

Group- Oracles Eleven

Assignment-4

3.1 Responsibility of G1:

1. The G1 takes two feedbacks from the stakeholders, [one initial feedback](#) (on or before 30th March 11:59 PM), and then makes relevant changes as suggested per the first feedback, then [final feedback](#) (on or before 10th 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 (Initial and final feedback forms). **(30 Points)**

Feedback 1 :-

1. Use a single login page. Store the data of all users in a single table and later segregate into student, professor etc.
2. Define the budget and number of students that can be taken.
3. use bootstrap for data tables.
4. operation wise its correct.
5. work on design.
6. Don't keep passwords in the frontend as admins can see them directly, and have access to change them.
7. Do not keep cpi in the frontend as that is personal for every student.
8. Rename the tables name

Before first feedback

- No direct button for the application process for the specific job

| Job ID | Employee Name | Employee Id | Activity | Job desc | Pre requisite | Eligibility | total job | Duration | App status |
|--------|---------------|-------------|-----------|------------------------|---------------------|------------------------|-----------|-----------|------------|
| 1 | John Smith | 1001 | Sales | Sales Manager | Bachelor's degree | 5 years | 3 | 12 months | Open |
| 10 | Amanda Brown | 1010 | Sales | Sales Manager | Bachelor's degree | 5+ years sales | 2 | 12 months | Open |
| 11 | Steven Kim | 1011 | IT | Software Engineer | Bachelor's degree | 3+ years software | 2 | 18 months | Open |
| 12 | Lisa Lee | 1012 | HR | HR Generalist | Bachelor's degree | 2+ years HR exp | 4 | 6 months | Open |
| 13 | Peter Chen | 1013 | Finance | Financial Analyst | Bachelor's degree | 1+ years finance | 5 | 3 months | Open |
| 14 | Jennifer Lee | 1014 | Marketing | Marketing Manager | Master's degree | 5+ years marketing | 1 | 24 months | Closed |
| 15 | Kevin Kim | 1015 | IT | IT Manager | Bachelor's degree | 5+ years IT | 1 | 24 months | Closed |
| 16 | Olivia Park | 1016 | HR | Benefits Specialist | Bachelor's degree | 2+ years HR exp | 3 | 9 months | Open |
| 17 | Thomas Lee | 1017 | Finance | Accountant | Bachelor's degree | 2+ years accounting | 3 | 6 months | Open |
| 18 | Maggie Brown | 1018 | Sales | Sales Associate | High school diploma | No experience required | 15 | 3 months | Open |
| 19 | Jacob Lee | 1019 | IT | Database Administrator | Bachelor's degree | 3+ years database | 2 | 18 months | Open |
| 2 | Sarah Johnson | 1002 | IT | Web Developer | Bachelor's degree | 2 years | 2 | 18 months | Open |
| 20 | Rachel Kim | 1020 | Marketing | Marketing Coordinator | Bachelor's degree | 1+ years marketing | 4 | 6 months | Open |
| 3 | David Lee | 1003 | HR | HR Manager | Master's degree | 5 years HR exp | 1 | 24 months | Closed |
| 4 | Karen Brown | 1004 | Finance | Finance Analyst | Bachelor's degree | 2 years finance | 4 | 6 months | Open |
| 5 | Michael Chen | 1005 | Sales | Sales Representative | High school diploma | 1+ years sales | 10 | 3 months | Open |
| 6 | Emily Wilson | 1006 | Marketing | Marketing Specialist | Bachelor's degree | 3+ years marketing | 2 | 12 months | Open |
| 7 | Daniel Kim | 1007 | IT | IT Support Technician | Associate's degree | 1+ years IT | 5 | 6 months | Open |

- Cpi and password visible for the admin view of the student.

| ID | Name | email | program | deptname | cpi | pswd | Update | Delete |
|-------------|------------------------------|------------------|------------|------------------------|------|-------|--------|--------|
| 20110060 | divya.chinchhole@iitgn.ac.in | divya chinchhole | Btech | CSE | 8.2 | None | Update | Delete |
| 22120052 | jain.tanvi@iitgn.ac.in | Tanvi Jain | Btech | CSE | 9.7 | None | Update | Delete |
| 22120056 | tmj07042@gmail.com | Tanvi | Btech | cseee | 9.0 | None | Update | Delete |
| 22120057 | tmj0704222@gmail.com | Tanvi | Btech | cseee | 9.2 | None | Update | Delete |
| 22120059 | tmj0704224@gmail.com | Tanvi | Btech | cse | 9.0 | None | Update | Delete |
| 22120060 | tmj070421@gmail.com | Tanvi | Btech | cseeeeii | 9.0 | 123 | Update | Delete |
| 22120060 | tmj070422@gmail.com | Tanvi | Btech | mech | 10.0 | None | Update | Delete |
| 3523 | tmj0704@gmail.com | zsfhf | Btech | ec | 7.8 | 12345 | Update | Delete |
| 998791232 | braun.dino@example.com | ubgo | consequatu | ea | 4.0 | None | Update | Delete |
| 999096848 | grant.leonard@example.net | szaf | tempora | dolorem | 4.0 | None | Update | Delete |
| STU10000"8" | "rcaesman"8"@google.es | Reine | B.Tech | Computer Science | 6.0 | None | Update | Delete |
| STU100000 | jsturge0@amazon.co.uk | Johnathon | B.Tech | Computer Science | 5.0 | None | Update | Delete |
| STU100001 | gstotherfield1@vimeo.com | Gideon | B.Tech | Mechanical Engineering | 8.0 | None | Update | Delete |
| STU100002 | ptingle2@gov.uk | Norby | B.Tech | Electrical Engineering | 9.0 | None | Update | Delete |
| STU100003 | rlorriman3@parallels.com | Ryon | B.Tech | Civil Engineering | 10.0 | None | Update | Delete |
| STU100004 | mpercy4@altervista.org | Milty | B.Tech | Computer Science | 9.0 | None | Update | Delete |
| STU100005 | prenaman5@rediff.com | Paulo | B.Tech | Mechanical Engineering | 8.0 | None | Update | Delete |
| STU100006 | mlenox6@harvard.edu | Maridel | B.Tech | Electrical Engineering | 8.0 | None | Update | Delete |
| STU100007 | raithby7@cisco.com | Stan | B.Tech | Civil Engineering | 6.0 | None | Update | Delete |
| STU100009 | bbayldon9@redcross.org | Belvia | B.Tech | Mechanical Engineering | 6.0 | None | Update | Delete |
| STU100010 | bmayergera@wix.com | Birdie | B.Tech | Electrical Engineering | 8.0 | None | Update | Delete |

templates 3.zip

Show all

3. All login pages displayed at a single place

4. Renaming the tables name of student and professor to student_details and professor_details

```

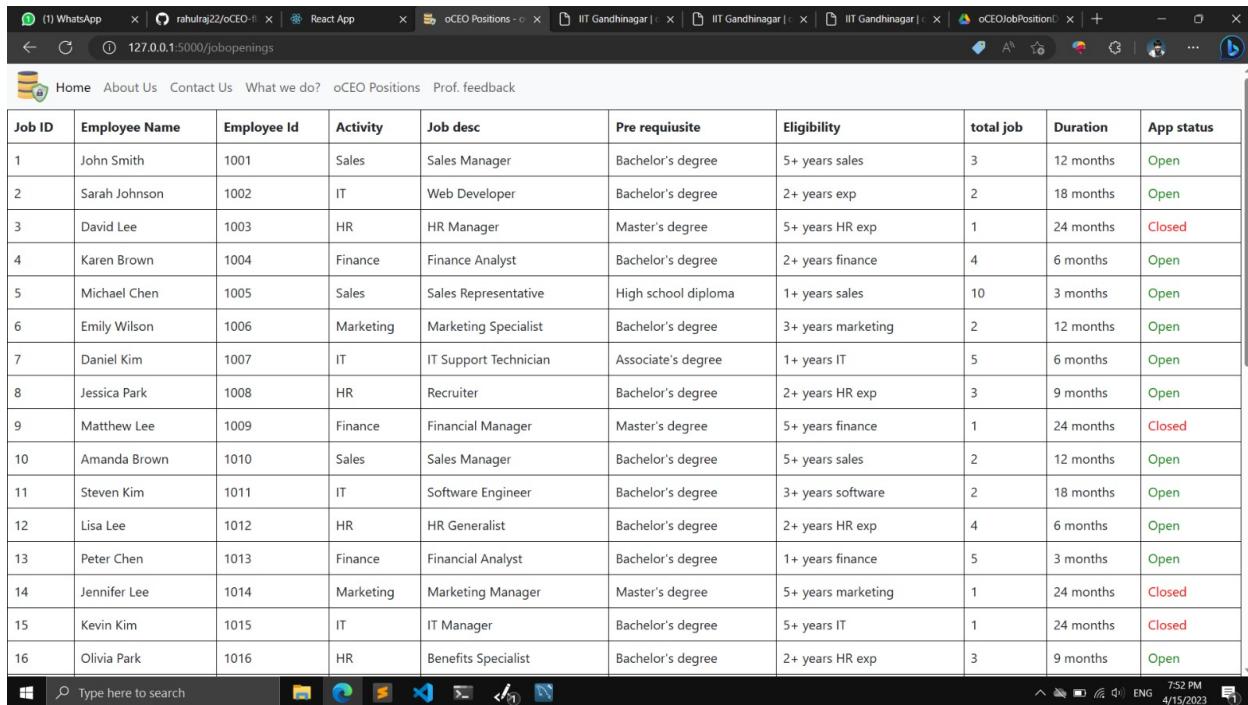
78 • show create table student;
79 • desc student;
80 • desc professor;
81
82 • alter table student rename student_details;
83 • alter table professor rename professor_details;

Output:
Action Output
# Time Action Message
 39 19:48:41 create table professor (email varchar(200), emp_id varchar(20), emp_name char(30), pswd varchar(200), primar... 0 row(s) affected
 40 19:48:50 INSERT INTO professor (email, emp_id, emp_name, pswd) VALUES ('professor1@university.edu', 'EMP10000...', 5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0
 41 19:51:08 desc oceo_positions 10 row(s) returned
 42 19:59:23 desc student desc professor Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds t
 43 19:59:31 desc student 7 row(s) returned
 44 20:00:18 altertable student rename student_details 0 row(s) affected

