

# CS 432: Databases

## Assignment 4: Deploying the DBMS

**Group Name:** La Retro's

**Topic:** Placement Management

### **Group Members**

**G1:** Aaryan Darad - 21110001

Abhay Kumar Upparwal - 21110004

Adit Kaushik - 21110010

Aditya Deshmukh - 21110014

**G2:** Ahaan Giriya - 21110015

Anugu Arun Reddy - 21110029

Gaurav Joshi - 21110065

Rutwik More - 21110133

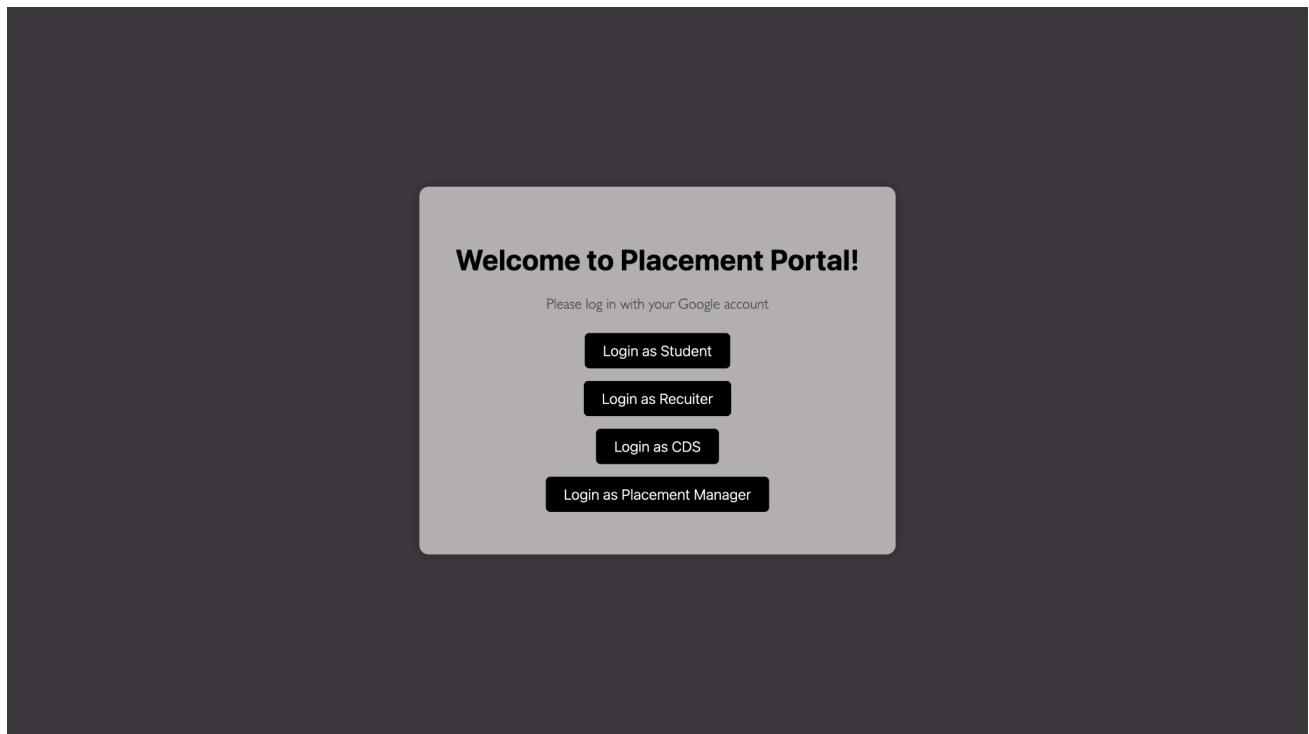
### **3.1 Responsibility of G1:**

**40 Pts.**

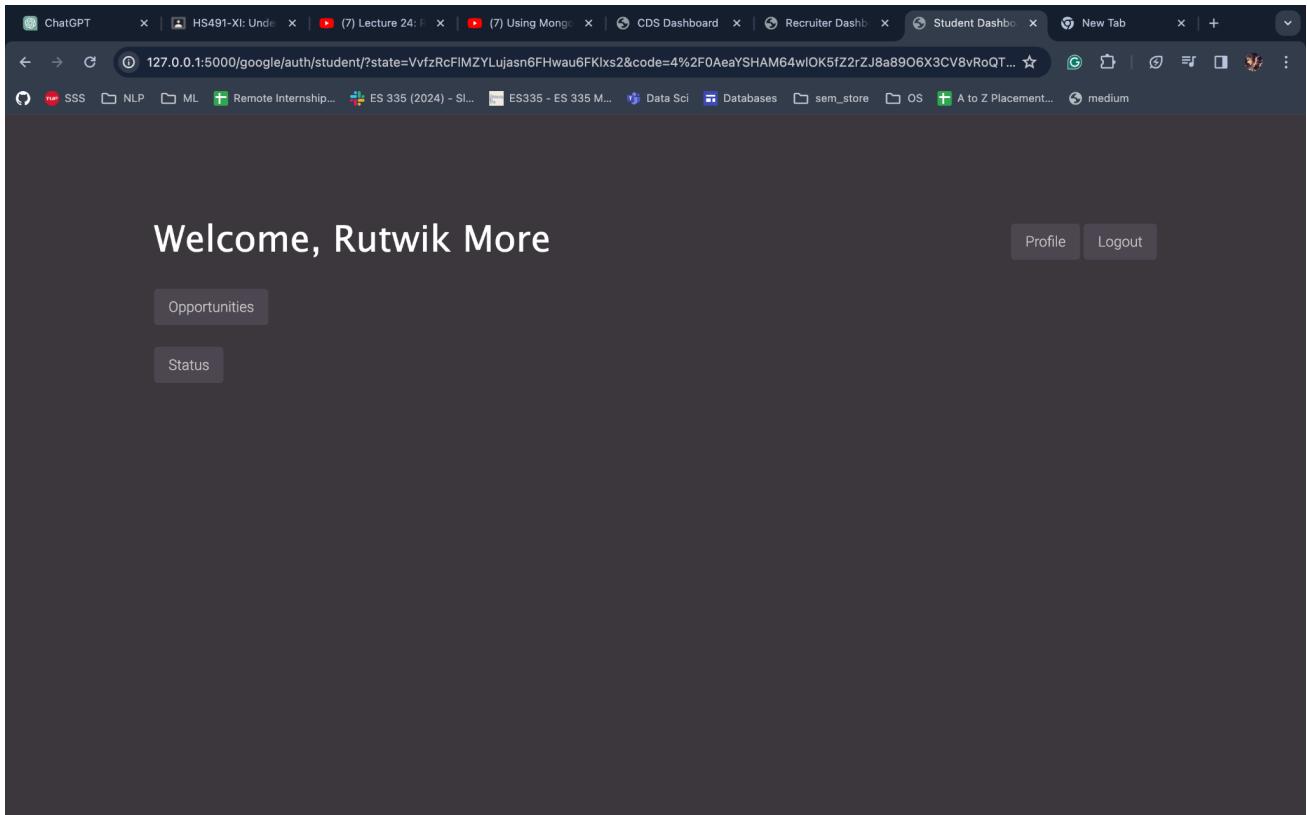
1. The G1 takes two feedbacks from the stakeholders, one initial feedback (on or before 8th April 11:59 PM), and then makes relevant changes as suggested per the first feedback, then final feedback (on or before 11th April 11:59 PM) post changes. The write-up/documentation should have screenshots before 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 (Initial and final feedback forms). **(30 Points)**
2. Attach screenshots of different views [along with a write-up on their privileges] of the database as seen by different classes of users. **(10 Points)**

#### **1. Student**

- Privileges :
  - Students can see the opportunities and apply for them.
  - They can create and edit their profile.
- Login Page



- Student Dashboard



- Student Profile

### Create Profile

**First Name**

**Middle Name**

**Last Name**

**Department**

**Active Backlog**

**Current Year**

**Minor**

**Contact Number**

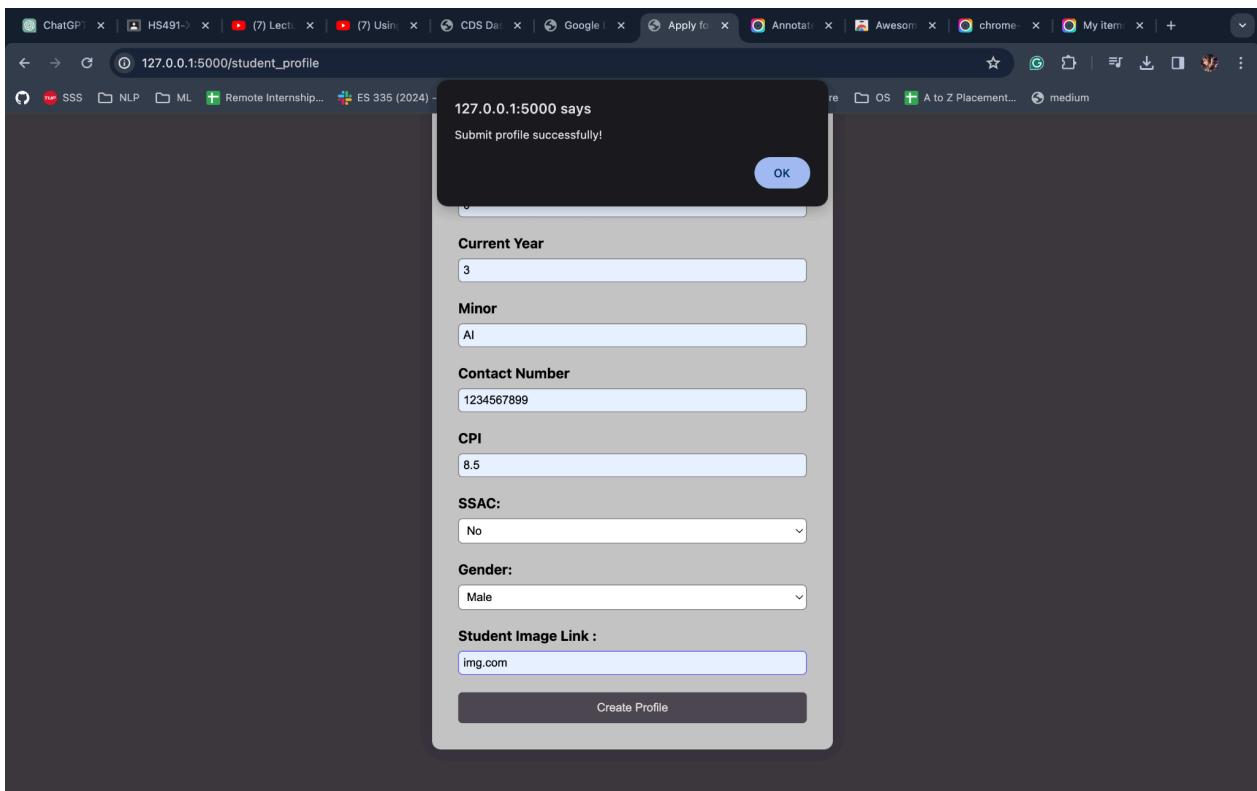
**CPI**

**SSAC\_or\_not:**

**Gender:**

**Student Image Link :**

- Pop Up on Creating Profile successfully



- Profile Page of Student

A screenshot of a web browser window with the URL 127.0.0.1:5000/student\_profile. The page title is "Profile Page". It displays the following information:

Fist name : Rutwik  
Middle name : Babruwan  
Last name : More  
Department: Computer Engineering  
Active Backlog : 0  
Gender : Male  
Current Year : 3  
Minor : AI  
Email : babruwanmore@iitgn.ac.in  
Contact Number : 1234567899  
CPI : 8.5  
SSAC\_or\_not : No

At the bottom left is an "Edit Profile" button, and at the top right are "Dashboard" and "Logout" buttons.

- Student can also edit their profile

### Edit Profile

**First Name**  
Rutwik

**Middle Name**  
Babruwan

**Last Name**  
More

**Department**  
Computer Engineering

**Active Backlog**  
0

**Current Year**  
3

**Minor**  
AI

**Contact Number**  
1234567899

**CPI**  
8.5

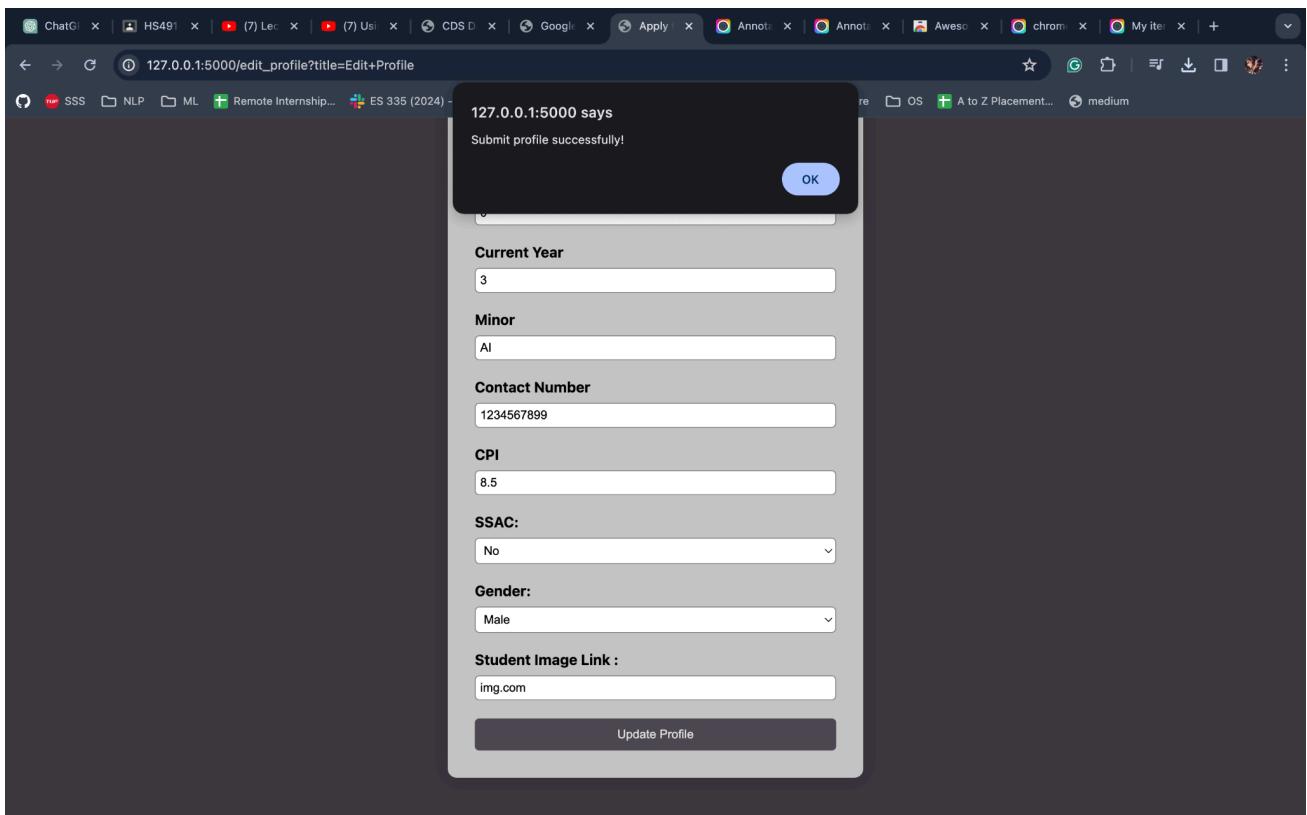
**SSAC:**  
No

**Gender:**  
Male

**Student Image Link :**  
img.com

**Update Profile**

- Pop Up after editing the profile successfully.



