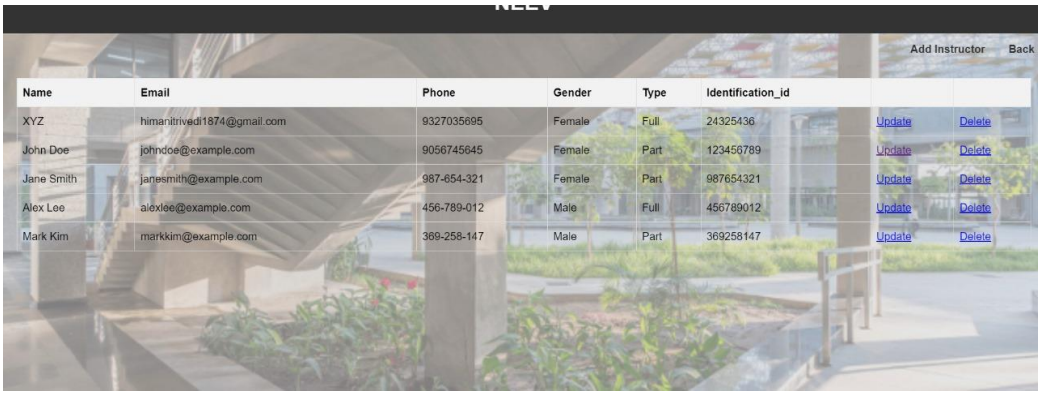
1. INSERT

- **Before Adding:**



- **After Adding:**

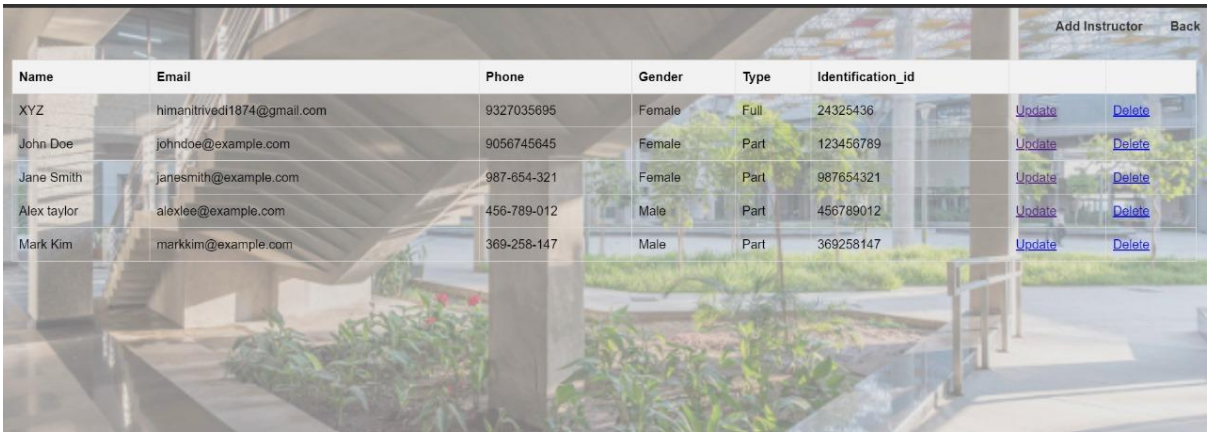
  We are adding the instructor:



2. Update:

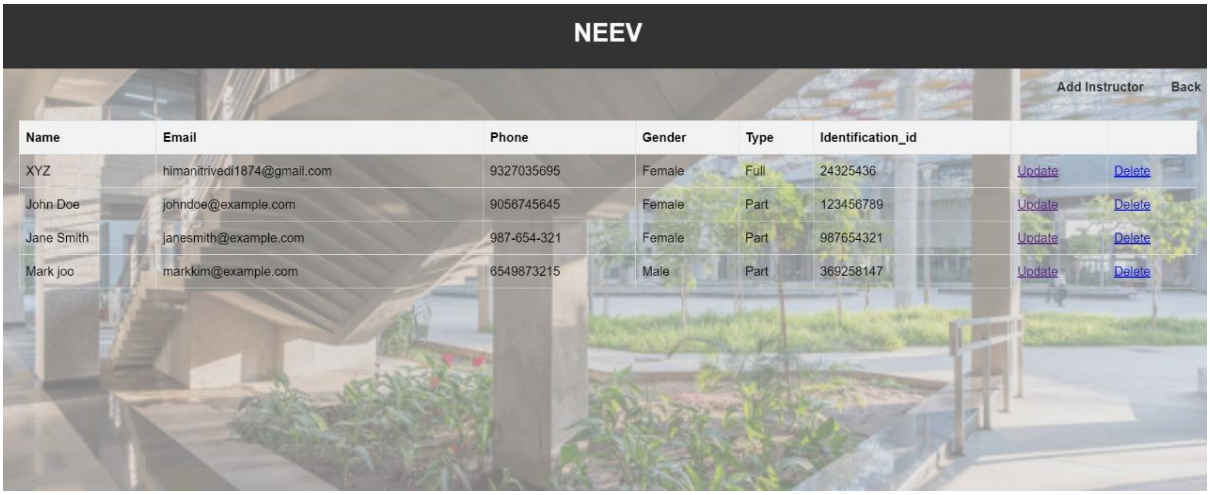
- **Before:**

  Here the surname of Mark is Kim



- **After:**

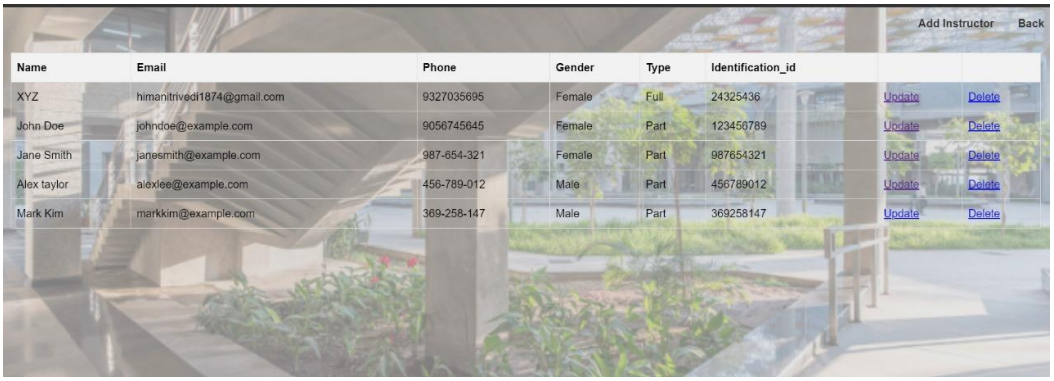
  Updated the surname and mobile.no. of the Mark joo



- **It also shows the image not available if the image is not uploaded**

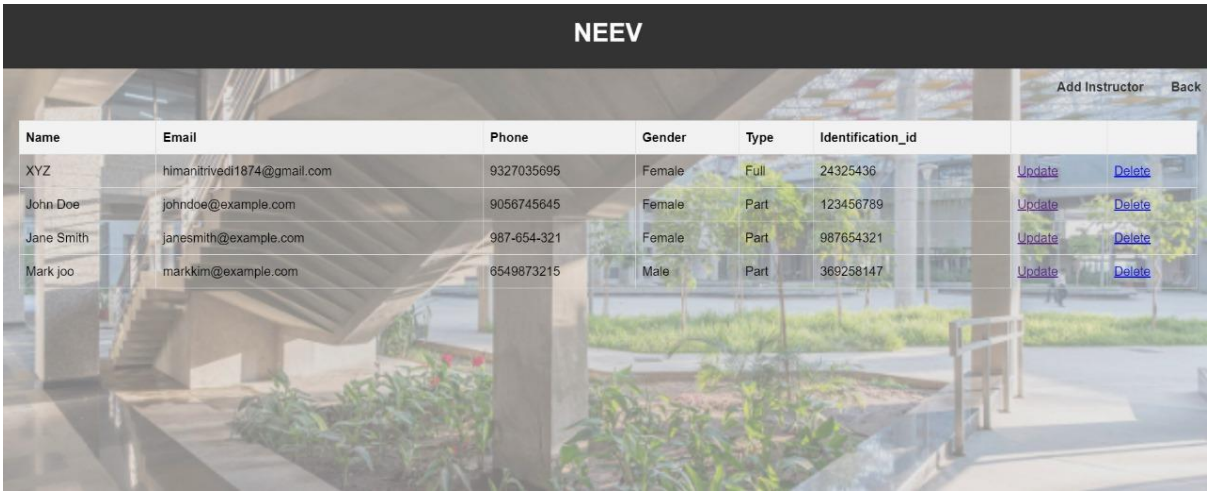
3.Delete:

- **Before**



- **After:**

Deleted the Instructor named Alex Taylor

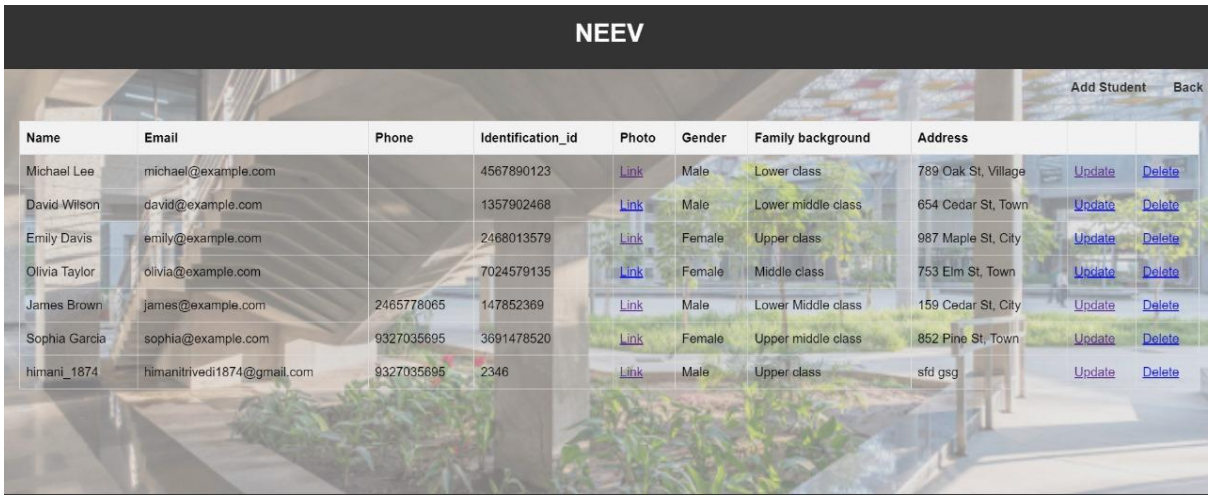


4. Rename:

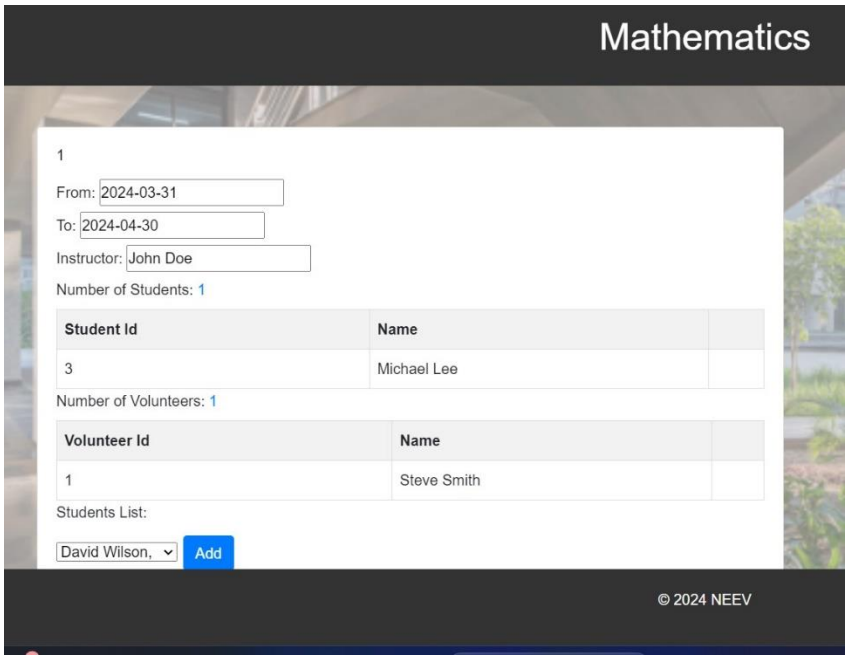
Our system doesn't require any rename operation to do so we haven't implemented/shown it and there is no point to rename table, database or column.

5. WHERE clause:

The Course list:



The Course Detail:



Contribution:

G1 Priya Darji, Om Ambekar, Diya Parashar

G2: Himani Trivedi, Dhru Jain

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

Documentation: Dhru Jain