```

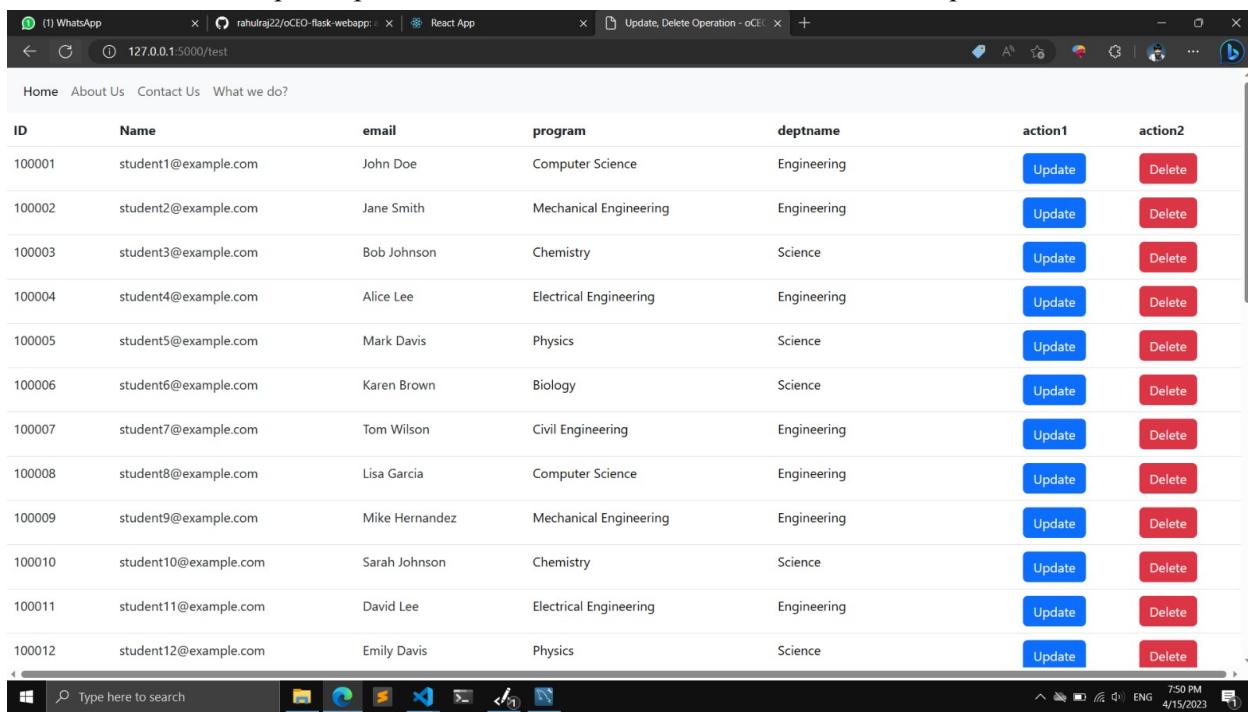
After first feedback

- Added a hyperlinked button on the open/close button for direct application process



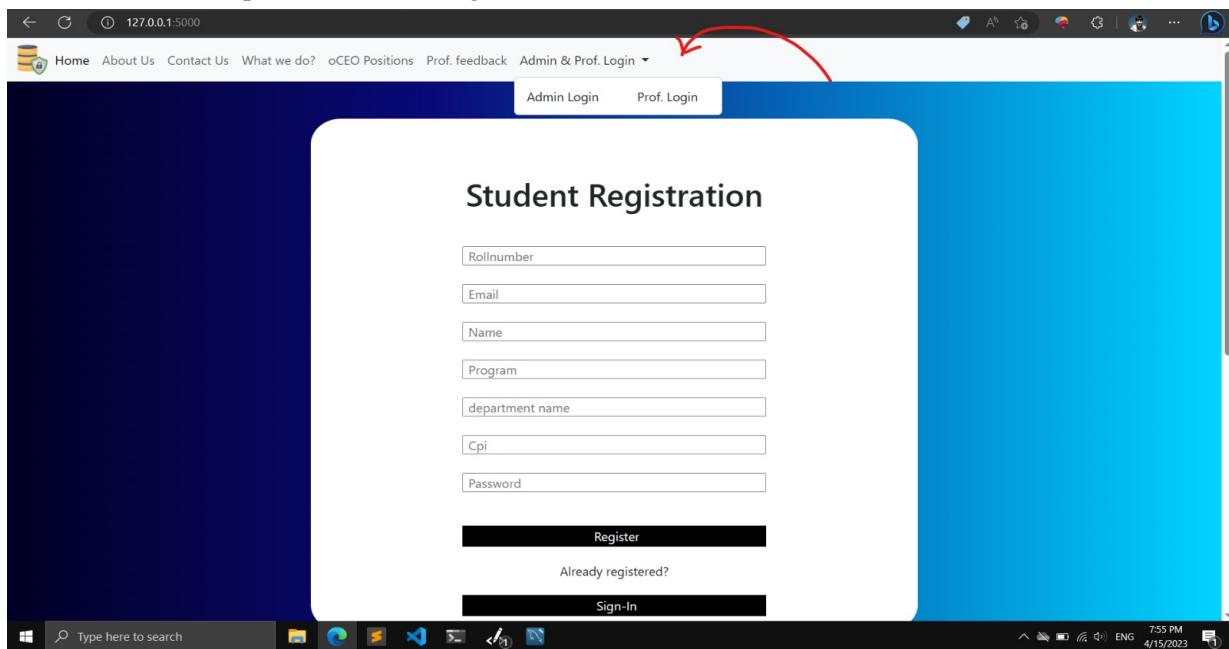
| Job ID | Employee Name | Employee Id | Activity | Job desc | Pre requisite | Eligibility | total job | Duration | App status |
|--------|---------------|-------------|-----------|-----------------------|---------------------|--------------------|-----------|-----------|------------|
| 1 | John Smith | 1001 | Sales | Sales Manager | Bachelor's degree | 5+ years sales | 3 | 12 months | Open |
| 2 | Sarah Johnson | 1002 | IT | Web Developer | Bachelor's degree | 2+ years exp | 2 | 18 months | Open |
| 3 | David Lee | 1003 | HR | HR Manager | Master's degree | 5+ years HR exp | 1 | 24 months | Closed |
| 4 | Karen Brown | 1004 | Finance | Finance Analyst | Bachelor's degree | 2+ years finance | 4 | 6 months | Open |
| 5 | Michael Chen | 1005 | Sales | Sales Representative | High school diploma | 1+ years sales | 10 | 3 months | Open |
| 6 | Emily Wilson | 1006 | Marketing | Marketing Specialist | Bachelor's degree | 3+ years marketing | 2 | 12 months | Open |
| 7 | Daniel Kim | 1007 | IT | IT Support Technician | Associate's degree | 1+ years IT | 5 | 6 months | Open |
| 8 | Jessica Park | 1008 | HR | Recruiter | Bachelor's degree | 2+ years HR exp | 3 | 9 months | Open |
| 9 | Matthew Lee | 1009 | Finance | Financial Manager | Master's degree | 5+ years finance | 1 | 24 months | Closed |
| 10 | Amanda Brown | 1010 | Sales | Sales Manager | Bachelor's degree | 5+ years sales | 2 | 12 months | Open |
| 11 | Steven Kim | 1011 | IT | Software Engineer | Bachelor's degree | 3+ years software | 2 | 18 months | Open |
| 12 | Lisa Lee | 1012 | HR | HR Generalist | Bachelor's degree | 2+ years HR exp | 4 | 6 months | Open |
| 13 | Peter Chen | 1013 | Finance | Financial Analyst | Bachelor's degree | 1+ years finance | 5 | 3 months | Open |
| 14 | Jennifer Lee | 1014 | Marketing | Marketing Manager | Master's degree | 5+ years marketing | 1 | 24 months | Closed |
| 15 | Kevin Kim | 1015 | IT | IT Manager | Bachelor's degree | 5+ years IT | 1 | 24 months | Closed |
| 16 | Olivia Park | 1016 | HR | Benefits Specialist | Bachelor's degree | 2+ years HR exp | 3 | 9 months | Open |

- Removed the cpi and password for the admin view of the student and improved the css



| ID | Name | email | program | deptname | action1 | action2 |
|--------|-----------------------|----------------|------------------------|-------------|------------------------|------------------------|
| 100001 | student1@example.com | John Doe | Computer Science | Engineering | Update | Delete |
| 100002 | student2@example.com | Jane Smith | Mechanical Engineering | Engineering | Update | Delete |
| 100003 | student3@example.com | Bob Johnson | Chemistry | Science | Update | Delete |
| 100004 | student4@example.com | Alice Lee | Electrical Engineering | Engineering | Update | Delete |
| 100005 | student5@example.com | Mark Davis | Physics | Science | Update | Delete |
| 100006 | student6@example.com | Karen Brown | Biology | Science | Update | Delete |
| 100007 | student7@example.com | Tom Wilson | Civil Engineering | Engineering | Update | Delete |
| 100008 | student8@example.com | Lisa Garcia | Computer Science | Engineering | Update | Delete |
| 100009 | student9@example.com | Mike Hernandez | Mechanical Engineering | Engineering | Update | Delete |
| 100010 | student10@example.com | Sarah Johnson | Chemistry | Science | Update | Delete |
| 100011 | student11@example.com | David Lee | Electrical Engineering | Engineering | Update | Delete |
| 100012 | student12@example.com | Emily Davis | Physics | Science | Update | Delete |

3. Added a dropdown menu for login



4. Renaming the tables name of student and professor to student_details and professor_details

```

82 • alter table student rename student_details;
83 • alter table professor rename professor_details;
84
85 • desc student_details;
86 • desc professor_details;

Output
Action Output
# Time Action
43 19:59:31 desc student
44 20:00:18 altertable student rename student_details
45 20:00:39 desc student
46 20:01:04 desc student_details
47 20:01:24 desc professor
48 20:01:39 altertable professor rename professor_details

Message
7 row(s) returned
0 row(s) affected
Error Code: 1146. Table 'oceo.student' doesn't exist
7 row(s) returned
4 row(s) returned
0 row(s) affected

```

Why did we not implement a single login page?

- Having multiple login pages for each stakeholder, such as students and professors, is better than having one single database for several reasons. Firstly, having separate login pages provides greater security by restricting access to the system based on specific user roles and permissions. This ensures that sensitive data is protected, and users can only access the information they need to perform their tasks.
- Having separate login pages makes it easier to manage the database as the system can be designed and optimized based on the specific needs of each stakeholder. For example, the user interface and functionalities can be tailored to meet the requirements of each group, leading to a more efficient and user-friendly system.

3. As the number of students and professors increases in the future, having multiple login pages ensures that the system can handle the growing number of users without compromising on performance. With a single database, the increase in users can lead to slower response times and increased processing times, resulting in reduced efficiency.

Why is CPI being shown on the front end?

The CPI attribute present on the frontend is only for the registration process for the student. Once the student has logged in , then he can see his CPI as it is a deciding factor for the qualification of the student. The registered students can directly login and check the job opportunities.