- Students can see the listed opportunities.

COMPANY	TITLE	DEPARTMENT	DESCRIPTION	SALARY	CPI REQUIREMENT	NUMBER OF POSITION	MORE DETAILS	APPLY
Tesla	SDE	BTech	file.com	120000	6.0	2	<button>Details</button>	<button>Apply</button>
Devin	ML	BTech MTech	bhoot.com	80000	7.0	4	<button>Details</button>	<button>Apply</button>

- Student can see each opportunity in detail after clicking Details button

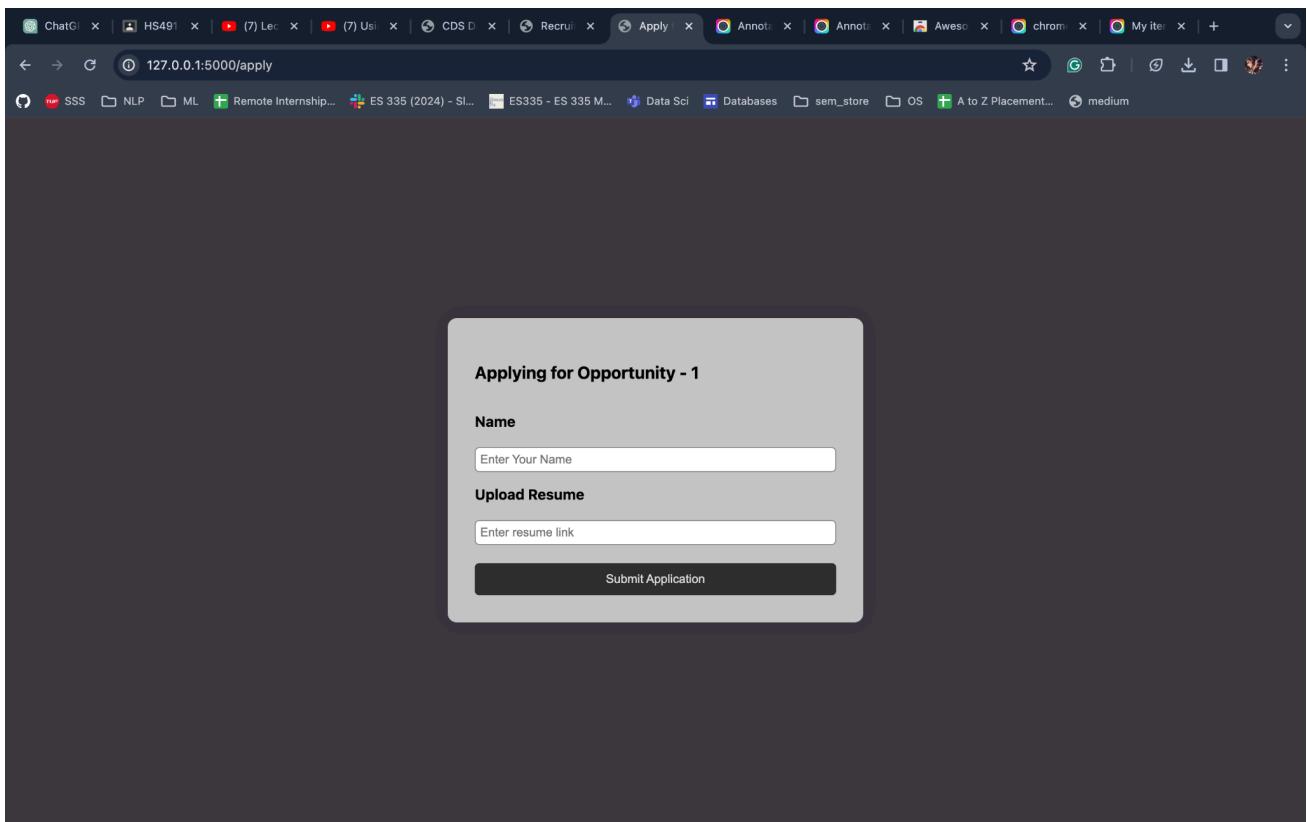
The screenshot shows a web browser window with the URL [127.0.0.1:5000/opportunity\\_details\\_student/1](http://127.0.0.1:5000/opportunity_details_student/1). The page title is "Opportunity Details". On the right, there are "Dashboard" and "Logout" buttons. The main content area displays various details about an opportunity:

- Title : SDE
- Positions : 2
- Salary : 120000
- Requirements : file.com
- CPI requirement : 6.0
- Active Backlog : 0
- Student Year Requirement : 4
- Program requirement : BTech
- Job Description : job.com

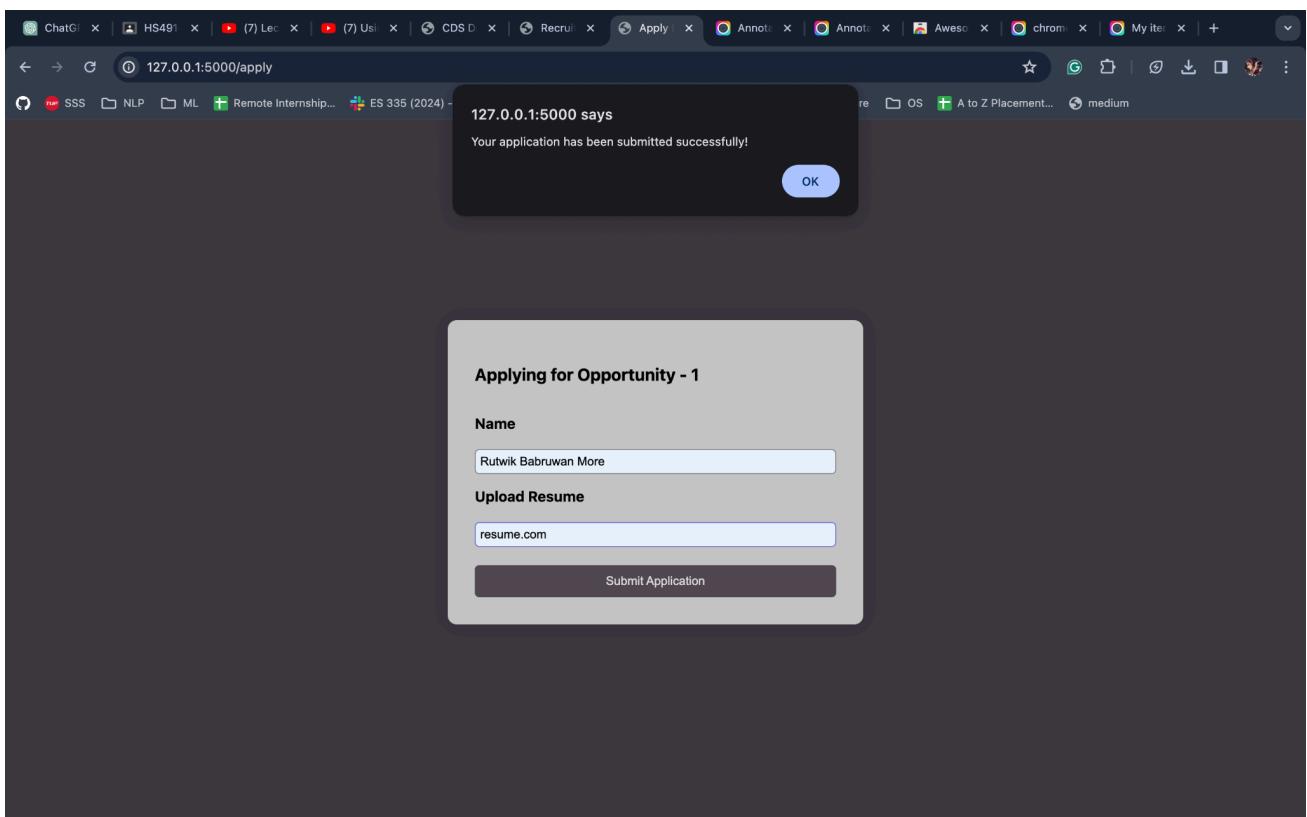
Below this, under "Round Details:", is a table showing two rounds:

ROUND NUMBER	TYPE	VENUE	DATE	START TIME	END TIME
1	Online	Gmeet	2024-04-26	13:20:00	13:30:00
2	Offline	AB 9/101	2024-04-27	14:20:00	15:40:00

- Student can also apply for the opportunity by clicking on Apply button



- Student get pop up after successful application



- Students can see the status of applied opportunities.

The screenshot shows a web browser window with the URL [127.0.0.1:5000/status\\_opp\\_student](http://127.0.0.1:5000/status_opp_student). The page has a dark background and features a table with three columns: COMPANY, TITLE, and STATUS. A single row of data is present: Tesla, SDE, Accepted. At the top right, there are buttons for Dashboard and Logout.

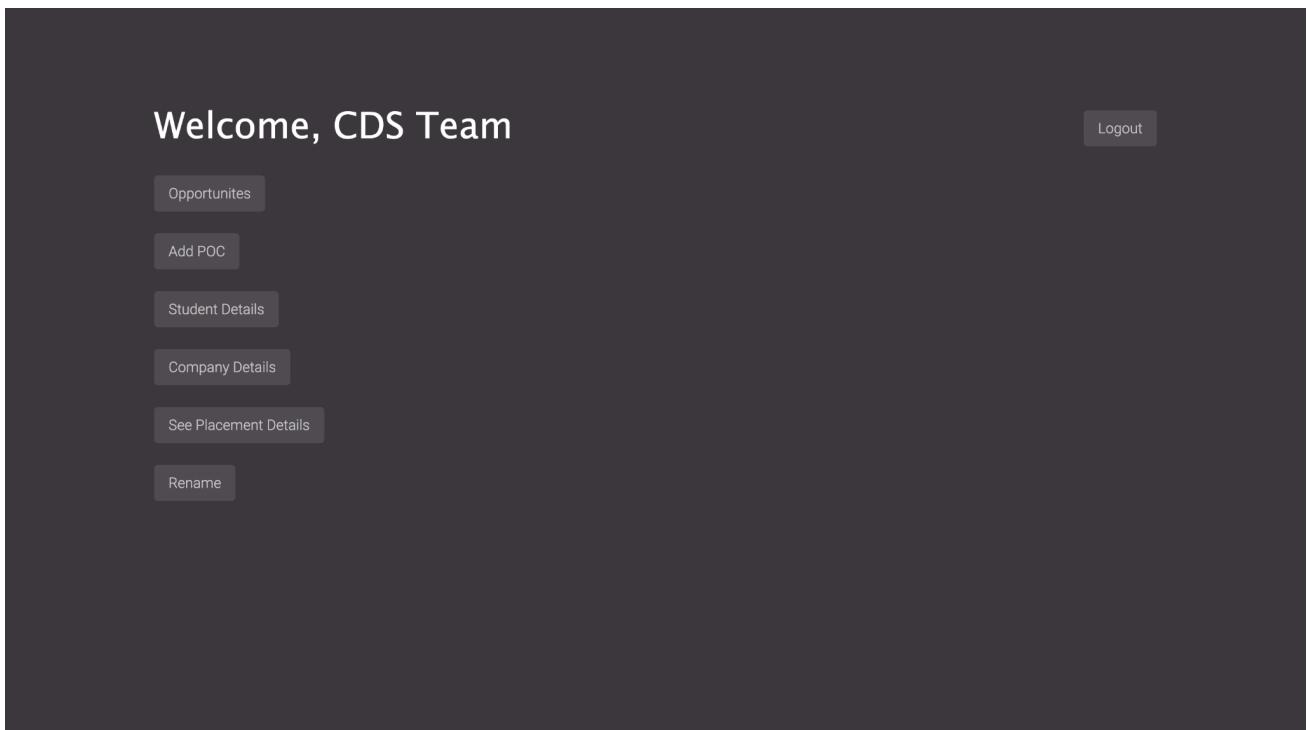
COMPANY	TITLE	STATUS
Tesla	SDE	Accepted

Dashboard    Logout

## 2. CDS Employee:

- Privileges
  - CDS have authority to add the person of contact. Only CDS authorised POC have opportunity to host the opportunity
  - CDS also can change the details of the students
  - They have the view access of the opportunity, placement.

- Dashboard of CDS



- CDS can see all the opportunities listed by the recruiter by clicking on opportunity.

The screenshot shows a web browser window with four tabs open: "Assignment3 - Google Docs", "CDS Dashboard", "Student Dashboard", and "Recruiter Dashboard". The "CDS Dashboard" tab is active, displaying the URL "127.0.0.1:5000/see\_opportunities". The main content area is titled "Opportunities". It contains a table with the following data:

COMPANY	TITLE	DEPARTMENT	DESCRIPTION	SALARY	REMOTE	CPI REQUIREMENT	NUMBER OF POSITION	POC EMAIL
Tesla	SDE	BTech	file.com	120000	Yes	6.0	2	rutwikmore1440@gmail.com
Devin	ML	BTech MTech	bhoot.com	80000	Yes	7.0	4	rutwikmore1440@gmail.com

- CDS can add Person of Contact details by clicking on Add POC button.

The screenshot shows a web browser window with multiple tabs open. The active tab displays the URL "127.0.0.1:5000/add\_poc". A modal dialog box is centered on the screen with the title "Create Profile for POC". The form contains the following fields:

- First Name: Alan
- Middle Name: M
- Last Name: Walker
- Email: rutwikmore1440@gmail.com
- Designation: CEO
- Company Name: IronMan
- Contact Number: 1234567898
- Are you an interviewer? Yes

At the bottom of the form is a "Add Profile" button.

- CDS will get pop up after successfully adding the POC

ChatGPT x | HS49 x | (7) Lec x | (7) Usi x | Create x | Google x | Studen x | Annot x | Annot x | Aweso x | chrome x | My item x | +

127.0.0.1:5000/add\_poc

SSS NLP ML Remote Internship... ES 335 (2024) -

127.0.0.1:5000 says  
Added successfully!

OK

**First Name:**

**Middle Name:**

**Last Name:**

**Email:**

**Designation:**

**Company Name:**

**Contact Number:**

**Are you an interviewer?**

Add Profile



- CDS can see the Details of each student.