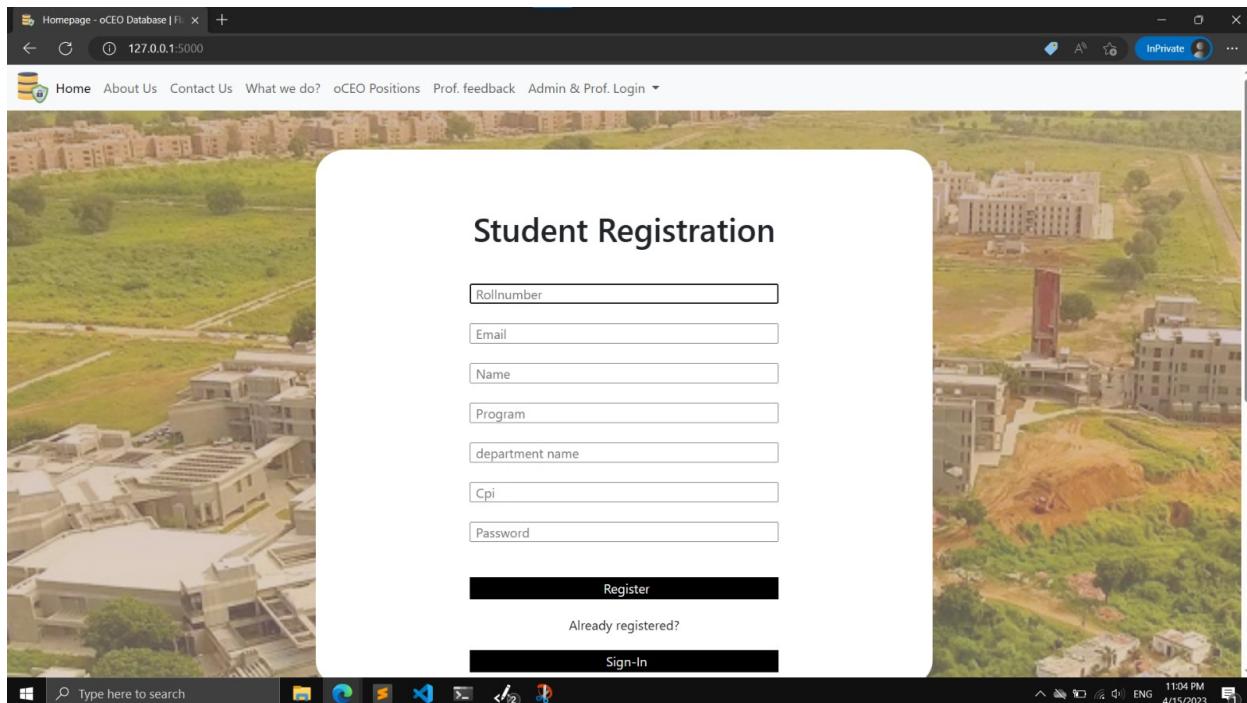
Defining the annual budget for oCEO positions is done by the administration for which we cannot hold any accountability as of now.

Feedback 2:-**Screenshots of the feedback from shared:**

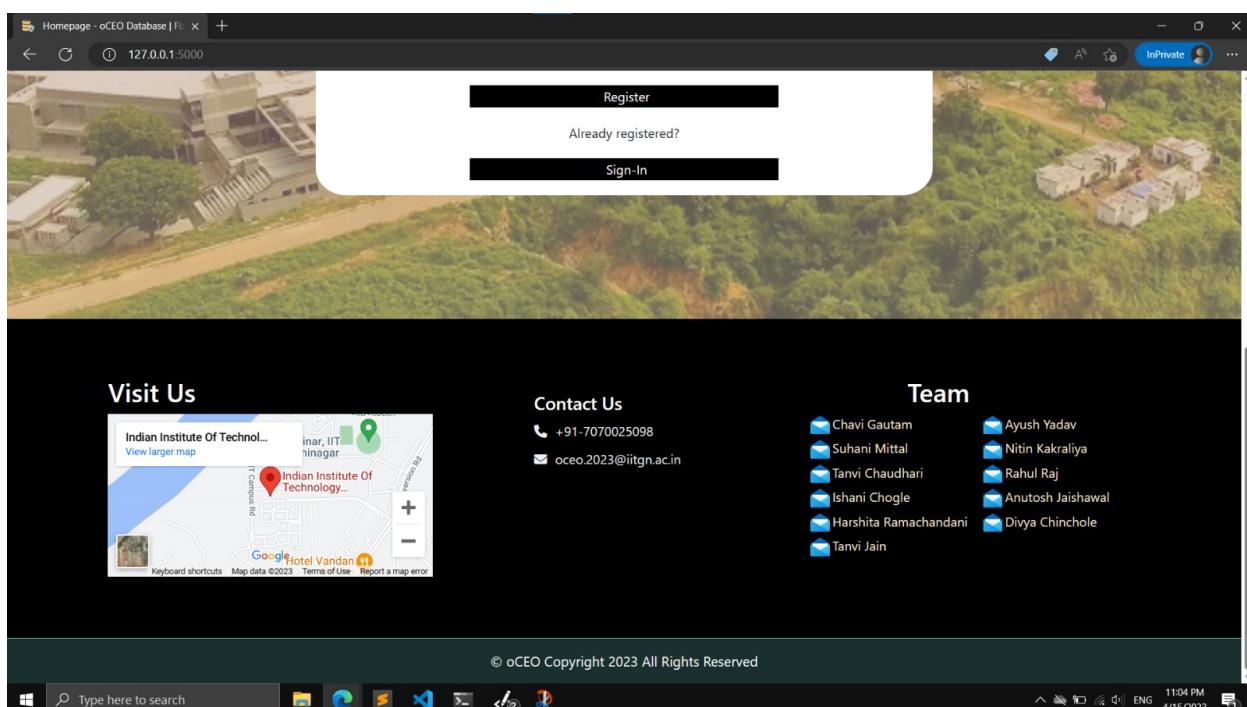
| | | |
|---|--|--|
| <p>Feedback Form</p> <p>2</p> <p>This is the final feedback form for the web app. Thanks for your giving constant suggestions through out the development process which gave us deeper insights into making the app user friendly.</p> <p>Email *</p> <p><u>chogle.ishani@iitgn.ac.in</u></p> <p>Team name *</p> <p><u>Oracles eleven</u></p> | <p>Name of stakeholder discussed with</p> <p><u>Ms Jinal Panchal, head, oCEO</u></p> <p>Do you feel the web app is more user friendly than the previous version? *</p> <p><input checked="" type="radio"/> Yes <input type="radio"/> No <input type="radio"/> Maybe</p> <p>Any feature that is still missing? *</p> <p>The front end can be improved more.</p> <p>Please explain your previous answer briefly. It would be really helpful to us. *</p> <p>The front end can be improved in</p> | <p>Please explain your previous answer briefly. It would be really helpful to us. *</p> <p>The front end can be improved in terms of the aesthetics involved. The other shortcomings pointed out during the first feedback have been taken care of</p> <p>Do you feel the changes you suggested earlier were incorporated in the new design? *</p> <p><input checked="" type="radio"/> Yes <input type="radio"/> No <input type="radio"/> Maybe</p> <p>Please mention any other comments concerning the application. *</p> <p>-</p> <p>Will the current version of our webapp, replace the deployed version we follow? *</p> <p><input checked="" type="radio"/> Yes <input type="radio"/> No <input type="radio"/> NA</p> |
|---|--|--|

After second feedback

After making changes in frontend and adding professor feedback:-



Footer Section:



We also hyperlinked the email id's of all the team members and changed the background image for a better user experience.

The screenshot shows a web browser window with the URL 127.0.0.1:5000/loginprof. The page title is "Professor's Feedback to Students". Below the title is a table with ten rows of student information and their professor feedback. Each row has "Update" and "Delete" buttons at the end. A red arrow points from the handwritten note "operations" to the "Delete" button in the last row.

| Student ID | Email | Name | Program | Dept. Name | Professor Feedback | Update | Delete |
|------------|-----------------------|----------------|------------------------|-------------|-------------------------|-------------------------|-------------------------|
| 100001 | student1@example.com | John Doe | Computer Science | Engineering | Good work! | <button>Update</button> | <button>Delete</button> |
| 100002 | student2@example.com | Jane Smith | Mechanical Engineering | Engineering | Keep it up! | <button>Update</button> | <button>Delete</button> |
| 100003 | student3@example.com | Bob Johnson | Chemistry | Science | No feedback available | <button>Update</button> | <button>Delete</button> |
| 100004 | student4@example.com | Alice Lee | Electrical Engineering | Engineering | You can do better. | <button>Update</button> | <button>Delete</button> |
| 100005 | student5@example.com | Mark Davis | Physics | Science | Great job! | <button>Update</button> | <button>Delete</button> |
| 100006 | student6@example.com | Karen Brown | Biology | Science | No feedback available | <button>Update</button> | <button>Delete</button> |
| 100007 | student7@example.com | Tom Wilson | Civil Engineering | Engineering | Keep up the good work! | <button>Update</button> | <button>Delete</button> |
| 100008 | student8@example.com | Lisa Garcia | Computer Science | Engineering | You need to study more. | <button>Update</button> | <button>Delete</button> |
| 100009 | student9@example.com | Mike Hernandez | Mechanical Engineering | Engineering | No feedback available | <button>Update</button> | <button>Delete</button> |
| 100010 | student10@example.com | Sarah Johnson | Chemistry | Science | Good effort! | <button>Update</button> | <button>Delete</button> |

3.1 G1:

1. The feedback taken from the stakeholders has been shared above.
2. Attach screenshots of different views [along with a write-up on their privileges] of the database as seen by different classes of users.

Answer:

View for Students:-

The student view allows the student to view and apply for job opportunities, update their profile, and view their application status. This view should be accessible and user-friendly to ensure that students can easily navigate the site and apply for job opportunities.

Registered Student Details

| ID | Email id | Name | Discipline | Department | CPI | Password |
|--------|----------------------|----------|------------------|-------------|-----|-----------|
| 100001 | student1@example.com | John Doe | Computer Science | Engineering | 3.5 | password1 |

Back to Login Page

Visit Us

Contact Us

Team

Chavi Gautam, Suhani Mittal, Ayush Yadav, Nitin Kakruliya

View for Professors:-

The screenshot shows a web browser window with a blue header bar containing various tabs like 'What?', 'DBMS_Asc...', 'DBMS_Asc...', 'oCEO-flas...', 'Homepage', 'oCEOJob', 'FontAwe...', 'A4_CS432', 'Homepage', 'Student D...', 'Font Awe...', and a search bar with '127.0.0.1:5000/loginprof'. Below the header is a navigation menu with links: Home, About Us, Contact Us, What we do?, oCEO Positions, Prof. feedback, Admin & Prof. Login.