The screenshot shows a web application titled "Student Details". At the top right are "Dashboard" and "Logout" buttons. Below is a grid of six student profiles, each in a rounded rectangular box:

- Alice Johnson**  
Electrical Engineering  
alice.johnson@iitgn.ac.in  
[View Details](#)
- Bob Williams**  
Mechanical Engineering  
bob.williams@iitgn.ac.in  
[View Details](#)
- Emily Brown**  
Chemical Engineering  
emily.brown@iitgn.ac.in  
[View Details](#)
- Michael Taylor**  
Civil Engineering  
michael.taylor@iitgn.ac.in  
[View Details](#)
- Sophia Anderson**  
Biomedical Engineering  
sophia.anderson@iitgn.ac.in  
[View Details](#)
- Rutwik More**  
Computer Engineering  
babruwanmore@iitgn.ac.in  
[View Details](#)

- CDS can See student details in detail by clicking on the button 'View Details'.

The screenshot shows a "Student Profile" page for a student named Aditya. At the top right are "Dashboard" and "Logout" buttons. The profile information is listed as follows:

Fist name : Aditya  
Middle name : Santosh  
Last name : Deshmukh  
Department: Mechanical  
Active Backlog : 0  
Gender : Male  
Current Year : 3  
Minor : CS  
Email : deshmukhaditya@iitgn.ac.in  
Contact Number : 09579443667  
CPI : 8.0  
SSAC\_or\_not : No

[Edit details](#)  
[Go Back](#)

- CDS can see placement details of each student.

Placement Details				
STUDENT NAME	COMPANY	SALARY	ROLE	MEDIUM
Adit Kaushik	Skan	90000	Data Scientist	Non_CDS
Gaurav Joshi	Purdue	100000	Researcher	CDS

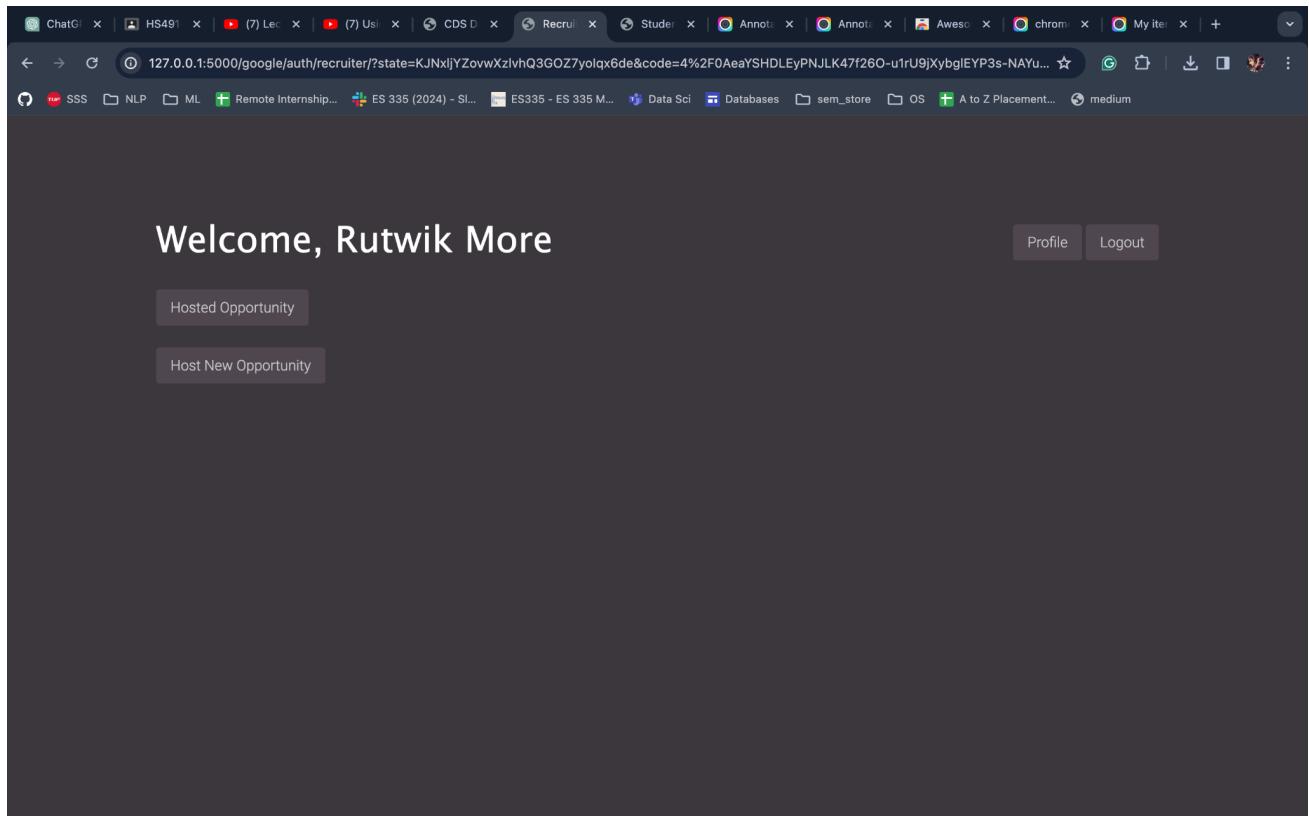
- CDS can see details of all companies.

Company Details				
COMPANY	PERSON OF CONTACT	EMAIL	CONTACT NO	
Avengers	Captain America	rutwik1440@gmail.com	987654321	
IronMan	Alan Walker	rutwikmore1440@gmail.com	1234567898	

## **1. Person of Contact (POC):**

- Privileges :
  - Have access to add the opportunity
  - Accepts, reject the student application

### - Dashboard of Recruiter



- Recruiter can host new Opportunities.

### Create Opportunity

**Opportunity Title:**  
SDE

**Company Name:**  
Tesla

**Number of Positions:**  
2

**Specific Requirements File Link:**  
file.com

**Minimum CPI Requirement:**  
6

**Number of Active Backlogs:**  
0

**Student Year Requirement:**  
4

**Program Requirement:**  
BTech

**Job Description File Link:**  
job.com

**Salary:**  
120000

**Number of Rounds:**  
2

**Round 1**

**Round Type:**  
Online

**Round Date:**  
26/04/2024

**Venue:**  
Gmeet

**Start Time:**  
01:20 PM

**End Time:**  
01:30 PM

**Round 2**

**Round Type:**  
Offline

**Round Date:**  
27/04/2024

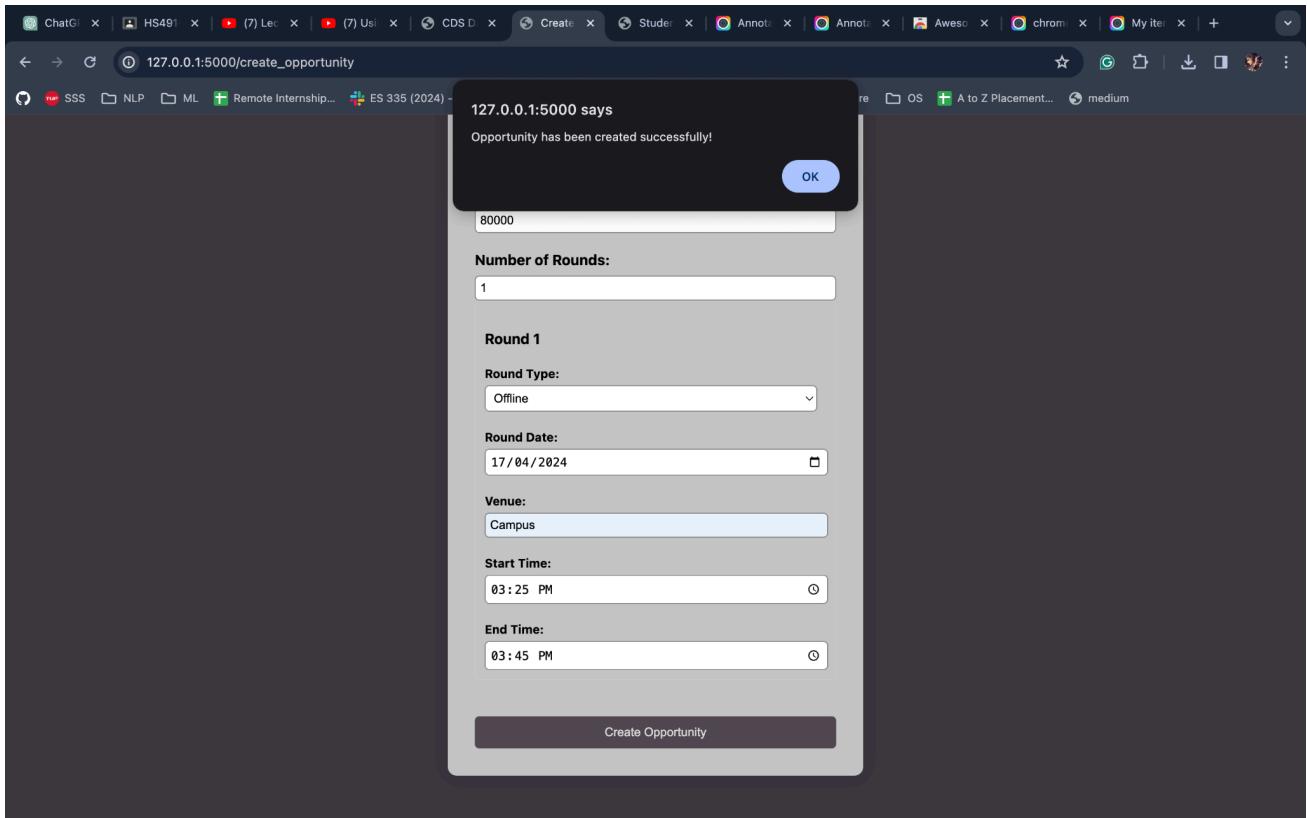
**Venue:**  
AB 9/101

**Start Time:**  
02:20 PM

**End Time:**  
03:40 PM

**Create Opportunity**

- Recruiter will get pop after successfully adding the opportunity.



- Recruiter can see opportunities details of hosted opportunities.

The screenshot shows a dark-themed web application interface. The title of the page is "Opportunity Details". On the right side, there are "Dashboard" and "Logout" buttons. The main content area displays various details about a job opportunity, including:

- Title : SDE
- Positions : 2
- Salary : 120000
- Requirements : file.com
- CPI requirement : 6.0
- Active Backlog : 0
- Student Year Requirement : 4
- Program requirement : BTech
- Job Description : job.com

Below this, under "Round Details:", there is a table showing two rounds:

ROUND NUMBER	TYPE	VENUE	DATE	START TIME	END TIME
1	Online	Gmeet	2024-04-26	13:20:00	13:30:00
2	Offline	AB 9/101	2024-04-27	14:20:00	15:40:00

- Recruiter can view applications submitted by student.

TITLE	POSITIONS	SALARY	APPLICATIONS	OPPORTUNITY DETAILS	DELETE OPPORTUNITY
SDE	2	120000	<a href="#">View Applications</a>	<a href="#">Opportunity Details</a>	<a href="#">Delete Opportunity</a>
ML	4	80000	<a href="#">View Applications</a>	<a href="#">Opportunity Details</a>	<a href="#">Delete Opportunity</a>

- Recruiter can change the status of opportunity of student.

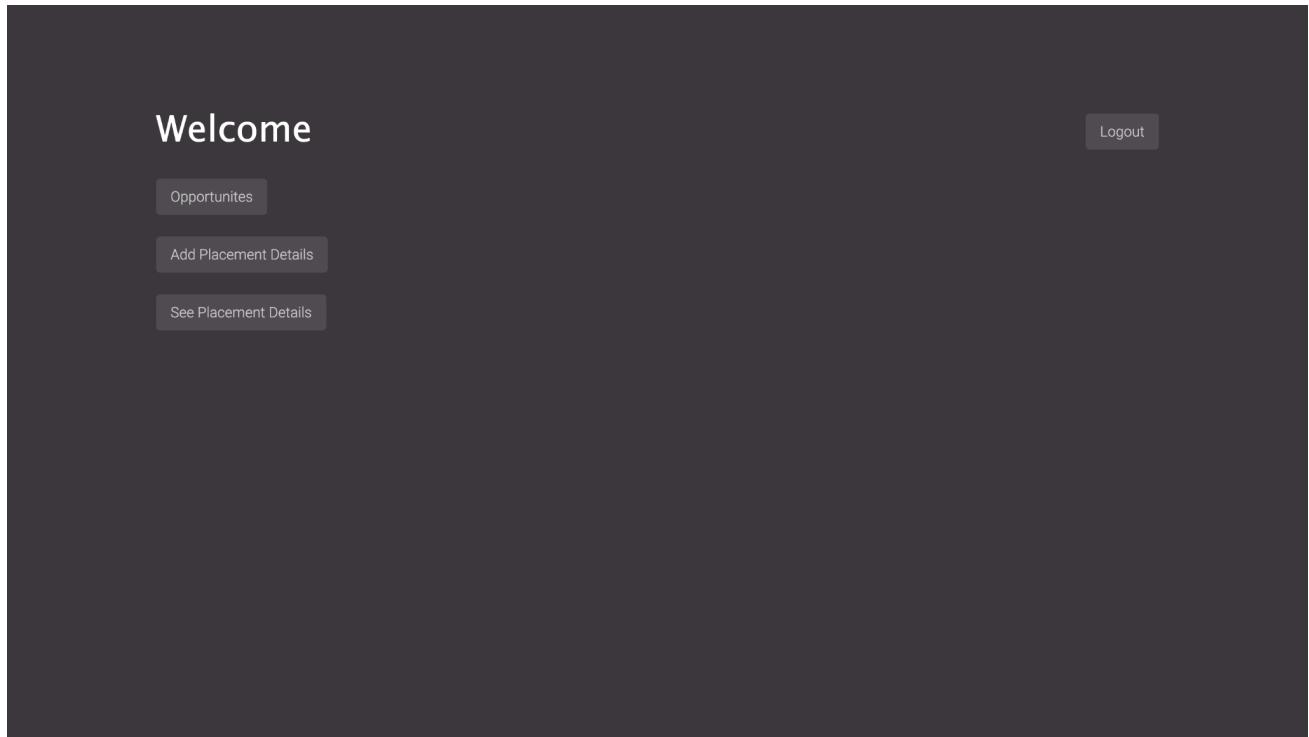
STUDENT NAME	RESUME
Rutwik More	resume.com

Pending  
Round : 1  
Round : 2  
✓ Accepted  
Rejected

Update Status

#### 4. Placement manager view

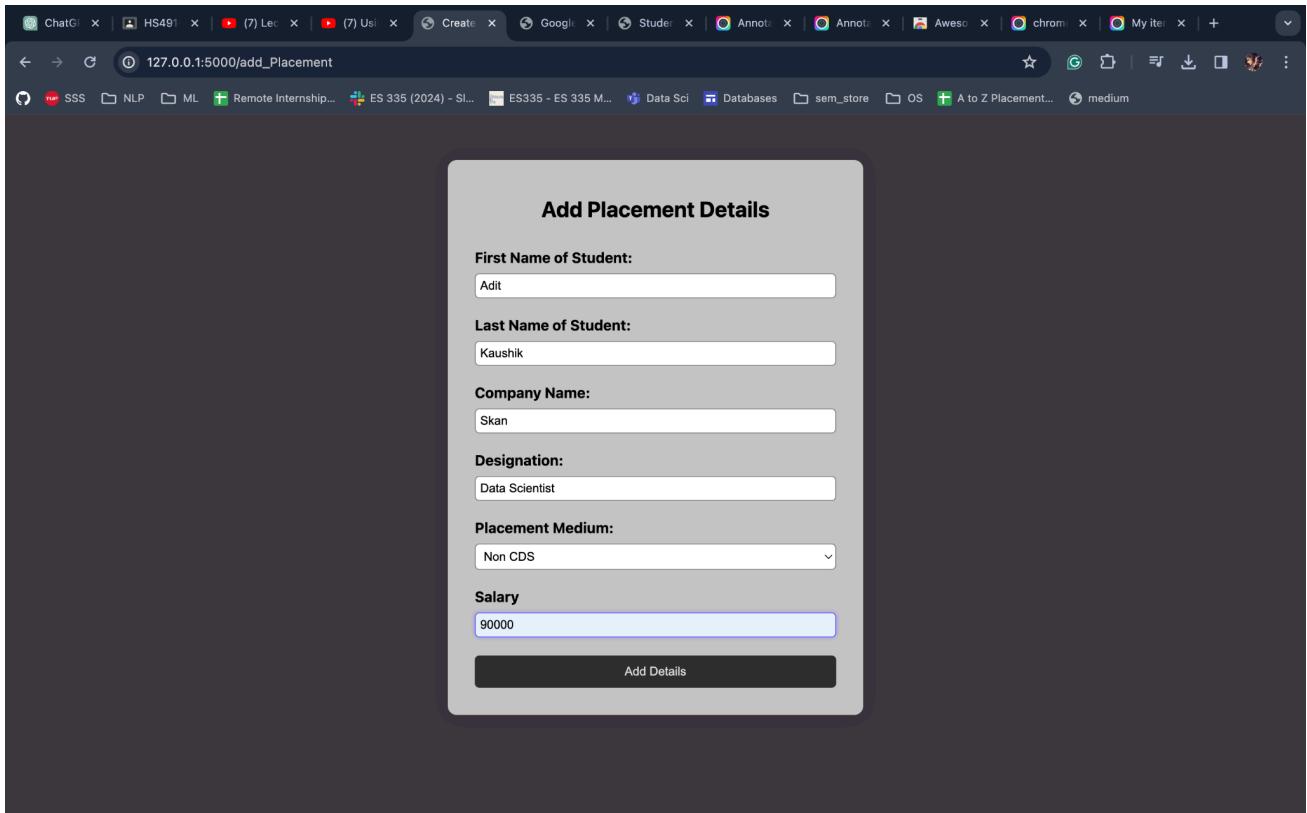
- Privileges:
  - Placement manager have access to add the placement details of the students
  - They can see the opportunities and also placement details add



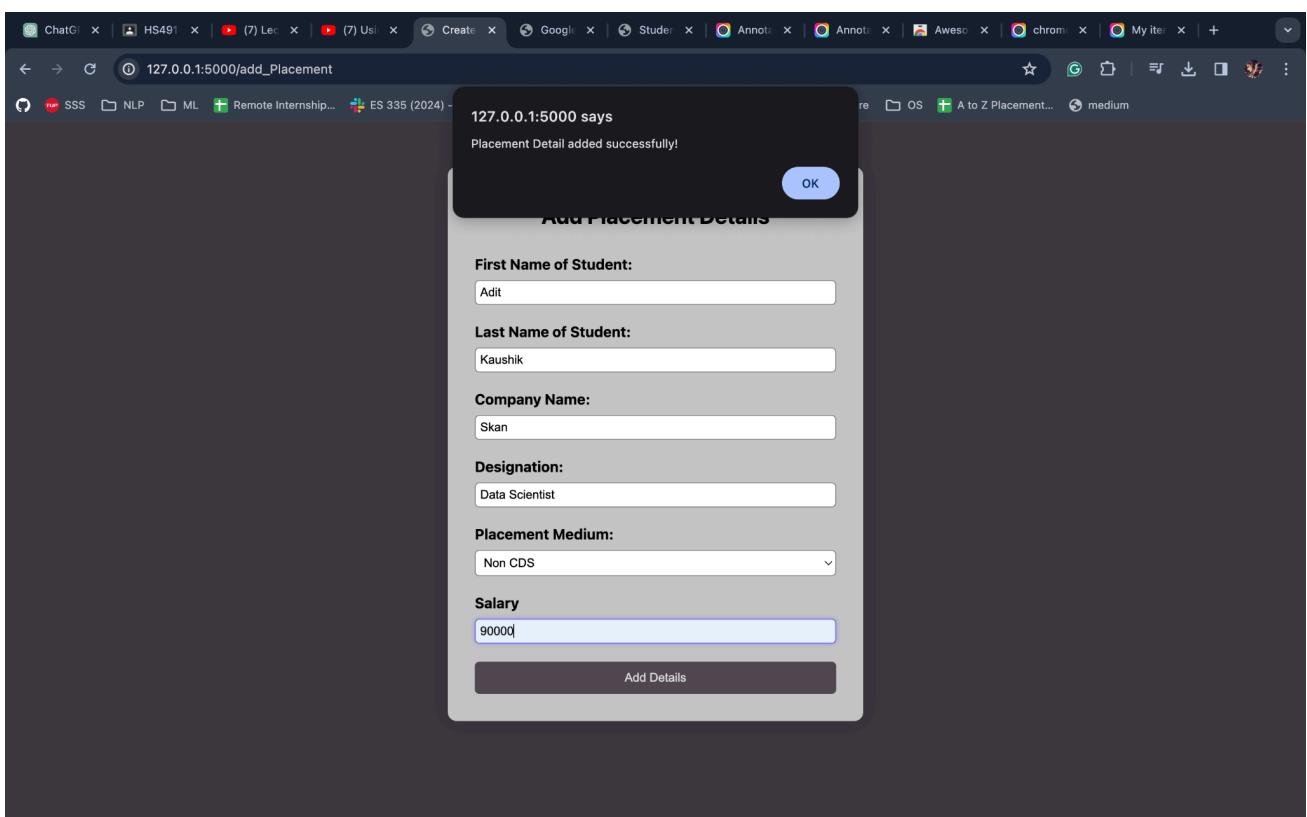
- Placement manager can view the opportunities available

Opportunities									Download	Dashboard	Logout
COMPANY	TITLE	DEPARTMENT	DESCRIPTION	SALARY	REMOTE	CPI REQUIREMENT	NUMBER OF POSITION	POC EMAIL			
NextGen	Machine Learning	Btech	file.txt	80000	Yes	7.0	4	adityadeshmukh932@gmail.com			
OpenAI	SDE	Btech	file.txt	90000	Yes	4.0	4	adityadeshmukh932@gmail.com			
Boston Dynamics	Data Science	Btech	file.txt	70000	Yes	7.0	4	adityadeshmukh932@gmail.com			

- The Placement Manager can add placement details of each student.



- CDS will pop up after adding the details successfully.



- Placement manager can see the details of the placed students

Placement Details				
STUDENT NAME	COMPANY	SALARY	ROLE	MEDIUM
Aditya Deshmukh	Next	80000	SDE	CDS
Zen Tau	ACC	50000	ML	Non_CDS
Michael Thompson	BrightFuture	50000	Marketing Specialist	CDS

### 3.2 Responsibility of G2:

**40 Pts.**

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)**
- To show the multi-user access, we gave the editing access of student profile to both the students and the CDS user
  - Concurrent editing is a common requirement in systems where multiple users need to access and modify shared resources simultaneously. However, allowing concurrent editing without proper synchronization mechanisms can lead to conflicts where one user's changes overwrite another's, resulting in data loss or inconsistency.
  - Student editing their profile.

# Profile Page

[Dashboard](#) [Logout](#)

First name : Aditya

Middle name : Santosh

Last name : Deshmukh

Department: Mechanical

Active Backlog : 0

Gender : Male

Current Year : 3

Minor : CS

Contact Number : 09579443667

CPI : 8.0

SSAC : No

[Edit Profile](#)

- CDS editing student profile

## Student Details

[Download](#) [Dashboard](#) [Logout](#)

### Alice Johnson

Electrical Engineering

alice.johnson@iitgn.ac.in

[View Details](#)

### Bob Williams

Mechanical Engineering

bob.williams@iitgn.ac.in

[View Details](#)

### Emily Brown

Chemical Engineering

emily.brown@iitgn.ac.in

[View Details](#)

### Michael Taylor

Civil Engineering

michael.taylor@iitgn.ac.in

[View Details](#)

### Sophia Anderson

Biomedical Engineering

sophia.anderson@iitgn.ac.in

[View Details](#)

### Aditya Deshmukh

Mechanical

deshmukhaditya@iitgn.ac.in

[View Details](#)

# Student Profile

[Dashboard](#) [Logout](#)

Fist name : Aditya  
Middle name : Santosh  
Last name : Deshmukh  
Department: Mechanical  
Active Backlog : 0  
Gender : Male  
Current Year : 3  
Minor : CS  
Email : deshmukhaditya@iitgn.ac.in  
Contact Number : 09579443667  
CPI : 8.0  
SSAC\_or\_not : No  
[Edit details](#)

Go Back

- In the system, locks are implemented to manage concurrent editing of student profiles effectively. Each profile is associated with a boolean variable `is\_locked`, denoting whether it is currently being edited by another user (`True`) or not (`False`).
- When a user initiates editing, the system checks the lock status; if the profile is locked, the user is notified to try again later, preventing conflicts. Upon verifying that the profile is available for editing, the system sets the `is\_locked` variable to `True`, effectively locking the profile and preventing concurrent access.
- After the user completes the edits and commits them to the database, the lock is released by resetting `is\_locked` to `False`, enabling other users to edit the profile.
- This locking mechanism ensures data integrity by preventing simultaneous conflicting modifications, thereby maintaining the accuracy and consistency of the student profiles.

```
is_locked = cur.fetchone()['is_locked']
if is_locked:
    flash("Another user is currently editing this profile. Please try again later.", "error")
    return render_template('student/dashboard.html')

cur.execute("UPDATE Student SET is_locked = TRUE WHERE Student_Email_Id = %s", (email,))
mysql.connection.commit()

student_id = existing_user[0]
query = "UPDATE Student SET Student_First_Name = %s, Student_Middle_Name = %s, Student_Last_Name = %s, Active_Backlog = %s, Contact_Number = %s, SSAC_or_not = %s WHERE Student_Email_Id = %s"
values = (firstName, middleName, lastName, activeBacklog, contactNumber, SSAC_or_not, email)
mysql.connection.commit()
flash("Profile updated successfully!", "success")

cur.execute("UPDATE Student SET is_locked = FALSE WHERE Student_Email_Id = %s", (email,))
mysql.connection.commit()
```

### **3.2 Responsibility of G2:**

**40 Pts.**

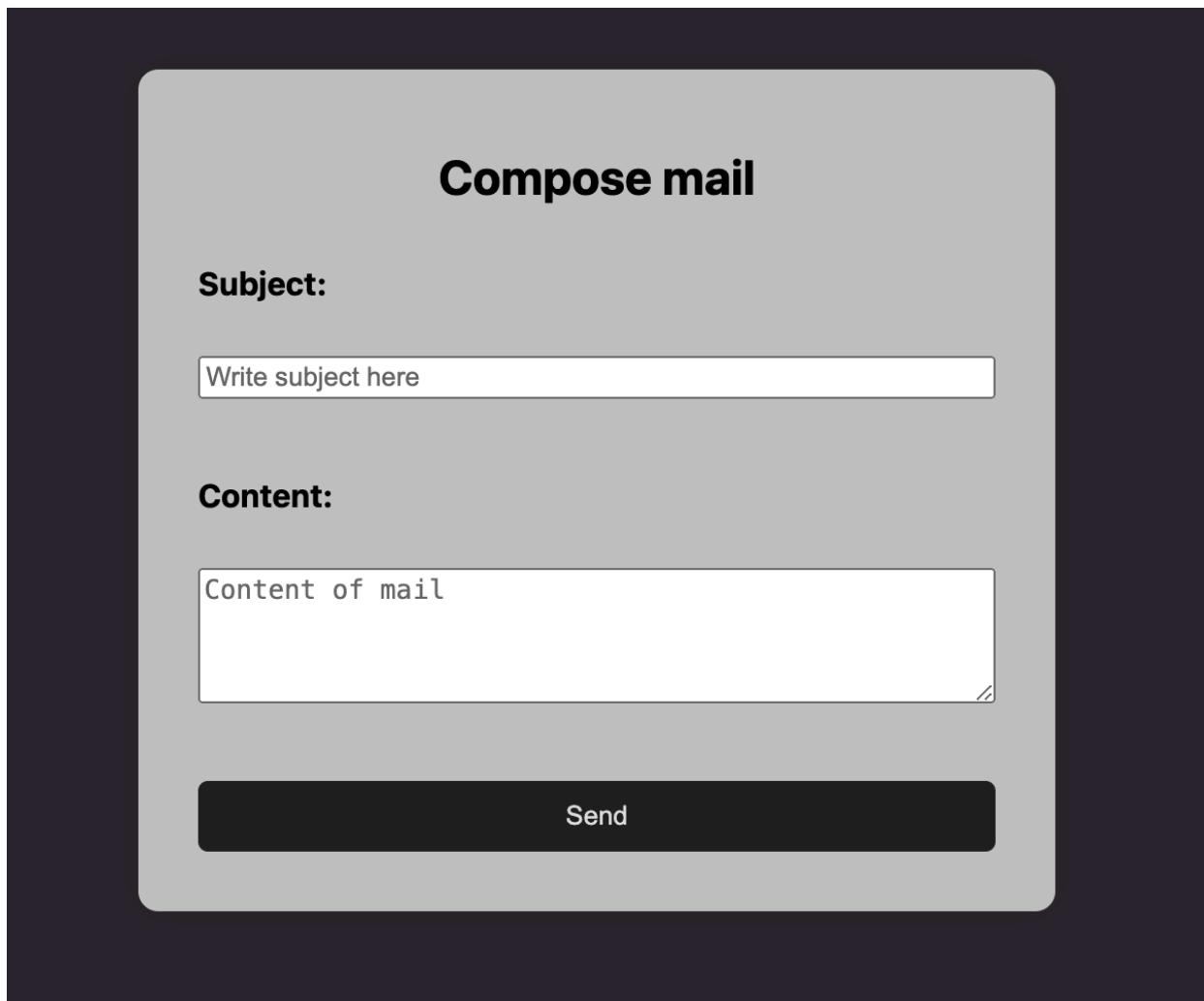
2. Implement the changes in the database as per the feedback received from stakeholders. **(20 Points)**

### **Changes made after the first feedback-**

1. Mailing Feature

In our discussion with stakeholders, they suggested that it would be very inconvenient for CDS members to communicate with the recruiters through the mail. They will have to search for the email id of the recruiter and go to the Gmail app and then send the mail to them.