The main content area has a dark blue background with a white rounded rectangle containing the following data:

Registered Professor Details

| Emp. ID | Email id | Emp. Name | Password |
|---------|-----------------|-----------|----------|
| 9999 | rahul@gmail.com | rahul | 123 |

Other Registered Professor Details

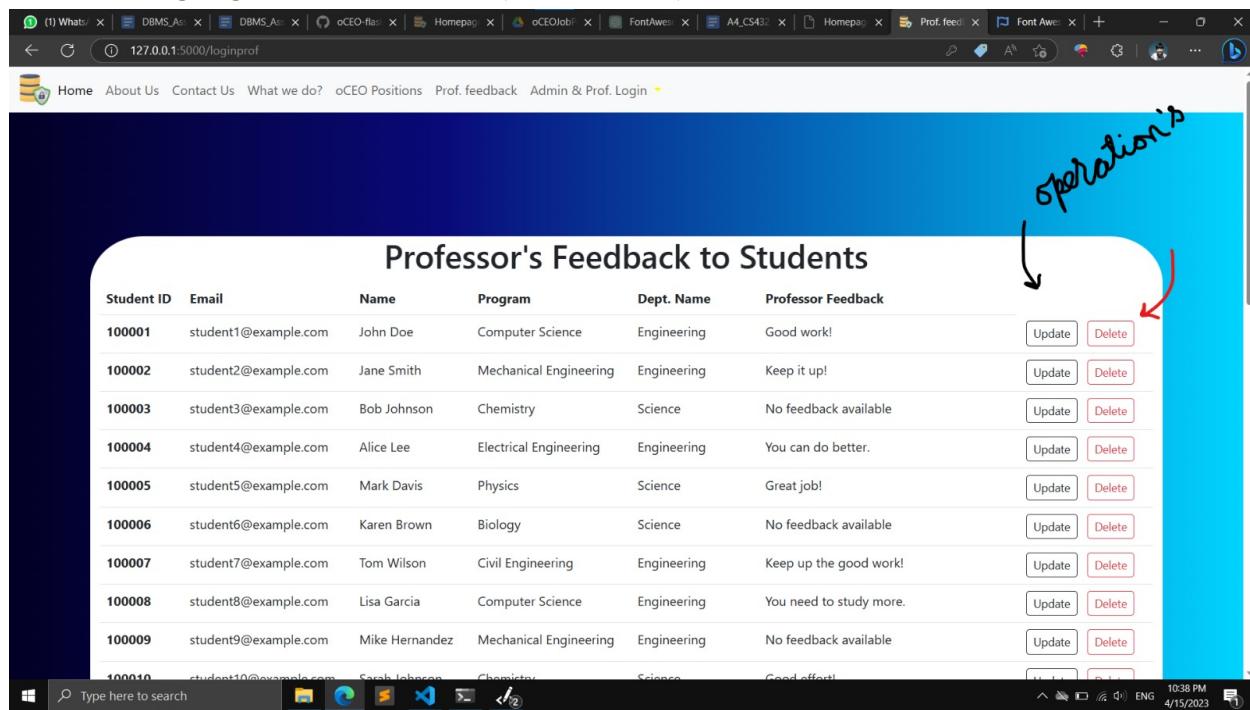
| Emp. ID | Email id | Emp. Name |
|---------|-----------------|-----------|
| 6666 | pearl@gmail.com | Pearl |
| 7777 | ayush@gmail.com | Ayush |
| 8888 | nitin@gmail.com | Nitin |
| 9999 | rahul@gmail.com | rahul |

[Back to Login Page](#)

The taskbar at the bottom shows the Windows logo, a search bar with 'Type here to search', and several pinned icons. The system tray shows the date and time as '4/15/2023 10:37 PM'.

The professor view allows the professor to provide feedback on student job performance, view student profiles, and recommend job opportunities to their students. This view should also be highly secure to ensure that no unauthorized access occurs and that only designated professors can view and provide feedback on student performance.

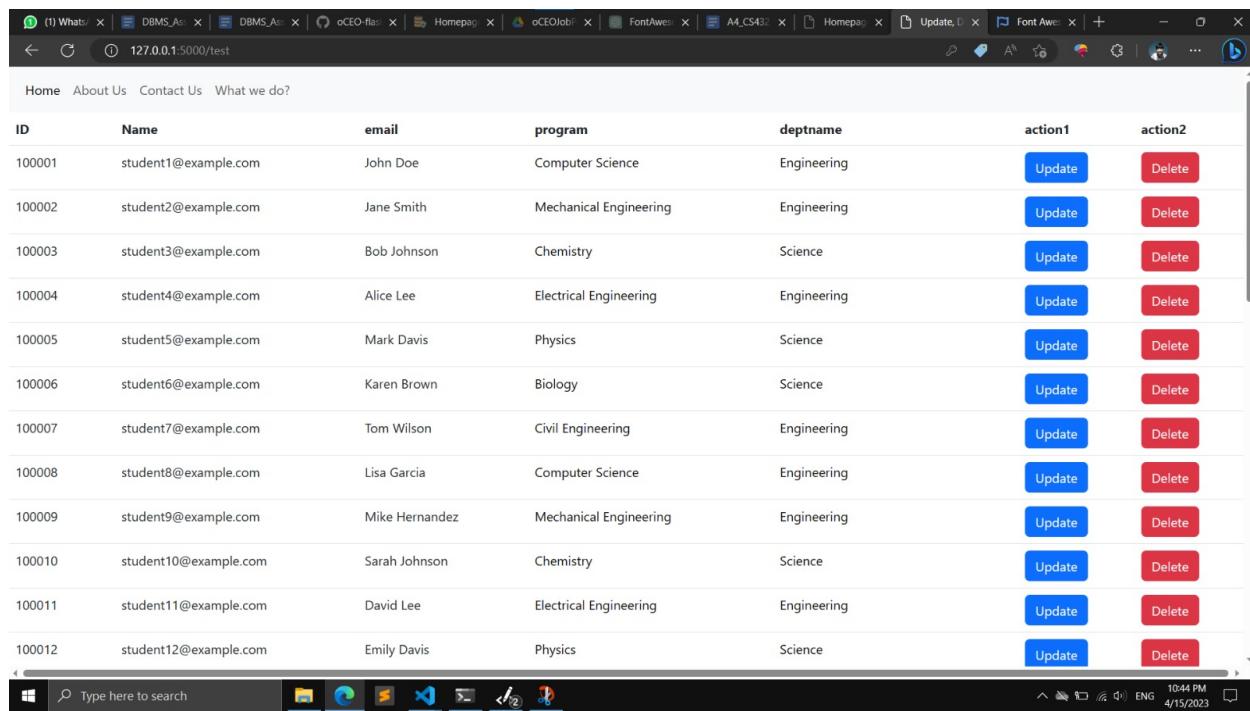
After going on Check Feedback(For Professor):-



The screenshot shows a table titled "Professor's Feedback to Students". The columns are: Student ID, Email, Name, Program, Dept. Name, and Professor Feedback. Each row contains a set of student information and their feedback, followed by two buttons: "Update" and "Delete". A red arrow points to the "Delete" button for the first row.

| Student ID | Email | Name | Program | Dept. Name | Professor Feedback | Action | Action |
|------------|-----------------------|----------------|------------------------|-------------|-------------------------|-------------------------|-------------------------|
| 100001 | student1@example.com | John Doe | Computer Science | Engineering | Good work! | <button>Update</button> | <button>Delete</button> |
| 100002 | student2@example.com | Jane Smith | Mechanical Engineering | Engineering | Keep it up! | <button>Update</button> | <button>Delete</button> |
| 100003 | student3@example.com | Bob Johnson | Chemistry | Science | No feedback available | <button>Update</button> | <button>Delete</button> |
| 100004 | student4@example.com | Alice Lee | Electrical Engineering | Engineering | You can do better. | <button>Update</button> | <button>Delete</button> |
| 100005 | student5@example.com | Mark Davis | Physics | Science | Great job! | <button>Update</button> | <button>Delete</button> |
| 100006 | student6@example.com | Karen Brown | Biology | Science | No feedback available | <button>Update</button> | <button>Delete</button> |
| 100007 | student7@example.com | Tom Wilson | Civil Engineering | Engineering | Keep up the good work! | <button>Update</button> | <button>Delete</button> |
| 100008 | student8@example.com | Lisa Garcia | Computer Science | Engineering | You need to study more. | <button>Update</button> | <button>Delete</button> |
| 100009 | student9@example.com | Mike Hernandez | Mechanical Engineering | Engineering | No feedback available | <button>Update</button> | <button>Delete</button> |
| 100010 | student10@example.com | Sarah Johnson | Chemistry | Science | Good effort! | <button>Update</button> | <button>Delete</button> |

View for Admins:-



The screenshot shows a table with columns: ID, Name, email, program, deptname, action1, and action2. Each row represents a student profile, with "action1" and "action2" buttons at the end of each row. A red arrow points to the "Delete" button for the first row.

| ID | Name | email | program | deptname | action1 | action2 |
|--------|-----------------------|----------------|------------------------|-------------|-------------------------|-------------------------|
| 100001 | student1@example.com | John Doe | Computer Science | Engineering | <button>Update</button> | <button>Delete</button> |
| 100002 | student2@example.com | Jane Smith | Mechanical Engineering | Engineering | <button>Update</button> | <button>Delete</button> |
| 100003 | student3@example.com | Bob Johnson | Chemistry | Science | <button>Update</button> | <button>Delete</button> |
| 100004 | student4@example.com | Alice Lee | Electrical Engineering | Engineering | <button>Update</button> | <button>Delete</button> |
| 100005 | student5@example.com | Mark Davis | Physics | Science | <button>Update</button> | <button>Delete</button> |
| 100006 | student6@example.com | Karen Brown | Biology | Science | <button>Update</button> | <button>Delete</button> |
| 100007 | student7@example.com | Tom Wilson | Civil Engineering | Engineering | <button>Update</button> | <button>Delete</button> |
| 100008 | student8@example.com | Lisa Garcia | Computer Science | Engineering | <button>Update</button> | <button>Delete</button> |
| 100009 | student9@example.com | Mike Hernandez | Mechanical Engineering | Engineering | <button>Update</button> | <button>Delete</button> |
| 100010 | student10@example.com | Sarah Johnson | Chemistry | Science | <button>Update</button> | <button>Delete</button> |
| 100011 | student11@example.com | David Lee | Electrical Engineering | Engineering | <button>Update</button> | <button>Delete</button> |
| 100012 | student12@example.com | Emily Davis | Physics | Science | <button>Update</button> | <button>Delete</button> |

The admin view has the highest privileges and allows the admin to manage job postings, student profiles, and application statuses. This view should be highly secure to ensure that no unauthorized access occurs.

and that only designated admins can manage the site's data. **The operation of the update and delete buttons are already shown in assignment 3 along with the images.**

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 can not be updated by two different users. For example, locks can be applied to tables in MySQL. **(10 Points)**

A shared lock is a type of lock that allows multiple users to read the same data simultaneously but prevents any of them from modifying it. This type of lock is used to ensure that data is not changed while it is being accessed by multiple users, which can lead to data inconsistencies and errors.

An exclusive lock, on the other hand, is a type of lock that prevents any other user from accessing or modifying the data while the lock is in place. This type of lock is used when a user needs to modify the data and ensure that no other user can access or modify it until the modification is complete.

In general, shared locks are used for read-only operations, while exclusive locks are used for write operations. By using these locks, the database management system ensures that multiple users can access and modify data simultaneously without causing data inconsistencies or errors. However, if these locks are not used correctly, they can lead to deadlocks, where multiple users are waiting for each other to release their locks, resulting in a system freeze. Therefore, it is essential to use these locks appropriately and carefully to ensure optimal database performance and prevent potential issues.

Code:- IN SHARED MODE FOR UPDATES

```
C:\> Users > Harshit Jain > Desktop > flaskdemo > oldhtml > app.py > test
108     deptname = updatedata['deptname']
109     cpi = updatedata['cpi']
110     pswd = updatedata['pswd']
111     id=updatedata['id']
112     print(id)
113
114     #cur.execute("UPDATE table student(prgm,deptname,cpi,pswd) set VALUES(%s, %s, %s, %s)",(prgm, deptname,cpi,pswd))
115     update1 = cur.execute("SELECT * FROM student WHERE student_id = %s FOR UPDATE", (id,))
116     update=cur.execute("UPDATE student SET program=%s, dept_name=%s, cpi=%s, pswd=%s WHERE student_id=%s", (prgm,
117     mysql.connection.commit()
118
119     if(update):
120         #flash('updated successfully','success')
121         resultValue = cur.execute("Select * from student LOCK IN SHARE MODE ")
122         if resultValue > 0:
123             userDetails = cur.fetchall()
124             return render_template('updatemsg.html')
125
126     except Exception as e:
127         print(f"Error: {str(e)}")
128     finally:
129         # Release the lock
```

```
C: > Users > Harshit Jain > Desktop > flaskdemo > oldhtml > app.py > editstudent
355     print(prgm)
356     deptname = updatedata['deptname']
357     cpi = updatedata['cpi']
358     pswd = updatedata['pswd']
359     id=updatedata['id']
360     print(id)
361
362     update=cur.execute("UPDATE student SET program=%s, dept_name=%s, cpi=%s, pswd=%s WHERE student_id=%s", (prgm,
363     mysql.connection.commit()
364     if(update):
365         #flash('updated succesfully','success')
366         | return render_template('updatemsg.html')
367
368
369 else:
370     email = session['email']
371     userDetails = cur.execute("Select * from student where email=%s FOR UPDATE" ,(email,))
372     if userDetails> 0:
373         userDetails = cur.fetchall()
374         return render_template('editstudent.html',userDetails=userDetails)
375 except Exception as e:
376     print(f"Error: {str(e)}")
377 finally:
```

```
C: > Users > Harshit Jain > Desktop > flaskdemo > oldhtml > app.py > editstudent
355     print(prgm)
356     deptname = updatedata['deptname']
357     cpi = updatedata['cpi']
358     pswd = updatedata['pswd']
359     id=updatedata['id']
360     print(id)
361
362     update=cur.execute("UPDATE student SET program=%s, dept_name=%s, cpi=%s, pswd=%s WHERE student_id=%s", (prgm,
363     mysql.connection.commit()
364     if(update):
365         #flash('updated succesfully','success')
366         | return render_template('updatemsg.html')
367
368
369 else:
370     email = session['email']
371     userDetails = cur.execute("Select * from student where email=%s FOR UPDATE" ,(email,))
372     if userDetails> 0:
373         userDetails = cur.fetchall()
374         return render_template('editstudent.html',userDetails=userDetails)
375 except Exception as e:
376     print(f"Error: {str(e)}")
377 finally:
```

G2:

2. Implement the changes in the database as per the feedback received from stakeholders. **(30 Points)**

1. Professor Feedback Page:-

After receiving feedback from stakeholders, we added a page for professor feedback on our website. This page allows professors to provide feedback to students on their job performance. This feedback can help students understand their strengths and weaknesses, and it can also provide valuable insights for future job opportunities. The professor feedback page is designed to be user-friendly and accessible, allowing professors to quickly and easily provide feedback on student work. Overall, this new feature enhances the communication and collaboration between students and professors, contributing to a better learning and job-seeking experience.

| Student ID | Email | Name | Program | Dept. Name | Professor Feedback |
|------------|-----------------------|----------------|------------------------|-------------|-------------------------|
| 100001 | student1@example.com | John Doe | Computer Science | Engineering | Good work! |
| 100002 | student2@example.com | Jane Smith | Mechanical Engineering | Engineering | Keep it up! |
| 100003 | student3@example.com | Bob Johnson | Chemistry | Science | No feedback available |
| 100004 | student4@example.com | Alice Lee | Electrical Engineering | Engineering | You can do better. |
| 100005 | student5@example.com | Mark Davis | Physics | Science | Great job! |
| 100006 | student6@example.com | Karen Brown | Biology | Science | No feedback available |
| 100007 | student7@example.com | Tom Wilson | Civil Engineering | Engineering | Keep up the good work! |
| 100008 | student8@example.com | Lisa Garcia | Computer Science | Engineering | You need to study more. |
| 100009 | student9@example.com | Mike Hernandez | Mechanical Engineering | Engineering | No feedback available |
| 100010 | student10@example.com | Sarah Johnson | Classmate | Science | Good effort! |

2. Edit student's profile

We added a feature that allows students to edit their profiles. With this feature, students can now easily update their information such as their name, mobile number, CPI, etc. This gives them greater control over their personal information and ensures that their profile remains accurate and up-to-date at all times. By allowing students to edit their profiles, we are providing them with a more user-friendly and personalized experience on our platform.

We update the profile of the student.

| ID | Name | Email | Program | Department | CPI | pswd |
|-------------------------------|------------|-------|---------|------------|-----|---------------------------------------|
| 221200522 tmj072234@gmail.com | Tanvi Jain | Btech | CSE | 3.5 | 123 | <input type="button" value="Update"/> |

The 'record updated successfully' message pops up.



In the current system, a user or student may update their CPI at any moment by selecting the "update" button, which enables CPI modification. This functionality exemplifies dynamic execution since it enables database updates based on user input in real-time.

3. Student application process

We added an apply option(hyperlink) after each job row. With this new feature, students can simply click on the apply option that is conveniently located next to each job listing, and they will be directed to the application form for that particular job. This feature will make it much easier for students to apply for jobs that they are interested in. Students will no longer have to navigate through several pages or scroll through a long list of jobs to find the one they want to apply for. This will save students time and effort and make the job application process more seamless and efficient.

| Job ID | Employee Name | Employee Id | Activity | Job desc | Pre requisite | Eligibility | total job | Duration | App status |
|--------|---------------|-------------|-----------|-----------------------|---------------------|--------------------|-----------|-----------|------------------------|
| 1 | John Smith | 1001 | Sales | Sales Manager | Bachelor's degree | 5+ years sales | 3 | 12 months | Open |
| 2 | Sarah Johnson | 1002 | IT | Web Developer | Bachelor's degree | 2+ years exp | 2 | 18 months | Open |
| 3 | David Lee | 1003 | HR | HR Manager | Master's degree | 5+ years HR exp | 1 | 24 months | Closed |
| 4 | Karen Brown | 1004 | Finance | Finance Analyst | Bachelor's degree | 2+ years finance | 4 | 6 months | Open |
| 5 | Michael Chen | 1005 | Sales | Sales Representative | High school diploma | 1+ years sales | 10 | 3 months | Open |
| 6 | Emily Wilson | 1006 | Marketing | Marketing Specialist | Bachelor's degree | 3+ years marketing | 2 | 12 months | Open |
| 7 | Daniel Kim | 1007 | IT | IT Support Technician | Associate's degree | 1+ years IT | 5 | 6 months | Open |
| 8 | Jessica Park | 1008 | HR | Recruiter | Bachelor's degree | 2+ years HR exp | 3 | 9 months | Open |
| 9 | Matthew Lee | 1009 | Finance | Financial Manager | Master's degree | 5+ years finance | 1 | 24 months | Closed |
| 10 | Amanda Brown | 1010 | Sales | Sales Manager | Bachelor's degree | 5+ years sales | 2 | 12 months | Open |
| 11 | Steven Kim | 1011 | IT | Software Engineer | Bachelor's degree | 3+ years software | 2 | 18 months | Open |
| 12 | Lisa Lee | 1012 | HR | HR Generalist | Bachelor's degree | 2+ years HR exp | 4 | 6 months | Open |
| 13 | Peter Chen | 1013 | Finance | Financial Analyst | Bachelor's degree | 1+ years finance | 5 | 3 months | Open |
| 14 | Jennifer Lee | 1014 | Marketing | Marketing Manager | Master's degree | 5+ years marketing | 1 | 24 months | Closed |
| 15 | Kevin Kim | 1015 | IT | IT Manager | Bachelor's degree | 5+ years IT | 1 | 24 months | Closed |
| 16 | Olivia Park | 1016 | HR | Benefits Specialist | Bachelor's degree | 2+ years HR exp | 3 | 9 months | Open |

4. Student and Professor Information:

| Student information | | | | | | | |
|---------------------|--------------------------|-----------------|---------|-------------------------|---------|---|---|
| ID | Name | email | program | deptname | action1 | action2 | |
| 001 | johndoe@gmail.com | John Doe | BSCS | Computer Science | None | <button>Update</button> <button>Delete</button> | |
| 002 | janedoe@yahoo.com | Jane Doe | BBA | Business Administration | None | <button>Update</button> <button>Delete</button> | |
| 003 | joshuasmith@gmail.com | Joshua Smith | BSCE | Civil Engineering | None | <button>Update</button> <button>Delete</button> | |
| 004 | sarahbrown@gmail.com | Sarah Brown | BSEE | Electrical Engineering | None | <button>Update</button> <button>Delete</button> | |
| 005 | peterparker@gmail.com | Peter Parker | BSCS | Computer Science | None | <button>Update</button> <button>Delete</button> | |
| 006 | clarkkent@yahoo.com | Clark Kent | BSEE | Electrical Engineering | None | <button>Update</button> <button>Delete</button> | |
| 007 | brucewayne@gmail.com | Bruce Wayne | BBA | Business Administration | None | <button>Update</button> <button>Delete</button> | |
| 008 | dianaprince@yahoo.com | Diana Prince | BSCS | Computer Science | None | <button>Update</button> <button>Delete</button> | |
| 009 | tonystark@gmail.com | Tony Stark | BSEE | Electrical Engineering | None | <button>Update</button> <button>Delete</button> | |
| 010 | stephenstrange@yahoo.com | Stephen Strange | BSCE | Civil Engineering | None | <button>Update</button> <button>Delete</button> | |
| 21 | abc@gmail.com | Yam | Diploma | Product Enginerring | 9.0 | fadfafiji | <button>Update</button> <button>Delete</button> |

| Professor Information | | | | |
|-----------------------|-----------------------|--------------|---------|---|
| emp_ID | email | emp_Name | action1 | action2 |
| 001 | johndoe@gmail.com | John Doe | BSCS | <button>Update</button> <button>Delete</button> |
| 002 | janedoe@yahoo.com | Jane Doe | BBA | <button>Update</button> <button>Delete</button> |
| 003 | joshuasmith@gmail.com | Joshua Smith | BSCE | <button>Update</button> <button>Delete</button> |
| 004 | sarahbrown@gmail.com | Sarah Brown | BSEE | <button>Update</button> <button>Delete</button> |
| 005 | peterparker@gmail.com | Peter Parker | BSCS | <button>Update</button> <button>Delete</button> |
| 006 | clarkkent@yahoo.com | Clark Kent | BSEE | <button>Update</button> <button>Delete</button> |
| 007 | brucewayne@gmail.com | Bruce Wayne | BBA | <button>Update</button> <button>Delete</button> |
| 008 | dianaprince@yahoo.com | Diana Prince | BSCS | <button>Update</button> <button>Delete</button> |
| 009 | tonystark@gmail.com | Tony Stark | BSEE | <button>Update</button> <button>Delete</button> |

Before there were only student information on the admin page, now admin can see both student and professor information and update or delete them as per requirement.