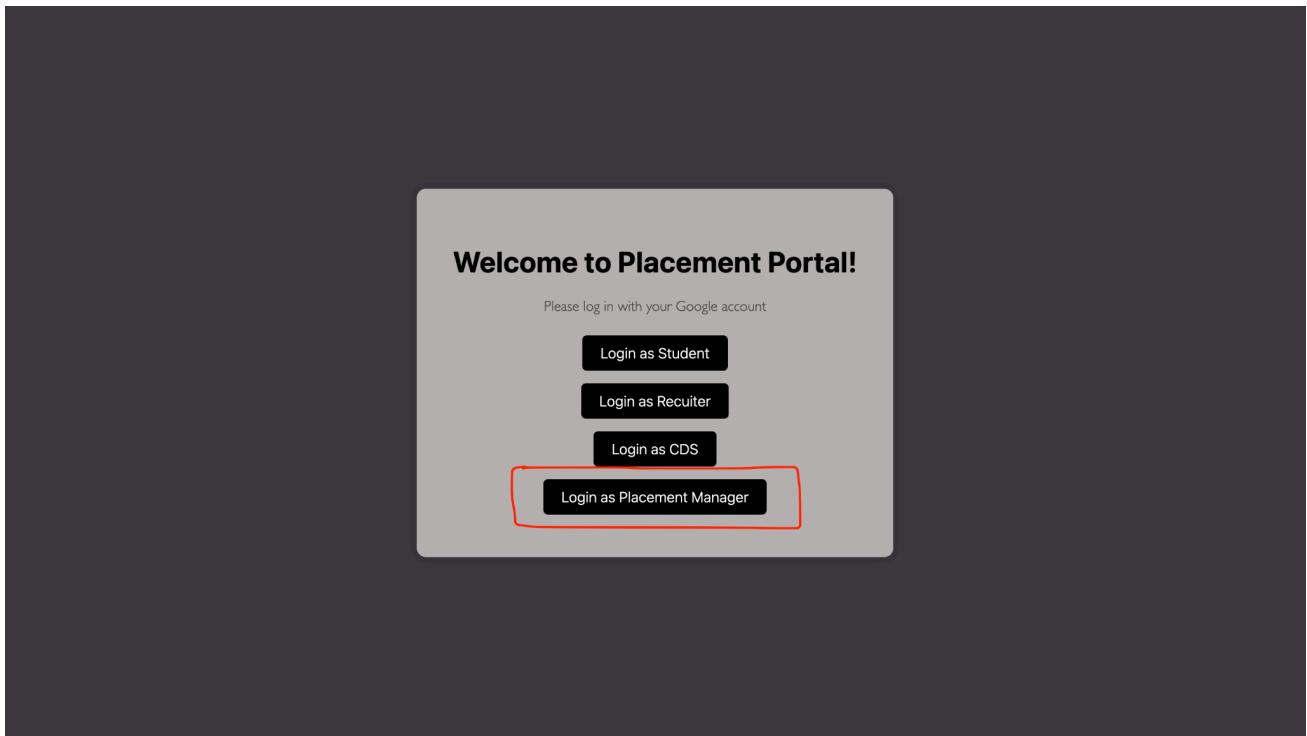
Instead of that, for the convenience of the CDS members, there should be a mail button alongside the name of the recruiter so that they can send the mail within our webapp.

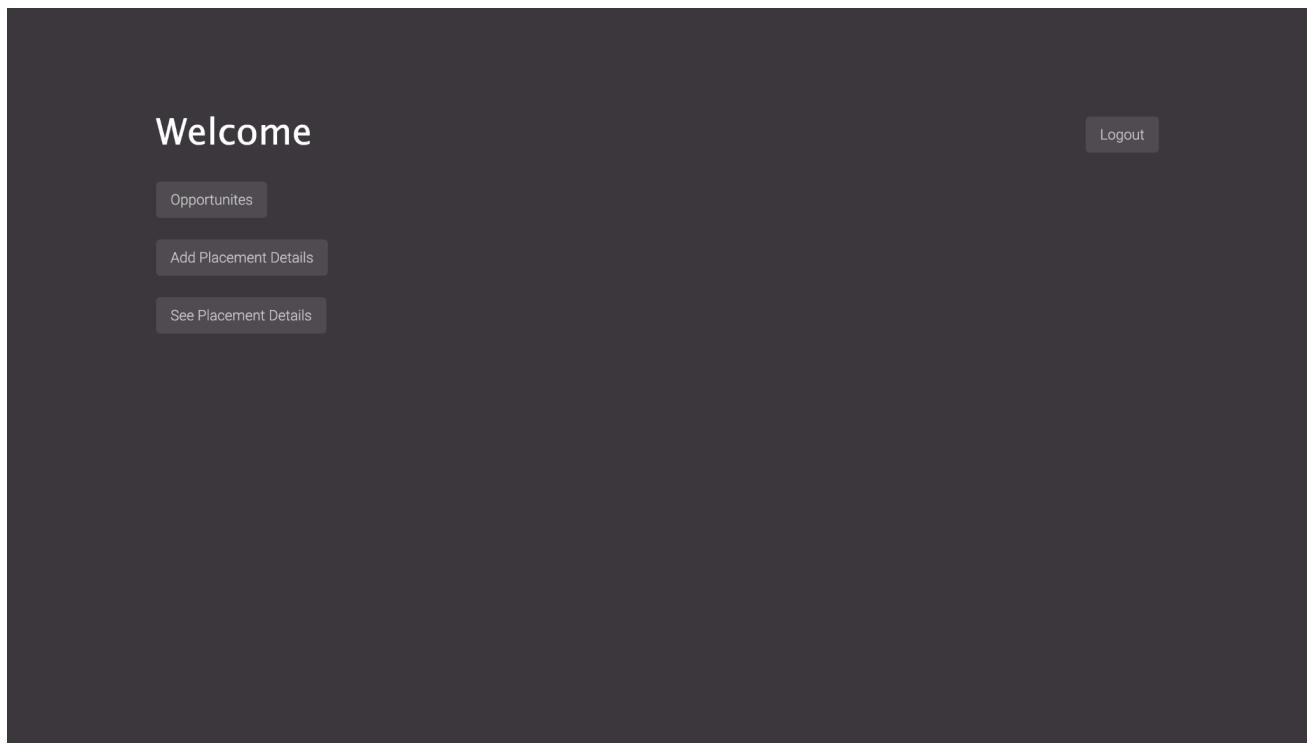


Company Details				<a href="#">Download</a>	<a href="#">Dashboard</a>	<a href="#">Logout</a>
COMPANY	PERSON OF CONTACT	EMAIL	CONTACT NO			
Avengers	Captain America	rutwik1440@gmail.com	987654321	<a href="#">Mail</a>		
IronMan	Alan Walker	rutwikmore1440@gmail.com	1234567898	<a href="#">Mail</a>		
Tesla	Tony Stark	tonystark@gmail.com	1234567899	<a href="#">Mail</a>		

## 2. Add Placement manager - Give access of adding placement details

The stakeholder proposed augmenting the CDS framework by introducing a dedicated placement manager role responsible for entering placement specifics. This enhancement not only enriches the hierarchical layout of the system but also aims to optimize the placement procedure, granting designated individuals the authority to proficiently overseas placement data. Moreover, it fosters a more streamlined approach to managing placements within the framework, ensuring efficient coordination and organization.





- Add Placement Details

The image displays a modal dialog titled "Add Placement Details". The form contains the following fields:

- First Name of Student:** An input field with placeholder text "Enter First Name".
- Last Name of Student:** An input field with placeholder text "Enter Last Name".
- Company Name:** An input field with placeholder text "Enter Company Name".
- Designation:** An input field with placeholder text "Enter Designation".
- Placement Medium:** A dropdown menu with placeholder text "Select Medium".
- Salary**: A section header followed by an input field with placeholder text "Enter Salary".

- See placement details

Placement Details				
STUDENT NAME	COMPANY	SALARY	ROLE	MEDIUM
Aditya Deshmukh	Next	80000	SDE	CDS
Zen Tau	ACC	50000	ML	Non_CDS
Michael Thompson	BrightFuture	50000	Marketing Specialist	CDS

### 3. Search Feature

Based on feedback from key stakeholders, we have decided to implement search functionality across our product to enhance the user experience. Wherever users can query data, we will now provide a search option to allow them to easily find the specific information they need.

We anticipate this search feature will have a significant positive impact by making it easier for users to pinpoint the critical data they need when and where they need it. Our goal is to help them make well-informed decisions quickly by providing seamless access to the information most relevant to each situation.

Search

## Opportunities

COMPANY	TITLE	DEPARTMENT	DESCRIPTION	SALARY	REMOTE	CPI REQUIREMENT	NUMBER OF POSITION	POC EMAIL
NextGen	Machine Learning	Btech	file.txt	80000	Yes	7.0	4	adityadeshmukh932@gmail.com
OpenAI	SDE	Btech	file.txt	90000	Yes	4.0	4	adityadeshmukh932@gmail.com
Boston Dynamics	Data Science	Btech	file.txt	70000	Yes	7.0	4	adityadeshmukh932@gmail.com

[Download](#) [Dashboard](#) [Logout](#)

Search

## Opportunities

COMPANY	TITLE	DEPARTMENT	DESCRIPTION	SALARY	REMOTE	CPI REQUIREMENT	NUMBER OF POSITION	POC EMAIL
NextGen	Machine Learning	Btech	file.txt	80000	Yes	7.0	4	adityadeshmukh932@gmail.com
OpenAI	SDE	Btech	file.txt	90000	Yes	4.0	4	adityadeshmukh932@gmail.com
Boston Dynamics	Data Science	Btech	file.txt	70000	Yes	7.0	4	adityadeshmukh932@gmail.com

[Download](#) [Dashboard](#) [Logout](#)

If the query matches it give corresponding results else it gives

## Opportunities

COMPANY	TITLE	DEPARTMENT	DESCRIPTION	SALARY	REMOTE	CPI REQUIREMENT	NUMBER OF POSITION	POC EMAIL
NextGen	Machine Learning	Btech	file.txt	80000	Yes	7.0	4	adityadeshmukh932@gmail.com

## Opportunities

No opportunities available at the moment for the searched query : no

## Changes made after the final feedback

### 1. Download tables as CSV

In our discussion with stakeholders, we got feedback that lists such as company details, and student placement details they need them in CSV format for their convenience. So here we have added the feature of downloading the tables in CSV format.

So we have added a button on top of the page. After we click the button CSV file is downloaded.

The screenshot shows a dark-themed web application titled "Placement Details". At the top right are three buttons: "Download" (circled in pink), "Dashboard", and "Logout". Below the title is a table with columns: STUDENT NAME, COMPANY, SALARY, ROLE, and MEDIUM. Two rows of data are visible: one for Adit Kaushik (Skan, 90000, Data Scientist, Non\_CDS) and one for Gaurav Joshi (Purdue, 100000, Researcher, CDS).

STUDENT NAME	COMPANY	SALARY	ROLE	MEDIUM
Adit Kaushik	Skan	90000	Data Scientist	Non_CDS
Gaurav Joshi	Purdue	100000	Researcher	CDS

Here we can see that CSV has been downloaded.

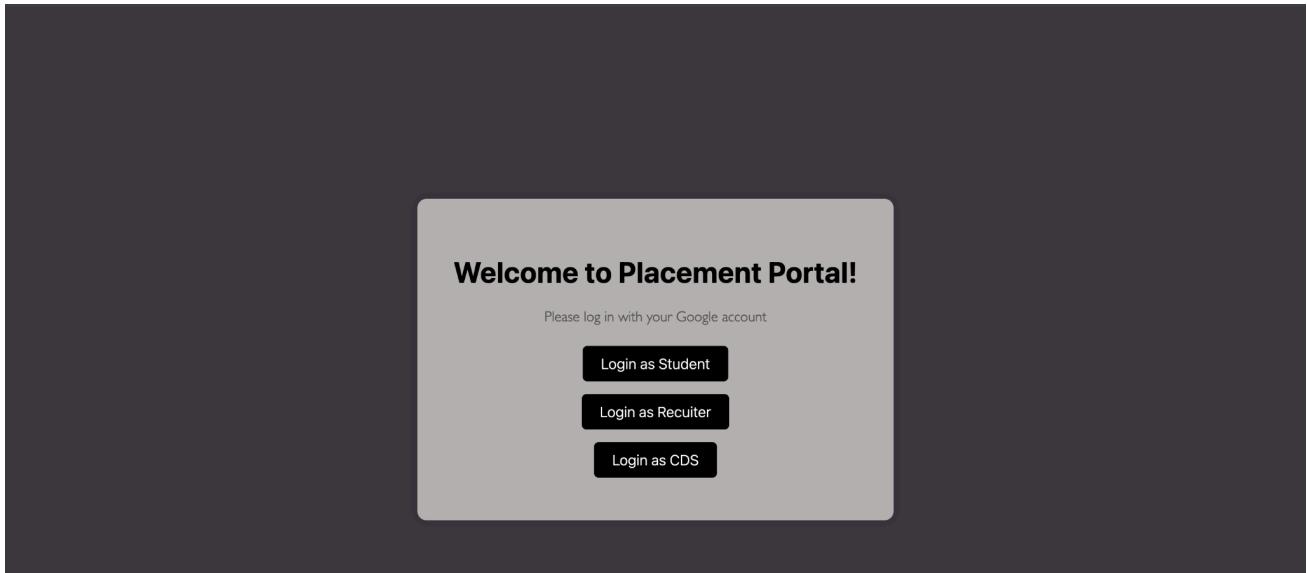
The screenshot shows the same "Placement Details" page. A download confirmation message is displayed in a dark overlay window in the browser's sidebar, indicating two files were downloaded: "placement\_details (1).csv" (128 B) and "videos\_SQLI.webm" (1,078 KB, 6 hours ago). The main content of the page remains the same, showing the student placement table.

STUDENT NAME	COMPANY	SALARY	ROLE	MEDIUM
Adit Kaushik	Skan	90000	Data Scientist	Non_CDS
Gaurav Joshi	Purdue	100000	Researcher	CDS

### 3.2 Responsibility of G2:

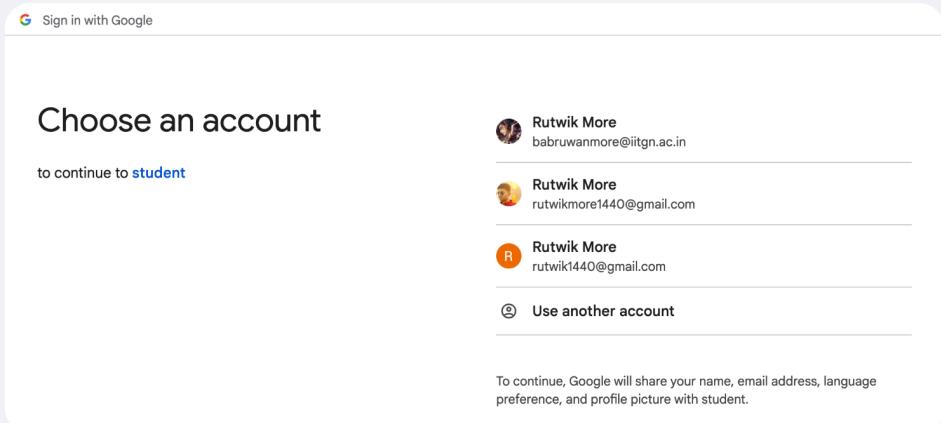
3. Add Google authentication for login and registration. (only IITGN users can login and register) **(10 Points)**

We have added Google authentication functionality. Where only IITGN users can login.