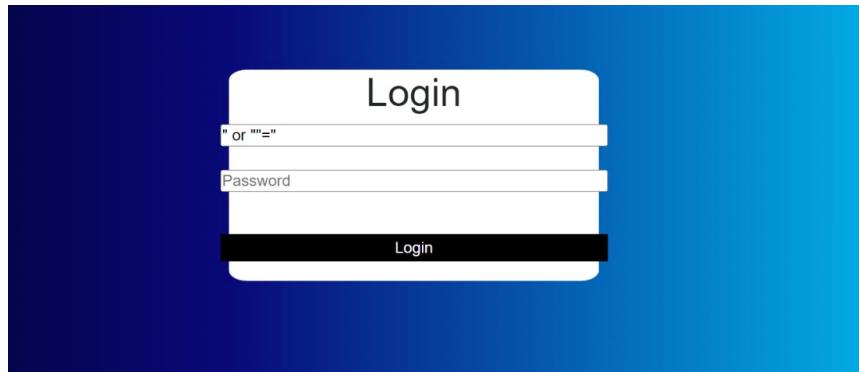
3.3 Responsibility of G1 & G2:

20 Pts

1. Documentation and screenshots of a total of 2 attacks [SQL Injection and XSS] performed and the defenses against those attacks. **(15 points)**
 - a. Additional attack and defense will lead to 10 **bonus** points for the team. (*Maximum 20 bonus points*)

Ans:

1. SQL Injection:



The query "or" "=" is often used as part of SQL injection attacks where an attacker tries to bypass authentication or access sensitive data by manipulating the query to modify its behavior.

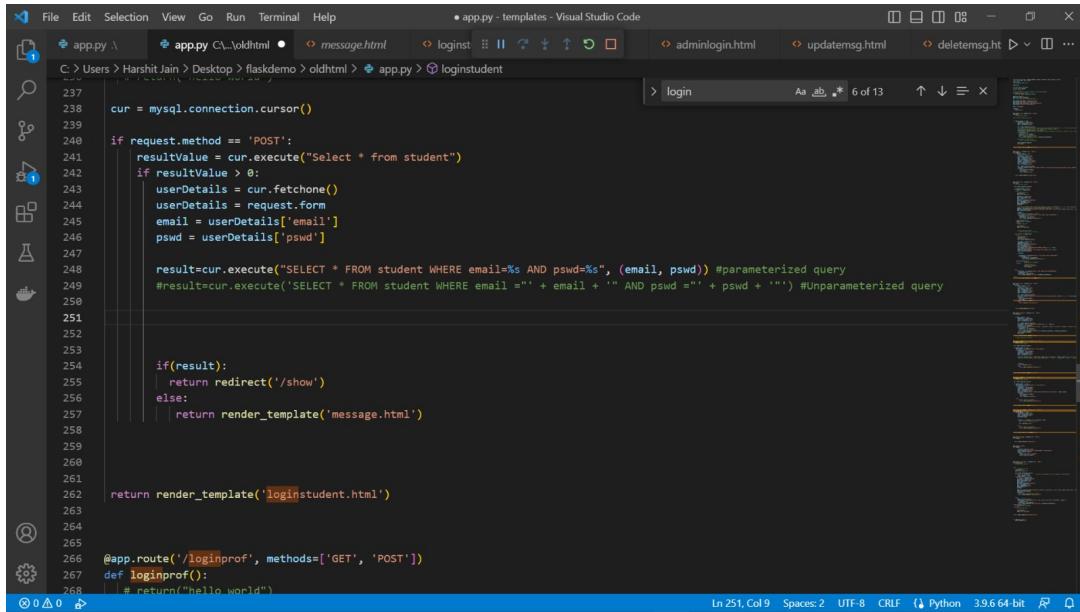
The OR condition with an empty string value will always be true, effectively bypassing the authentication check and allowing the attacker to login as the admin user.

On running this query in input,

| ID | Email id | Name | Discipline | Department | Cpi | pswd |
|-------------|-----------------------------|-----------------|------------|------------------------|------|-------|
| 20110060 | divya.chinchole@iitgn.ac.in | divya chinchole | Btech | CSE | 8.0 | None |
| 22120052 | jain.tanvi@iitgn.ac.in | Tanvi Jain | Btech | CSE | 10.0 | None |
| 221200522 | tmj072234@gmail.com | Tanvi Jain | Btech | CSE | 9.3 | 1234 |
| 22120056 | tmj07042@gmail.com | Tanvi | Btech | cseee | 9.3 | None |
| 22120057 | tmj0704222@gmail.com | Tanvi | Btech | cseee | 9.2 | None |
| 22120059 | tmj07042224@gmail.com | Tanvi | Btech | cse | 9.0 | None |
| 221200590 | tmj070421@gmail.com | Tanvi | Btech | cseeeeii | 9.0 | 123 |
| 22120060 | tmj070422@gmail.com | Tanvi | Btech | mech | 10.0 | None |
| 3523 | tmj0704@gmail.com | zsfhf | Btech | ec | 7.8 | 12345 |
| 998791232 | braun.dino@example.com | ubgo | consequatu | ea | 4.0 | None |
| STU10000"8" | rcaesman"8"@google.es | Reine | B.Tech | Computer Science | 6.0 | None |
| STU100000 | jsturge0@amazon.co.uk | Johnathon | B.Tech | Computer Science | 5.0 | None |
| STU100001 | gstotherfield1@vimeo.com | Gideon | B.Tech | Mechanical Engineering | 8.0 | None |
| STU100002 | ntingle2@gov.uk | Norby | B.Tech | Electrical Engineering | 9.0 | None |

Defense against these attacks:

Parameterized SQL queries are an effective way to prevent SQL injection attacks in web applications.



```

File Edit Selection View Go Run Terminal Help
app.py C:\Users\Harshit Jain\Desktop\flaskdemo\oldhtml message.html login adminlogin.html updatemsg.html deletemsg.html
237 cur = mysql.connection.cursor()
238
239 if request.method == 'POST':
240     resultValue = cur.execute("Select * from student")
241     if resultValue > 0:
242         userDetails = cur.fetchone()
243         userDetails = request.form
244         email = userDetails['email']
245         pswd = userDetails['pswd']
246
247         result=cur.execute("SELECT * FROM student WHERE email=%s AND pswd=%s", (email, pswd)) #parameterized query
248         #result=cur.execute('SELECT * FROM student WHERE email ='+ email + " AND pswd ="+ pswd + "'") #Unparameterized query
249
250
251
252
253
254     if(result):
255         return redirect('/show')
256     else:
257         return render_template('message.html')
258
259
260
261
262     return render_template('loginstudent.html')
263
264
265
266 @app.route('/loginprof', methods=['GET', 'POST'])
267 def loginprof():
268     if request.method == 'GET':
269         return "Hello World"

```

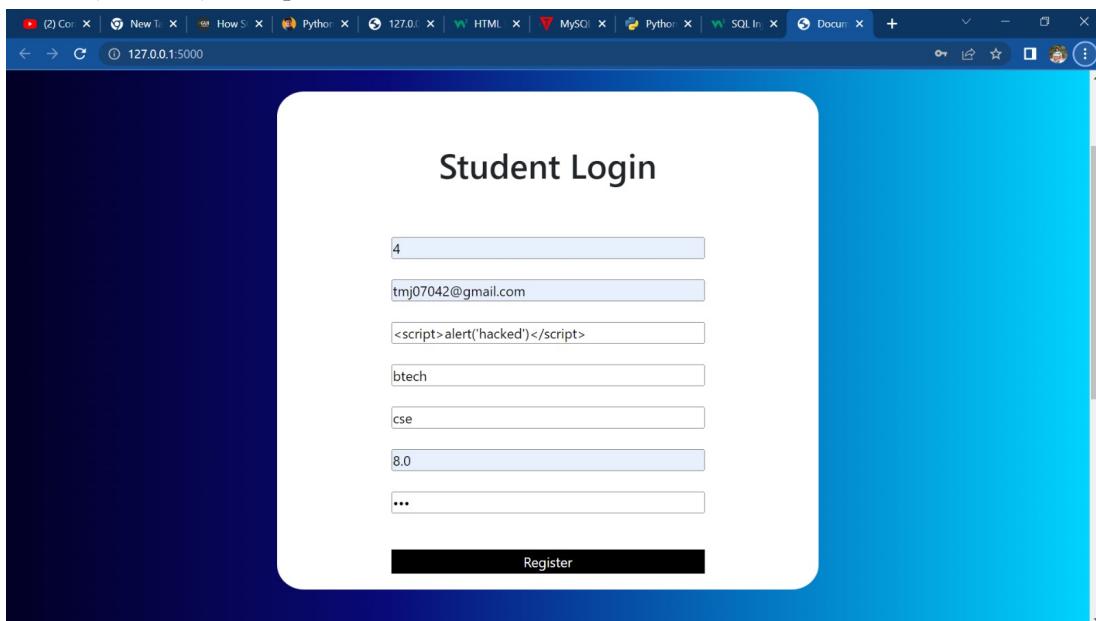
#the parameterized query used:

`"SELECT * FROM student WHERE email= %s AND pswd = %s", (email,pswd)"`

2. XSS Attack:

On giving input as:

`<script>alert('hacked')</script>`



The input gets saved as it is in our database.

The screenshot shows a MySQL query window with the following code:

```

240
241 • alter table studenti
242     add column pswd varchar(20);
243
244 • select * from studenti;
    
```

Below the code is a Result Grid showing one row of data:

| student_id | email | stud_name | program | dept_name | cpi | pswd |
|------------|-----------------------|----------------------------------|---------|-----------|-----|------|
| 4 | tmj07042222@gmail.com | <script>alert('hacked')</script> | btech | cse | 8 | 123 |

At the bottom, the Output pane shows the history of actions:

| # | Time | Action | Message |
|----|----------|---|--|
| 28 | 22:26:05 | select * from studenti LIMIT 0, 1000 | 37 row(s) returned |
| 29 | 22:43:22 | create table studenti(student_id varchar(20) NOT NULL, email varchar(50) UNIQUE, stud_name varchar(...) | 0 row(s) affected |
| 30 | 22:45:28 | alter table studenti add column pswd varchar(20) | 0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0 |
| 31 | 22:46:37 | select * from studenti LIMIT 0, 1000 | 1 row(s) returned |

To prevent XSS attack, autoescape is used,

The screenshot shows a Visual Studio Code interface with the following code in a student.html file:

```

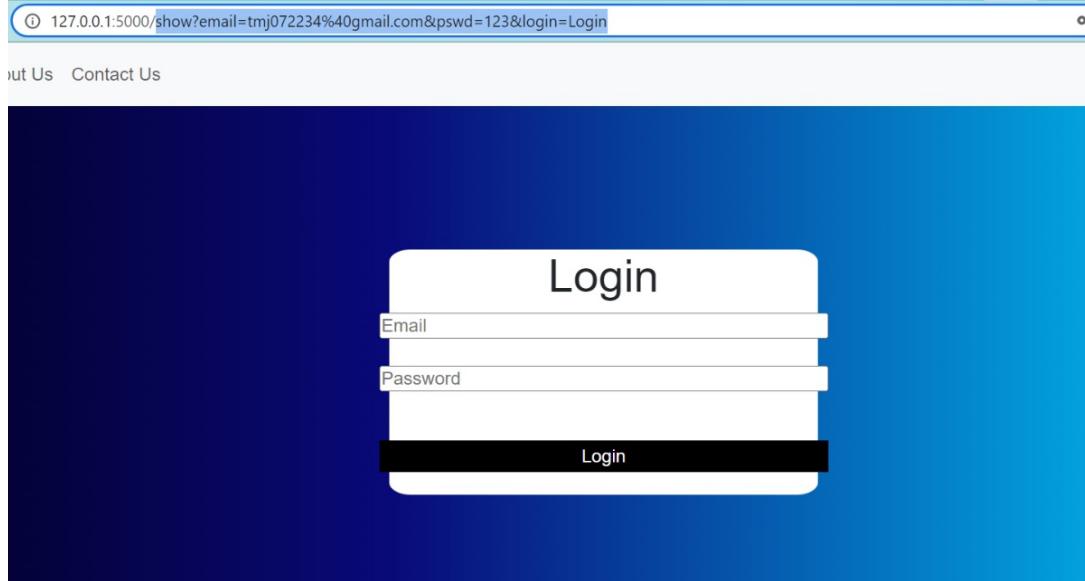
1  {% autoescape true %}
2  <!DOCTYPE html>
3  <html lang="en">
4  <head>
5      <meta charset="UTF-8">
6      <meta http-equiv="X-UA-Compatible" content="IE=edge">
7      <meta name="viewport" content="width=device-width, initial-scale=1.0">
8      <title>Document</title>
9      <link rel="stylesheet" href="{{ url_for('static', filename='student.css') }}"/>
10     <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css">
11  </head>
12  <body>
13      <nav class="navbar navbar-expand-lg bg-body-tertiary">
14          <div class="container-fluid">
15              <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNavDropdown" aria-
16                  <span class="navbar-toggler-icon"></span>
17              </button>
18              <div class="collapse navbar-collapse" id="navbarNavDropdown">
19                  <ul class="navbar-nav">
20                      <li class="nav-item">
21                          <a class="nav-link active" aria-current="page" href="{{ url_for('static', filename='student.css') }}>H
    
```

The code uses double curly braces {{ }} for interpolation, which enables autoescaping. The terminal output shows the server is running on port 5000.

When autoescape is enabled, all user input that is rendered on a web page is automatically cleaned up so that characters like, >, and & are substituted with their respective HTML entities. These HTML entities will be displayed as plain text on the page rather than being executed as part of a script. This lessens the likelihood of malicious code injection and other XSS attacks.

3. GET POST updation:

We have used GET before which shows email and password in the query string which can be harmful. As a solution we have used POST method that will not show the data in the query string like the password.



All the data (highlighted in the web page route) is in the querystring. Before, GET :

```

<form method="GET" action=""></pre>

```

```

<li class="nav-item">
  <a class="nav-link" href="index.html">About Us</a>
</li>
<li class="nav-item">
  <a class="nav-link" href="#footer">Contact Us</a>
</li>
</ul>
</div>
</div>
</nav>
<div class="container">
  <h1>Login</h1>
  <form method="post" action="">
    <input type="text" name="email" placeholder="Email" required />
    <br>
    <input type="password" name="pswd" placeholder="Password" />
    <br>
    <input type="submit" name="login" value="Login" id="submit">
  </form>
</div>

```

* Debugger PIN: 308-630-757
127.0.0.1 - [15/Apr/2023 23:06:00] "GET / HTTP/1.1" 200 -
127.0.0.1 - [15/Apr/2023 23:06:00] "GET /static/student.css HTTP/1.1" 304 -
127.0.0.1 - [15/Apr/2023 23:06:09] "GET /show HTTP/1.1" 200 -
127.0.0.1 - [15/Apr/2023 23:06:09] "GET /static/student.css HTTP/1.1" 304 -
127.0.0.1 - [15/Apr/2023 23:06:17] "GET /show?email=tmj072234@gmail.com&pswd=123&login=Login HTTP/1.1" 200 -
127.0.0.1 - [15/Apr/2023 23:06:17] "GET /static/student.css HTTP/1.1" 304 -

4. To prevent hackers from accessing the passwords of your website's users, password encryption is required. A user's password is encrypted into a hash when they establish an account, making it difficult to decode the password back to its original form.

Before: we havent encrypted the pswd which can be harmful for privacy

```

dept_name varchar(25) NOT NULL,
cpi float(2),
pswd varchar(200),
primary key (student_id)
);

253
254 • alter table studenti
add column pswd varchar(200);

255
256
257 • select * from studenti;
258

```

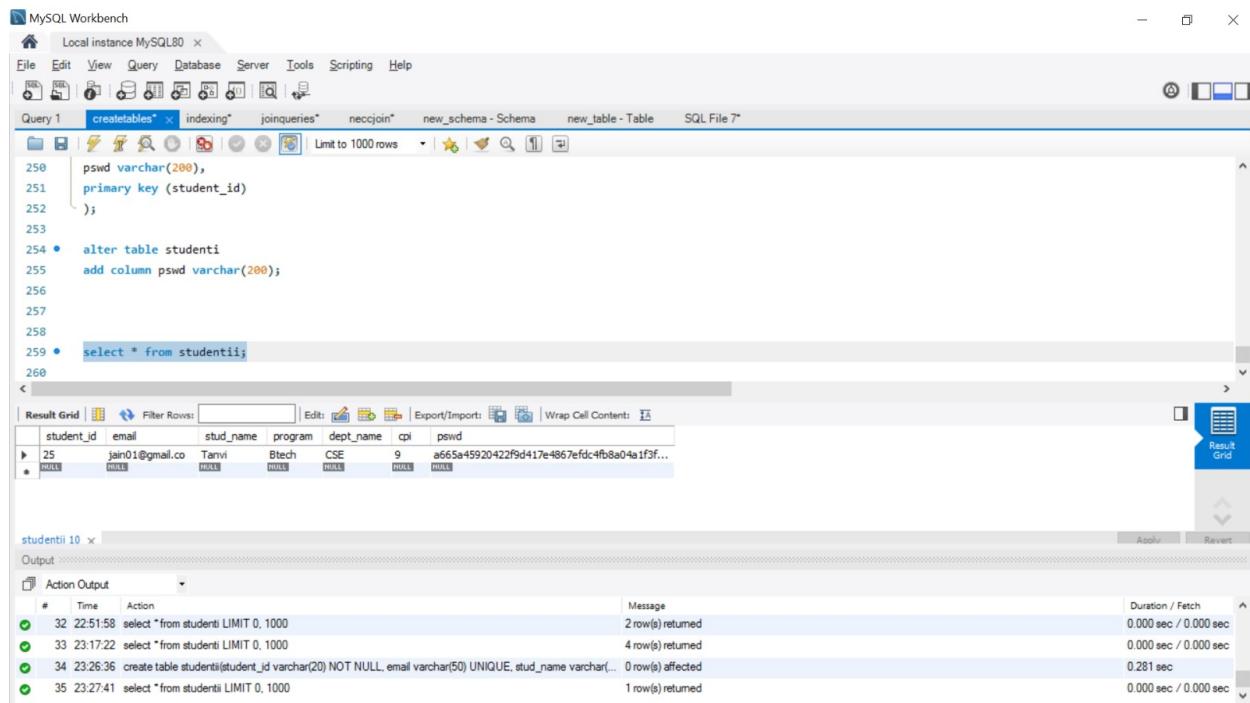
| student_id | email | stud_name | program | dept_name | cpi | pswd |
|------------|------------------------|------------|---------|-----------|-----|------|
| 22120052 | jain.tanvi@iitgn.ac.in | Tanvi Jain | Btech | CSE | 10 | None |
| 221200522 | tmj072234@gmail.com | Tanvi Jain | Btech | CSE | 9.3 | 1234 |
| 22120056 | tmj070422@gmail.com | Tanvi | Btech | cseee | 9.3 | None |
| 22120057 | tmj0704222@gmail.com | Tanvi | Btech | cseee | 9.2 | None |
| 22120059 | tmj07042224@gmail.com | Tanvi | Btech | cse | 9 | NULL |

Action Output

| # | Time | Action | Message | Duration / Fetch |
|----|----------|--|--------------------|-----------------------|
| 34 | 23:26:36 | create table studenti(student_id varchar(20) NOT NULL, email varchar(50) UNIQUE, stud_name varchar(25) NOT NULL, program varchar(20), dept_name varchar(25) NOT NULL, cpi float(2), pswd varchar(200), primary key (student_id)) | 0 row(s) affected | 0.281 sec |
| 35 | 23:27:41 | select * from studenti LIMIT 0, 1000 | 1 row(s) returned | 0.000 sec / 0.000 sec |
| 36 | 23:28:10 | alter table studenti | 0 row(s) affected | 0.078 sec |
| 37 | 23:28:29 | select * from studenti LIMIT 0, 1000 | 37 row(s) returned | 0.000 sec / 0.000 sec |

By doing this, you can make sure that even if an attacker manages to access the database of your website, they won't be able to see the real passwords. For password encryption, hashing algorithms like bcrypt, scrypt, and Argon2 are frequently used. However, to fully safeguard your website and its users, you should also implement other security measures like strict password policies and multi-factor authentication.

After Encryption: It was not required earlier but it was made required after encryption.