 Google sign-in has a new look  
We've improved the sign-in page with a more modern design

[Learn more](#) [Dismiss](#)



The image shows the Google sign-in interface. At the top left is the "Sign in with Google" button. Below it, the heading "Choose an account" is centered. Underneath, the text "to continue to student" is visible. A list of accounts is shown, each with a small profile picture, the name "Rutwik More", and the email address. The first account is "babruwanmore@iitgn.ac.in". The second is "rutwikmore1440@gmail.com". The third is "rutwik1440@gmail.com". At the bottom of the list is a link "Use another account". A note at the bottom states: "To continue, Google will share your name, email address, language preference, and profile picture with student." At the very bottom of the interface, there are links for "English (United States)", "Help", "Privacy", and "Terms".

### 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)*

#### 1. SQL Injection

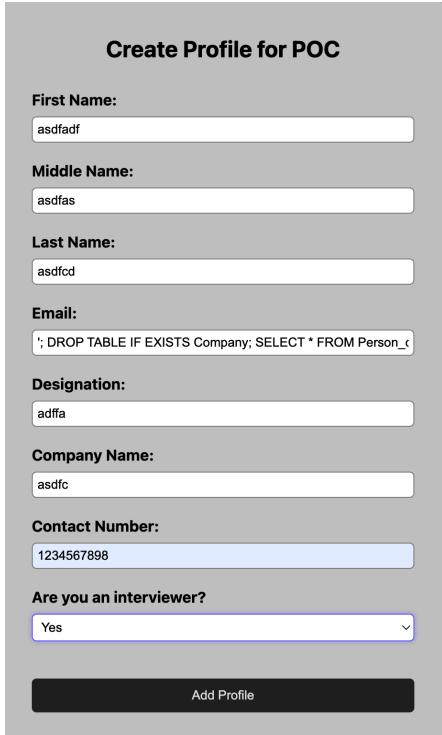
SQL injection is a common type of web attack in which an attacker inserts malicious SQL statements into an entry field, such as a login form or a search bar. This can result in the attacker being able to view, modify, or delete data from the application's database.

Steps to recreate the attack:

1) While creating the profile for Person of Contact, we can create a profile of POC with any email address. For example [rutwikmore1440@gmail.com](mailto:rutwikmore1440@gmail.com)

2) Then after creating the profile we create another new profile and while adding other entries we add following sql query in Email box input

`'; DROP TABLE IF EXISTS Company; SELECT * FROM Person_of_Contact WHERE Poc_Email_Id ='rutwikmore1440@gmail.com'`



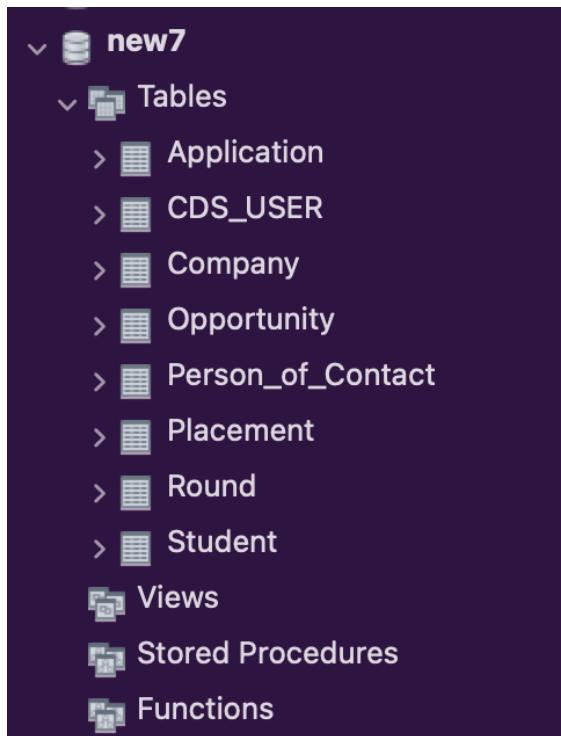
The screenshot shows a 'Create Profile for POC' form with the following fields and their values:

- First Name:** asdfadf
- Middle Name:** asdfas
- Last Name:** asdfcd
- Email:** `'; DROP TABLE IF EXISTS Company; SELECT * FROM Person_of_Contact WHERE Poc_Email_Id ='rutwikmore1440@gmail.com'`
- Designation:** adffa
- Company Name:** asdfc
- Contact Number:** 1234567898
- Are you an interviewer?** Yes

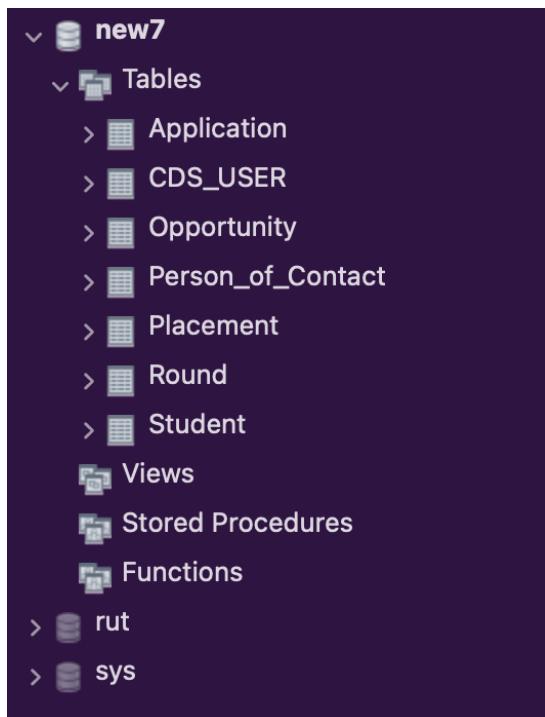
At the bottom is a large black 'Add Profile' button.

3) Here you can see email Id with email [rutwikmore1440@gmail.com](mailto:rutwikmore1440@gmail.com) is present.

4) Here we can see that Company table is present in the database.



5) But after submitting the form with SQL injecting query DROP TABLE IF EXISTS Company gets executed and and user can delete any table from the database. Or User can even delete all the entire database. Here in below picture we can see that the Company table got deleted.



We have vulnerability in our flask backend code in the line

```
cur.execute("SELECT Poc_Email_Id FROM Person_of_Contact WHERE Poc_Email_Id = "+emailPOC+" ")
```

It shows that Attack can inject any code in field of emailPOC.

```
emailPOC = request.form.get('Email')
cur = mysql.connection.cursor()
cur.execute("SELECT Poc_Email_Id FROM Person_of_Contact WHERE Poc_Email_Id = '"+emailPOC+"' ")
existing_user = cur.fetchone()

if existing_user:
    query = "UPDATE Person_of_Contact SET Employee_First_Name = %s, Employee_Middle_Name = %s, Employee_Last_Name = %s, Designation = %s, CompanyName = %s, Interviewer = %s, ContactNumber = %s"
    values = (firstName, middleName, lastName, designation, companyName, interviewer, contactNumber, e)
```

### Defence against this attack:

To protect against such attacks, we should use parameterized queries or prepared statements to pass user input as a parameter instead of embedding it directly into the SQL query string. Here code shows how to do it safely:

```
emailPOC = request.form.get('Email')
cur = mysql.connection.cursor()
print(emailPOC)
cur.execute("SELECT Poc_Email_Id FROM Person_of_Contact WHERE Poc_Email_Id = %s", (emailPOC,))
existing_user = cur.fetchone()
```

By using a parameterized query, you prevent any SQL injection attacks because the user's input is treated as a value rather than part of the SQL query.

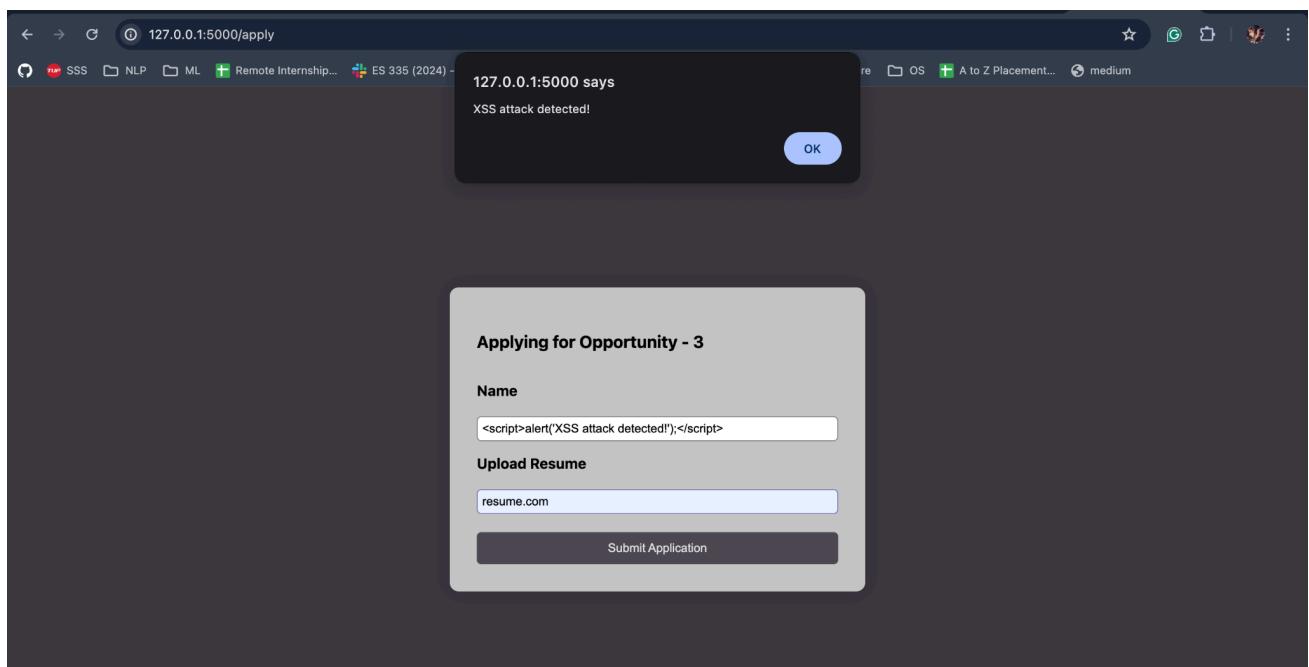
## 2. Cross-site scripting (XSS) attack

XSS is an attack in which we input some malicious JavaScript in the input box, which then gets embedded in the HTML and later gets executed when the infected page is opened.

If we enter the following script in the Input Box of the name

```
<script>alert("XSS attack detected!");</script>
```

Here we get the pop-up saying “XSS attack detected!”. This shows that JavaScript can be executed on the server using the input box. This means DOM elements and other data can be accessed using XSS.



So there is vulnerability present in our backend flask code. Here we can see those line:

```
var name = $('#name').val();
```

```
var resumeLink = $('#resumeLink').val();
```

```
var postData = {  
    name:name,  
    resume:resumeLink  
};
```

```
$('#apply-for-opportunity').submit(function (event) {  
    event.preventDefault();  
    var name = $('#name').val();  
    var resumeLink = $('#resumeLink').val();  
  
    var postData = {  
        name: name,  
        resume: resumeLink  
    };
```

### Defense against this attack:

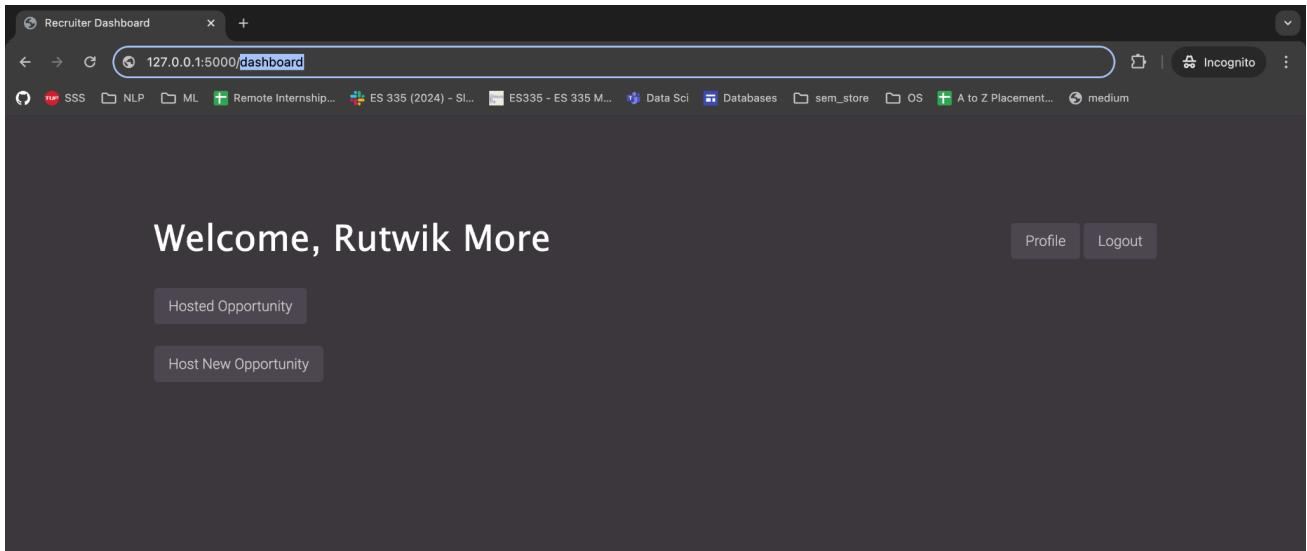
Use the encodeURIComponent() method to encode the input data before sending it to the server to prevent special characters from being interpreted as HTML code.

```
document.addEventListener('DOMContentLoaded', () => {  
    $('#apply-for-opportunity').submit(function (event) {  
        event.preventDefault();  
        var name = $('#name').val();  
        var resumeLink = $('#resumeLink').val();  
  
        var postData = {  
            name: encodeURIComponent(name),  
            resume: encodeURIComponent(resumeLink)  
        };
```

## Other Attacks (for Bonus Marks)

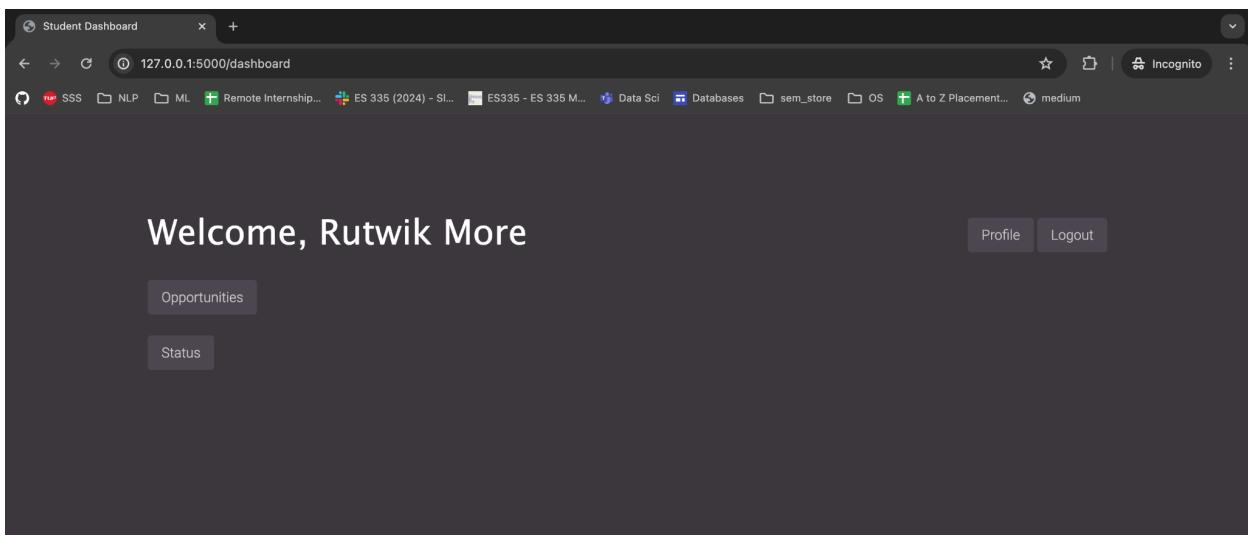
### 3. Insecure direct Object references (IDOR) (URL Attack)

1) The user logs into the application using their credentials. After logging in, the user is redirected to a page that displays their account information or a dashboard of some sort. Such as Student dashboard (which is located at <http://127.0.0.1:5000/dashboard>) which is intended to be accessible only to privileged users and is not displayed in the navigation menu or any other links visible to regular users.

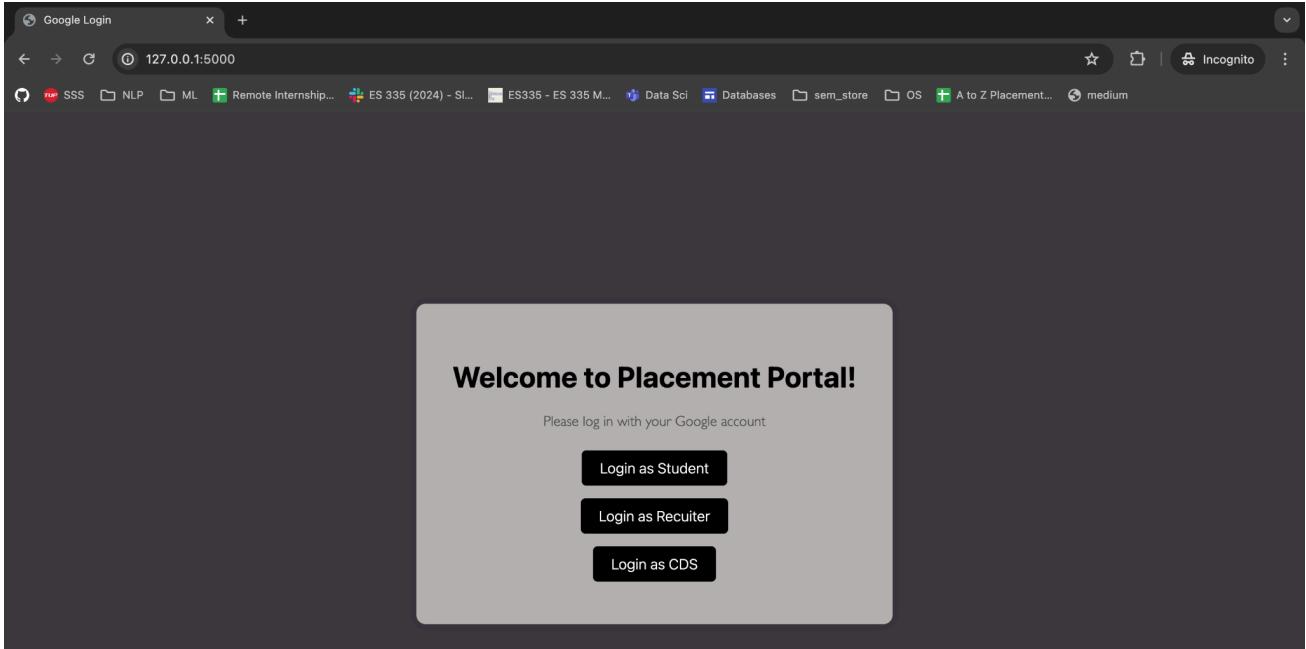


2) The user inspects the HTML source code of the page they are currently on and finds a link or reference to the admin page URL. Such as the Recruiter can see the URL for student dashboard and manually edits the URL in the browser's address bar to replace the current page's URL with the student dashboard page URL, and hits enter.

The server will respond with the students' dashboard page which is displayed to the recruiter even though they are not authorized to access it.



3) After we defend the attack user will get directed to the login page instead of the student page.



There is vulnerability is present in our code. This dashboard does not allow the attacker to access the welcome page if not logged in.

```
@app.route('/dashboard')
def dashboard():
    if 'email' not in session:
        return redirect(url_for('index'))

    email = session.get('email')
    name = session.get('name')
    opportunities = get_opportunities()
    return render_template('students/dashboard.html', email=email, name=name)
```

#### 4. Defense against this attack:

To counter this, the following condition is added to the code in multiple locations where such an attack is possible. One of the instances is shown in the figure. The condition is:

```
if session.get ('logged in', False) == False:
    return redirect ('/')
```

This condition redirects the attacker to the welcome page if not logged in.

Repeating the above steps leads the attacker back to the welcome page and doesn't allow the attacker to bypass the login.

```
@app.route('/dashboard')
def dashboard():
    if 'email' not in session:
        return redirect(url_for('index'))
    if session.get('logged in', False) == False:
        return redirect ['/']

    email = session.get('email')
    name = session.get('name')
    opportunities = get_opportunities()
    return render_template('students/dashboard.html', email=email, name=name)
```

## DOS attack

- A denial-of-service (DoS) attack is a method used to render a machine or network inaccessible by overwhelming it with an excessive amount of traffic. This results in the target system failing to function properly, thereby preventing legitimate users from accessing its services.
- To carry out the attack, we implemented a for loop iterating 1000 times, continuously transmitting packets from a single IP address. Below is the bash script for carrying out the attack

```
#!/bin/bash

# Number of times to execute the curl command
NUM_REQUESTS=10000

# URL to curl
URL="http://127.0.0.1:5000/"

# Loop to execute curl command NUM_REQUESTS times
for ((i=1; i<=$NUM_REQUESTS; i++))
do
    echo "Sending request $i..."
    curl -H "Cache-Control: no-cache" -H "Pragma: no-cache" $URL
done

echo "Requests completed."
```

**Defense against the attack :** To avoid this attack we have used flask\_limiter library

```

from flask import Flask, render_template
from flask_limiter import Limiter
from flask_limiter.util import get_remote_address

app = Flask(__name__)
limiter = Limiter(
    get_remote_address,
    app=app,
    default_limits=["100 per minute"]
)

```

- Using the above bash code when we made 100 requests it gave too many requests

```

~ -- -zsh
<a href="/cds" class="google-login-btn button btn">Login as CDS</a>
<br>
<a href="/placement_manager/" class="google-login-btn button btn">Login as Placement Manager</a>
</div>
</body>

</html>> Sending request 100...
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Google Login</title>
  <link rel="stylesheet" href="/static/style.css">
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;700&display=swap" rel="stylesheet">
</head>

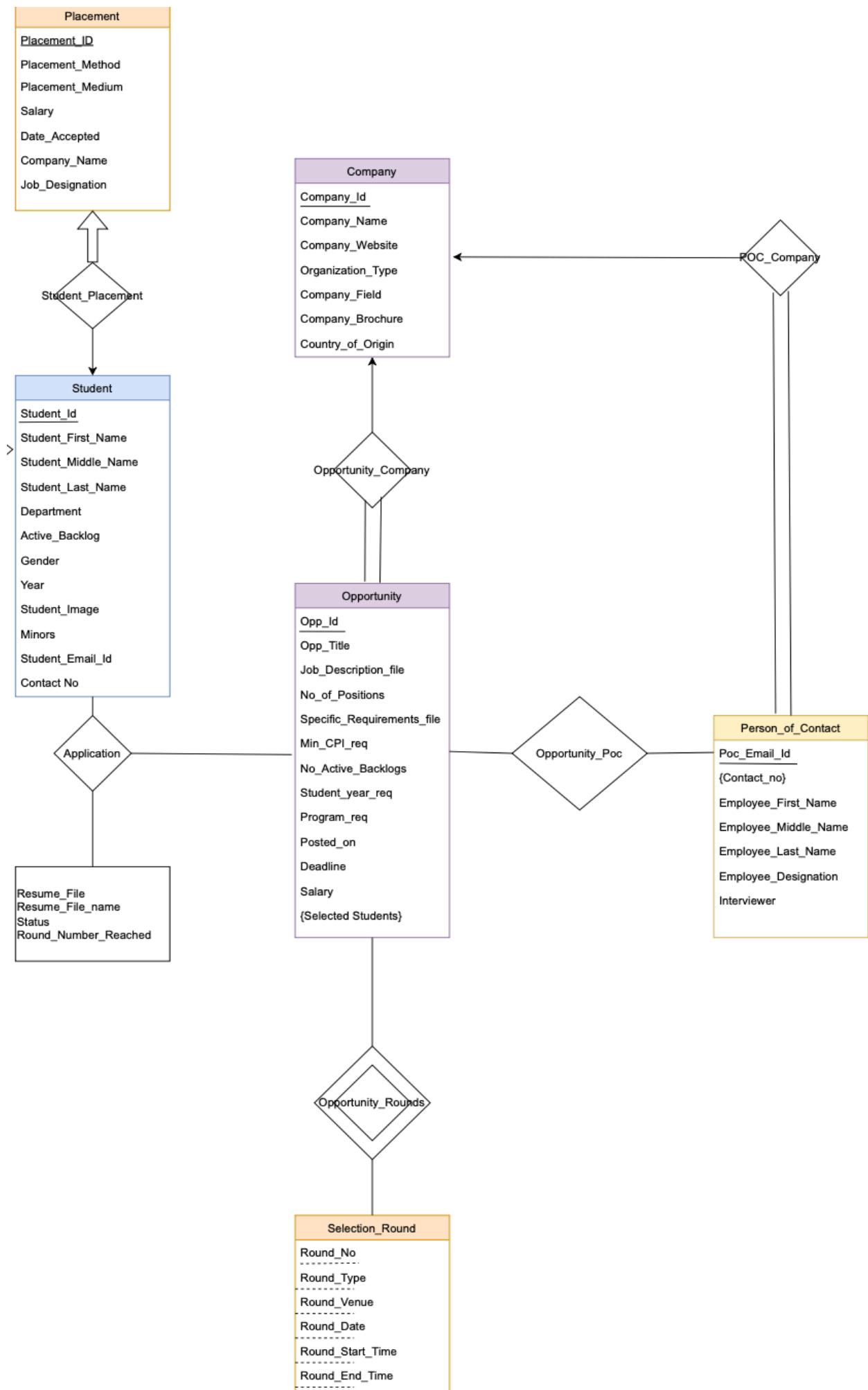
<body>
  <div class="container">
    <h1>Welcome to Placement Portal!</h1>
    <p>Please log in with your Google account</p>
    <a href="/student/" class="google-login-btn button btn">Login as Student</a>
    <br>
    <a href="/recruiter/" class="google-login-btn button btn">Login as Recruiter</a>
    <br>
    <a href="/cds" class="google-login-btn button btn">Login as CDS</a>
    <br>
    <a href="/placement_manager/" class="google-login-btn button btn">Login as Placement Manager</a>
  </div>
</body>

</html>> Sending request 101...
<!doctype html>
<html lang=en>
<title>429 Too Many Requests</title>
<h1>Too Many Requests</h1>
<p>100 per 1 minute</p>
> Sending request 102...
<!doctype html>
<html lang=en>

```

**3.3 Responsibility of G1 & G2:****20 Pts**

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. **(5 points)**



Relational Schema:

Student (Student\_id, Student\_first\_name, Student\_middle\_name, Student\_last\_name, Department, Active\_backlogs, Gender, Year, Student\_image, Minors, Student\_Email\_Id)

Company (Company\_Id, Company\_Name, Company\_Website, Organization\_Type, Company\_field, Company\_Broucher, Country\_of-Origin)

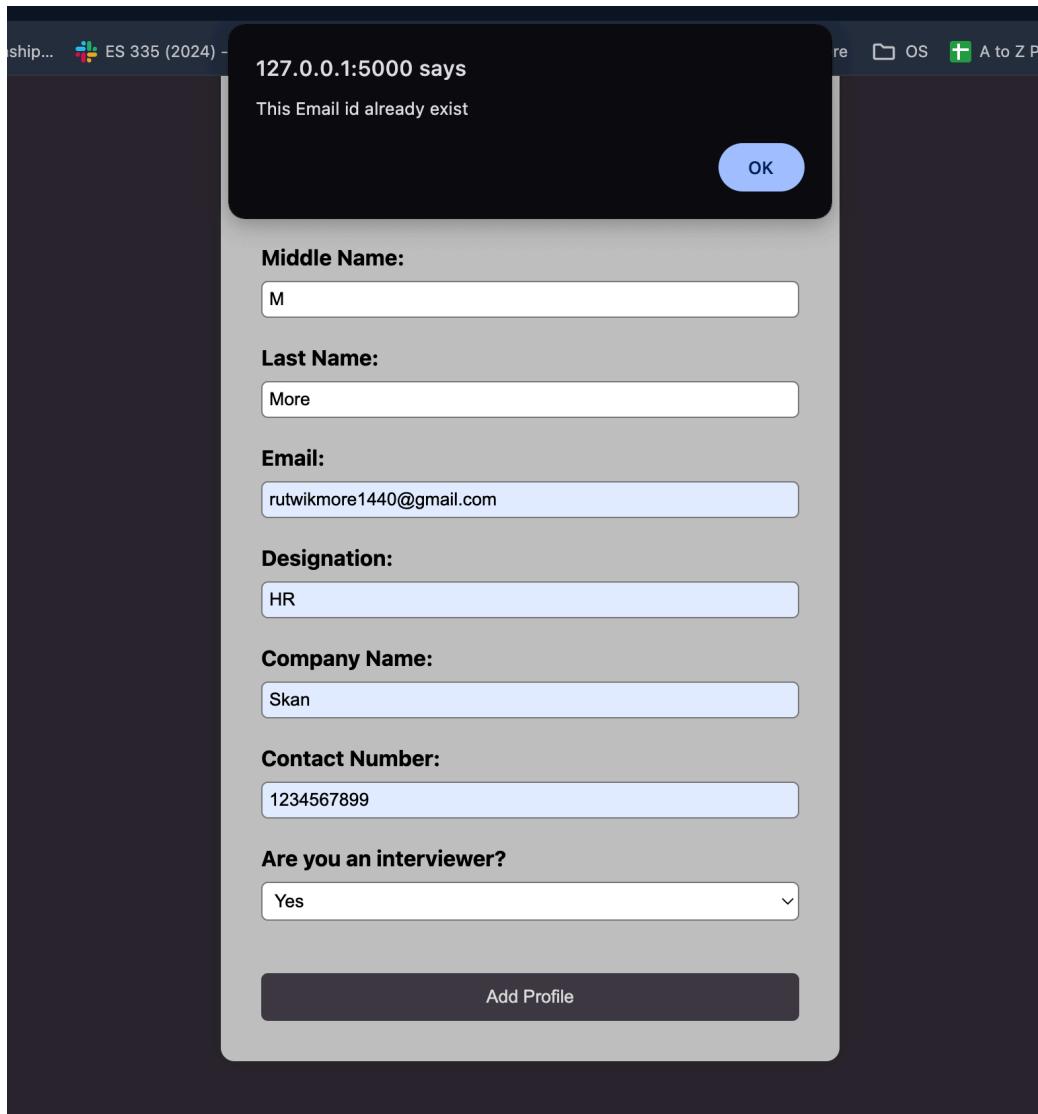
Opportunity ( Opp\_Id, Opp\_Title, Company\_Id, Company\_Name, No\_of\_Positions, Specific\_Requirements\_file, Min\_CPI\_req, No\_Active\_Backlogs, Student\_year\_req, Program\_req, Job\_Description\_file, Posted\_on, Deadline, Salary, {Selected Students})