The screenshot shows the MySQL Workbench interface. In the Query Editor, the following SQL code is run:

```

250     pswd varchar(200),
251     primary key (student_id)
252   );
253
254 • alter table studenti
255   add column pswd varchar(200);
256
257
258
259 • select * from studentii;
260

```

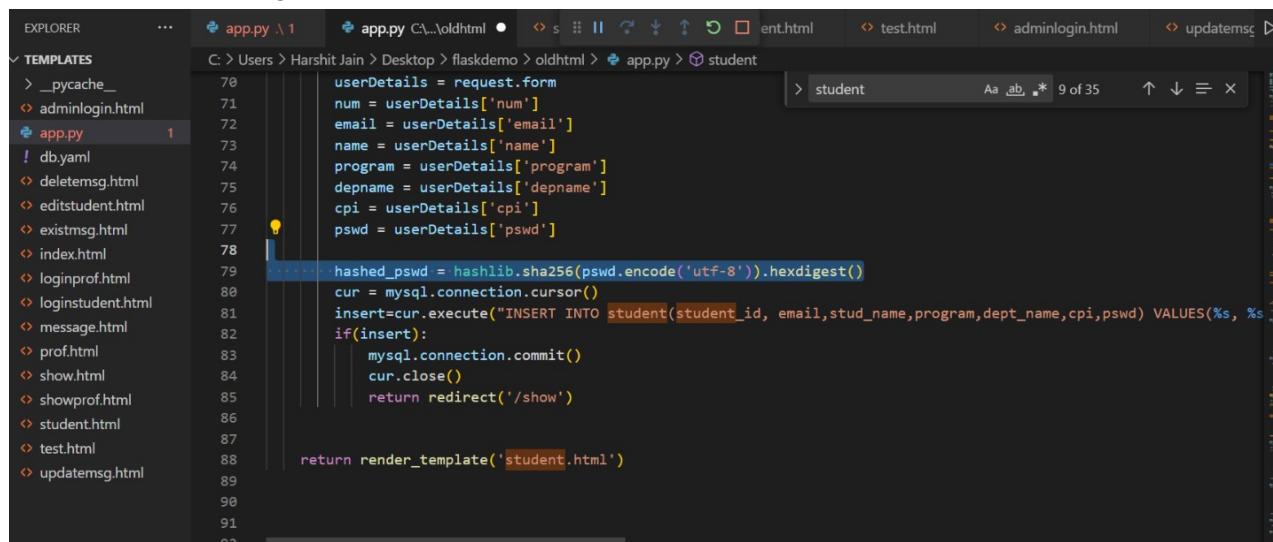
The Result Grid displays the data for the studenti table:

| student_id | email | stud_name | program | dept_name | cpi | pswd |
|------------|-----------------|-----------|---------|-----------|-----|---|
| 25 | jain01@gmail.co | Tanvi | Btech | CSE | 9 | a665a45920422f9d417e4867efdc4fb8a04a1f3f... |

The Output pane shows the execution log:

| # | Time | Action | Message | Duration / Fetch |
|----|----------|---|-------------------|-----------------------|
| 32 | 22:51:58 | select * from studenti LIMIT 0, 1000 | 2 row(s) returned | 0.000 sec / 0.000 sec |
| 33 | 23:17:22 | select * from studenti LIMIT 0, 1000 | 4 row(s) returned | 0.000 sec / 0.000 sec |
| 34 | 23:26:36 | create table studenti(student_id varchar(20) NOT NULL, email varchar(50) UNIQUE, stud_name varchar(...) | 0 row(s) affected | 0.281 sec |
| 35 | 23:27:41 | select * from studentii LIMIT 0, 1000 | 1 row(s) returned | 0.000 sec / 0.000 sec |

The code used for it is given below:



The screenshot shows a code editor with the file `app.py` open. The code contains the following logic:

```

userDetails = request.form
num = userDetails['num']
email = userDetails['email']
name = userDetails['name']
program = userDetails['program']
depname = userDetails['depname']
cpi = userDetails['cpi']
pswd = userDetails['pswd']

hashed_pswd = hashlib.sha256(pswd.encode('utf-8')).hexdigest()

cur = mysql.connection.cursor()
insert=cur.execute("INSERT INTO student(student_id, email,stud_name,program,dept_name,cpi,pswd) VALUES(%s, %s, %s, %s, %s, %s, %s)")
if(insert):
    mysql.connection.commit()
    cur.close()
    return redirect('/show')

return render_template('student.html')

```

5. We have implemented several security measures on our admin page to ensure the safety of sensitive information. A login option has been added to restrict access to authorized users only. An alert box pops up when invalid login credentials are entered, preventing potential brute force attacks. Email and password input validation has been set up to only accept valid input and an alert box notifies the user if an SQL injection or incorrect details are entered. These measures work together to minimize unauthorized access to our admin page,

The image contains two screenshots of a web browser window, likely Microsoft Edge, displaying an 'Admin Login' form. Both screenshots show a validation alert box.

Screenshot 1: This screenshot shows a standard validation message. A red arrow points from the text 'alert' to the top-left corner of the alert box. The alert box contains the text '127.0.0.1:5000 says' and 'Please enter the correct Details.' Below the alert box is a blue 'OK' button.

Screenshot 2: This screenshot shows a more detailed validation message. A red curly brace highlights the entire alert box area. The alert box contains an exclamation mark icon and the text 'Please include an '@' in the email address. '' or ''='' is missing an '@''. Below the alert box is a blue 'Login' button.

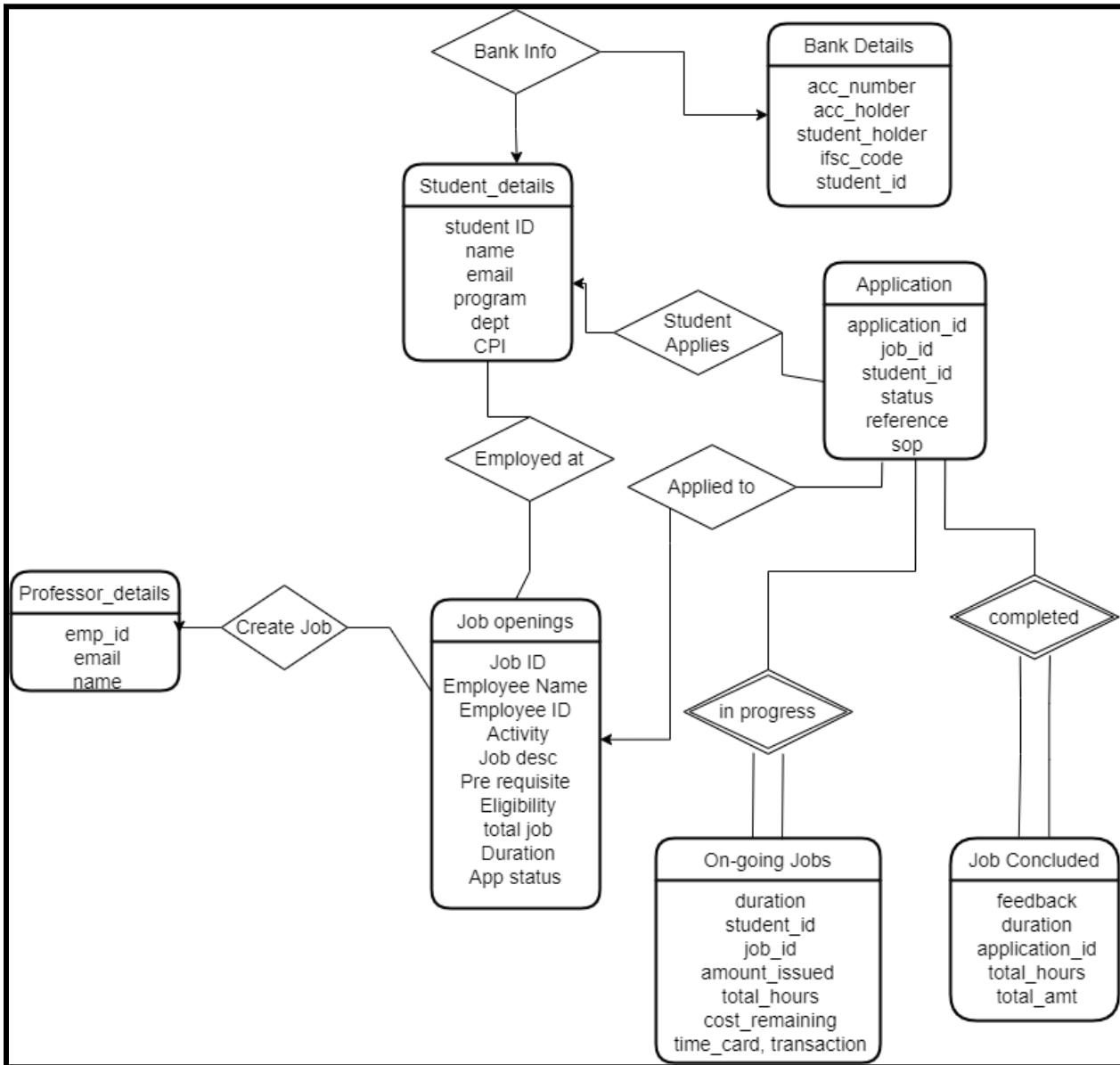
The browser's title bar shows the URL '127.0.0.1:5000/test'. The browser interface includes a navigation bar with links like 'Home', 'About Us', 'Contact Us', 'What we do?', 'oCEO Positions', and 'Prof. feedback'. At the bottom, there are links for 'Visit Us', 'Contact Us', and 'Team'.

G1&G2:

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.

Answer:

This is new ER diagram



And this table shows the all relationships between different tables as per in Assignment 1:

| Entity | Relationships | Attributes |
|----------------------|--|--|
| Student enrolled | Job Application: One to many(since one student can apply for more jobs) To professor: Many to one Ongoing and completed jobs—Many to one | Roll no ,program, student name ,program Dept,email,bank account, Previous job record |
| job | To student— many to one To applications—one to many | Eligibility, no. of seats, prerequisites, type of activity, job desc, current status |
| Applications | To student—one to one To professor—many to one | Jobid,studentid,reference,current status ,SOP |
| On going jobs | To students—one to many or many to one To professor—many to one | duration,student id,job id ,cost given,no of hours remaining,cost remaining ,transaction |
| Completed jobs | To students— one to many | Feedback,duration,student id ,job id,no of hours completed ,total amount |
| Bank details | To student id: one to one To jobs completed : one to one(use of foreign key) | Bank name,IFSC code,bank account no ,account holder name,student id |
| Professor associated | To jobs—One to many(since each professor can roll over many position) | Name,employee ,id department, Email ID |

It is made sure that all constraints are held while running sql queries for our web app.

Contributions:-

| Name | Contribution |
|----------------------|---|
| Harshita Ramchandani | Assisted in the backend and verification of documentation. |
| Divya Chinhole | Assisted in the final documentation and checked the correctness of the final document. Also did the research on attacks and SQL injection. |
| Ishani Chogle | Assisted in front end coding and documentation |
| Tanvi Chaudhari | Assisted in the backend. |
| Chhavi Gautam | Integrated the whole work along with pictures and did the documentation and answer the questions . Helped with G1 and G1 &G2 |
| Suhani Mittal | Helped in locating the earliest issues in the web app, contributed to the creation of the final document by responding to the questions in the G1, and G2, and compiling all the information together with images or screenshots. |
| Anutosh Jaishwal | Helped in finding the initial bugs throughout the web app. Contributed in the making of the final document including answering the questions of G1 and G1&G2 sections and compilation of all the data along with pictures. |
| Tanvi Jain | Contributed in backend, performed all attacks [SQL Injection and XSS and additional too] and the defenses against those attacks . Implemented multi user access(G2) and edit profile feature. |
| Rahul Raj | Improved the backend from SQL injection using truthy values (admin' OR '1' = '1) and similar other injections, also added strict check while admin login(through form). Added professor feedback for all students and |

| | |
|-----------------|--|
| | also actions like update(to update feedback) and delete(to delete the feedback, etc) |
| Ayush Yadav | Helped in changing the frontend like changing the background ,the login form.Improved the contact/footer section and included various features for better and efficient user experience. |
| Nitin kakraliya | Helped in the frontend part especially the questions raised in the second part like removing three different login tabs on the front page and converting them into single navbar for better user experience. |