Person\_of\_Contact (Poc\_Email\_Id, Company\_Id, {Contact\_no}, Employee\_First\_Name, Employee\_Middle\_Name, Employee\_Last\_Name, Employee\_Designation, Interviewer, Company\_Name)

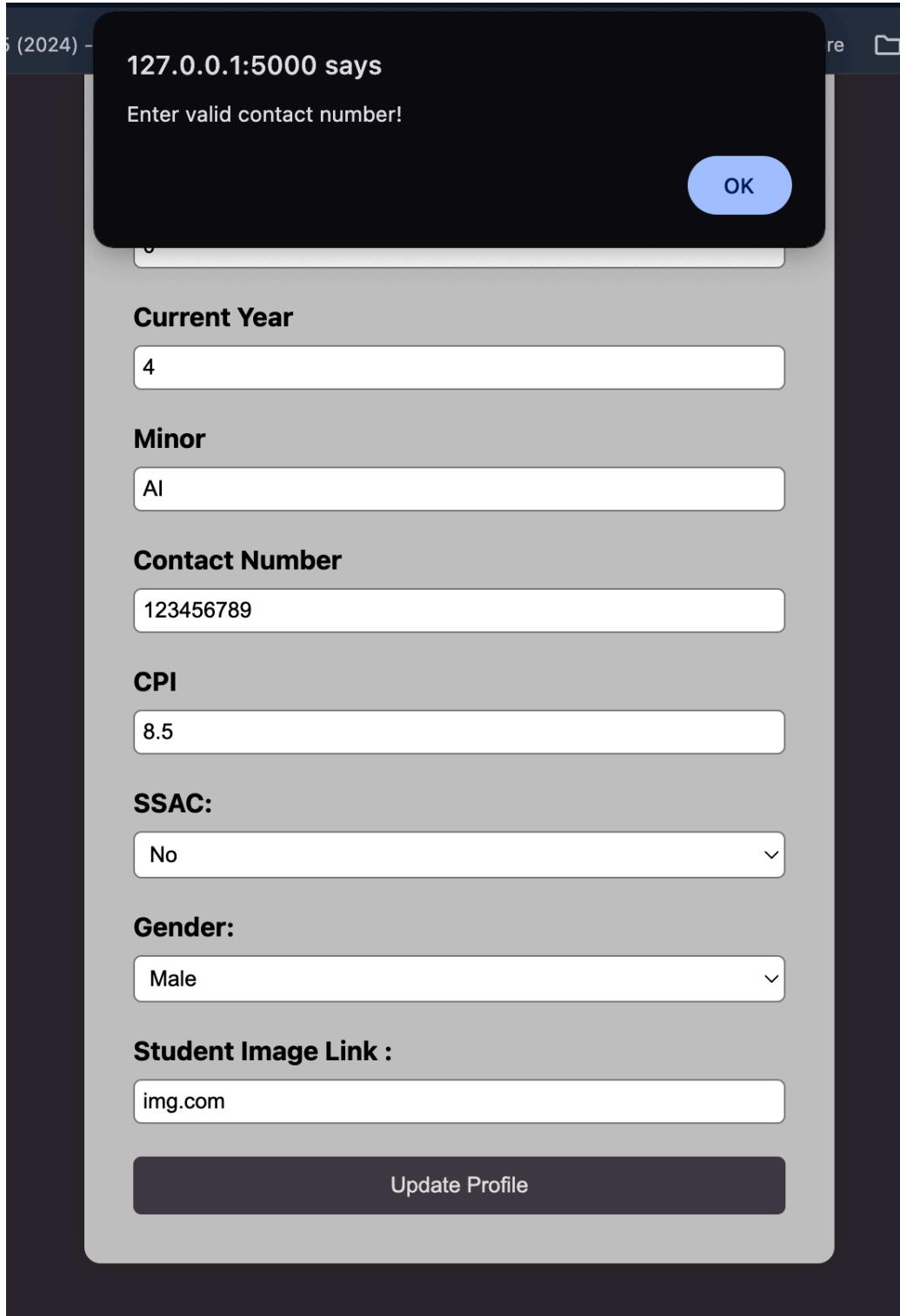
Placement (Placement\_Id, Placement\_Method, Placement\_Medium, Salary, Date\_Accepted, Company\_Name, Job\_Designation)

Selection\_round (Round\_No, Round\_Type, Round\_Venue, Round\_Date, Round\_Start\_Time, Round\_End\_Time)

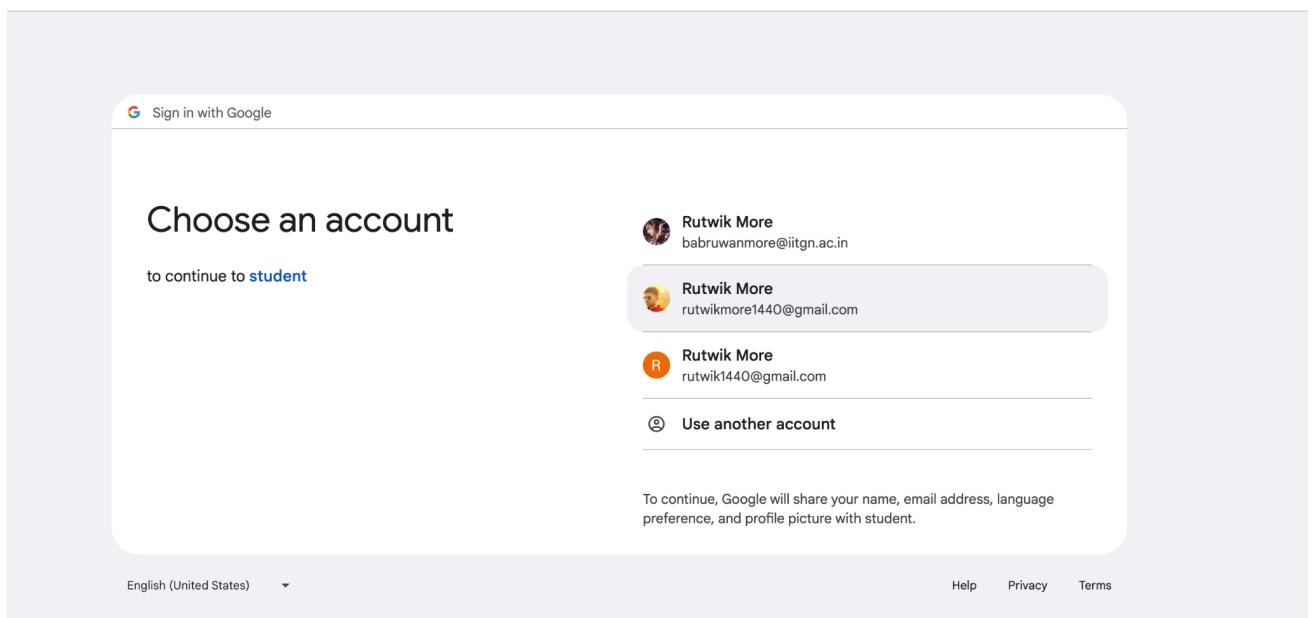
- When the person of contact email already exists in the database it gave the following pop up



- When we entered invalid phone number i.e not of 10 digits it gave the below pop up



- Students cannot login without an IITGN email. If someone tries to login with a non IITGN email id then it redirects the user to the home page.



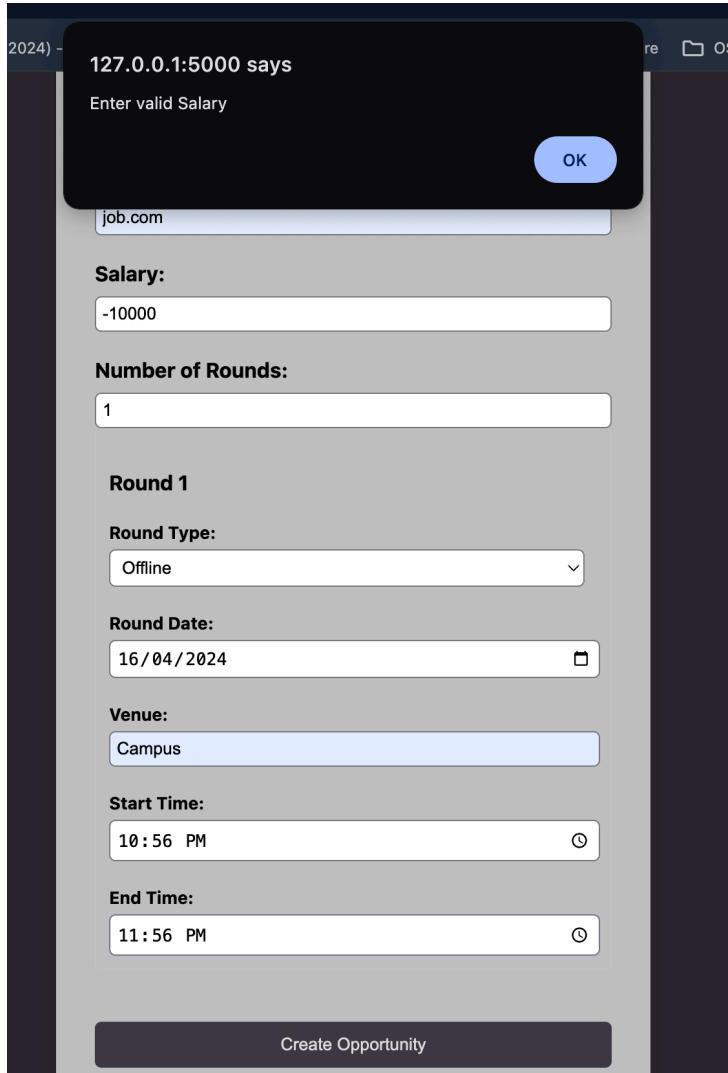
Here Not null constraint is violated As the first name can not be null.

The screenshot shows an "Edit Profile" form with the following fields and their values:

- First Name:** (empty input field)
- Middle Name:** M (highlighted with a red border and a tooltip: "Please fill in this field.")
- Last Name:** More
- Designation:** HR
- Company Name:** Skan
- Contact Number:** 1234567899
- Are you an interviewer?** Yes

At the bottom is a large blue "Update Profile" button.

Here we have set constraint such that salary can't be less than 0.



All constraints can be seen in the database.sql file also.

## **Important Links**

- 1) ER Diagram: Draw.io file:

<https://app.diagrams.net/?client=1#G1yCpFEy0jyzqNLsKFg6Lim8za2PPASyPd%7B%22pageId%22%3A%22n4RG28VcgyLvL5a97sLb%22%7D>

- 2) Code for the Data Generation process: Google Colab notebook:

<https://colab.research.google.com/drive/1thD8FtA0pBjFwbtnWxz85L4WhKX0a4s->

- 3) Generated Dataset for the Database: Google Drive:

[https://drive.google.com/drive/folders/1lr40jKZI\\_w58t7\\_DY7AVztuXKtXEc2A2](https://drive.google.com/drive/folders/1lr40jKZI_w58t7_DY7AVztuXKtXEc2A2)

- 4) SQL Files, and data dump:

[https://drive.google.com/drive/folders/1\\_MqzNjfXfNdrbH-PaVhR7fRd7r1Yt9vL](https://drive.google.com/drive/folders/1_MqzNjfXfNdrbH-PaVhR7fRd7r1Yt9vL)

- 5) Github Repository of project:

<https://github.com/adityadeshmukh369/Placement-System>

ALL THE ADDITIONAL ATTACKS HAVE BEEN DONE BY G2 GROUP ONLY.

## Contributions:

Name	Group	Roll Number	Contribution
Anugu Arun Reddy	G1	21110029	<ul style="list-style-type: none"> <li>Contributed in making CDS hierarchy code in backend.</li> <li>Debugged parts of the authentication code.</li> <li>Contributed to documentation.</li> <li>Interacted with the stakeholders and took the feedback.</li> </ul>
Abhay Kumar Upparwal	G1	21110004	<ul style="list-style-type: none"> <li>Reviewed the initial feedback, assessed alterations, and brainstormed ideas for enhancements.</li> <li>Contributed to the documentation.</li> </ul>
Aaryan Darad	G1	21110001	<ul style="list-style-type: none"> <li>Worked on making the report of the project</li> <li>Interacted with the stakeholders and understood the feedback, reviewed and communicated the suggested changes.</li> <li>Incorporated the feedback and ensured that all required features are implemented.</li> </ul>
Ahaan Giriya	G1	21110015	<ul style="list-style-type: none"> <li>Suggested some improvements</li> <li>Added CSV download functionality in the Webapp after first feedback</li> <li>Formatted the document</li> </ul>
Gaurav Joshi	G2	21110065	<ul style="list-style-type: none"> <li>Showed 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</li> <li>Documented the changes in the database as per the feedback received from stakeholders.</li> <li>Working on building XSS attack</li> </ul>

			and IDOR (URL) attack. Also built defenses against these attacks.
Adit Kaushik	G2	21110010	<ul style="list-style-type: none"> <li>Contributed for get requests and SQL commands for company POC and CDS Employee required for different requests</li> <li>Wrote the setup requirements and steps to run the Web App.</li> <li>Contributed to the DOS attack and XSS and the solution for it</li> </ul>
More Rutwik	G2	21110133	<ul style="list-style-type: none"> <li>Working on building SQL injection and IDOR (URL) attack. Also built defenses against these attacks.</li> <li>Added Mail functionality in the Webapp after first feedback</li> <li>Added CSV download functionality in the Webapp after first feedback</li> </ul>
Aditya Deshmukh	G2	21110014	<ul style="list-style-type: none"> <li>Contributed to the DOS attack and the solution for it</li> <li>Contributed in making placement manager backend</li> <li>Added search functionality</li> </ul>