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

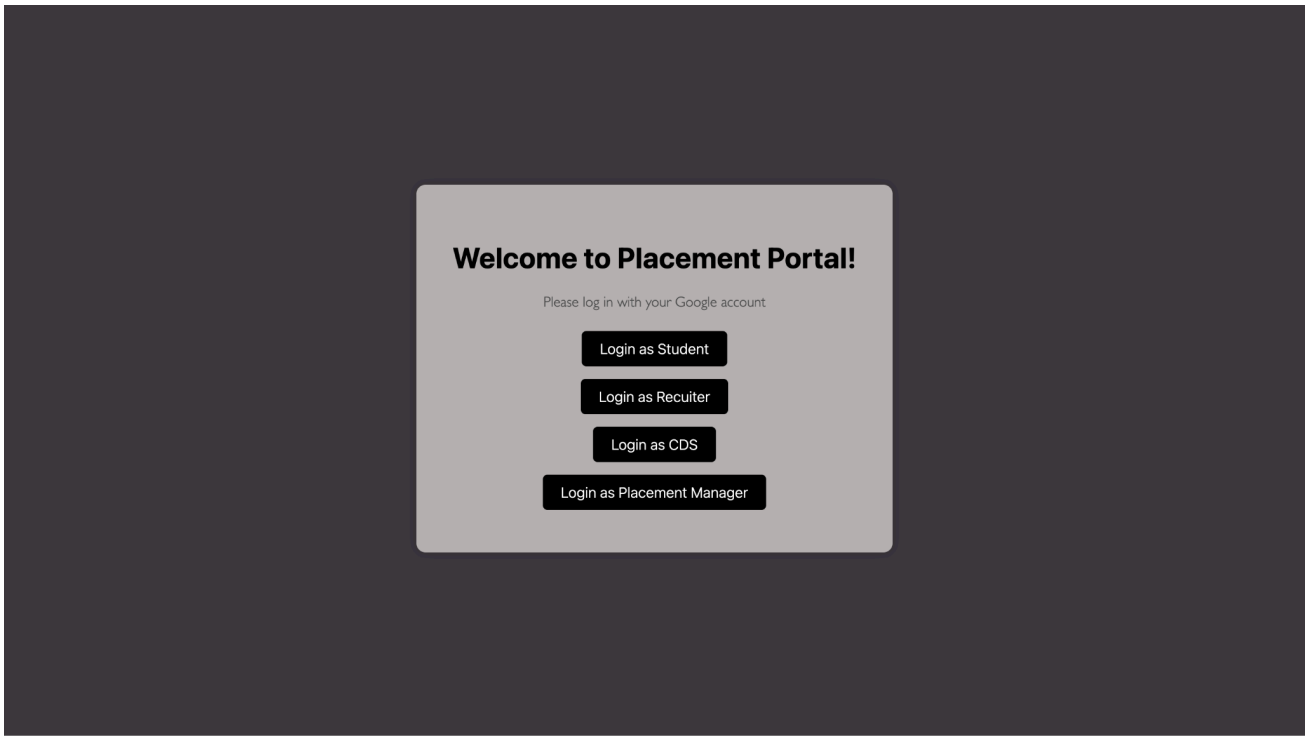
3.1 Responsibility of G1: **40 Pts.**

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)**

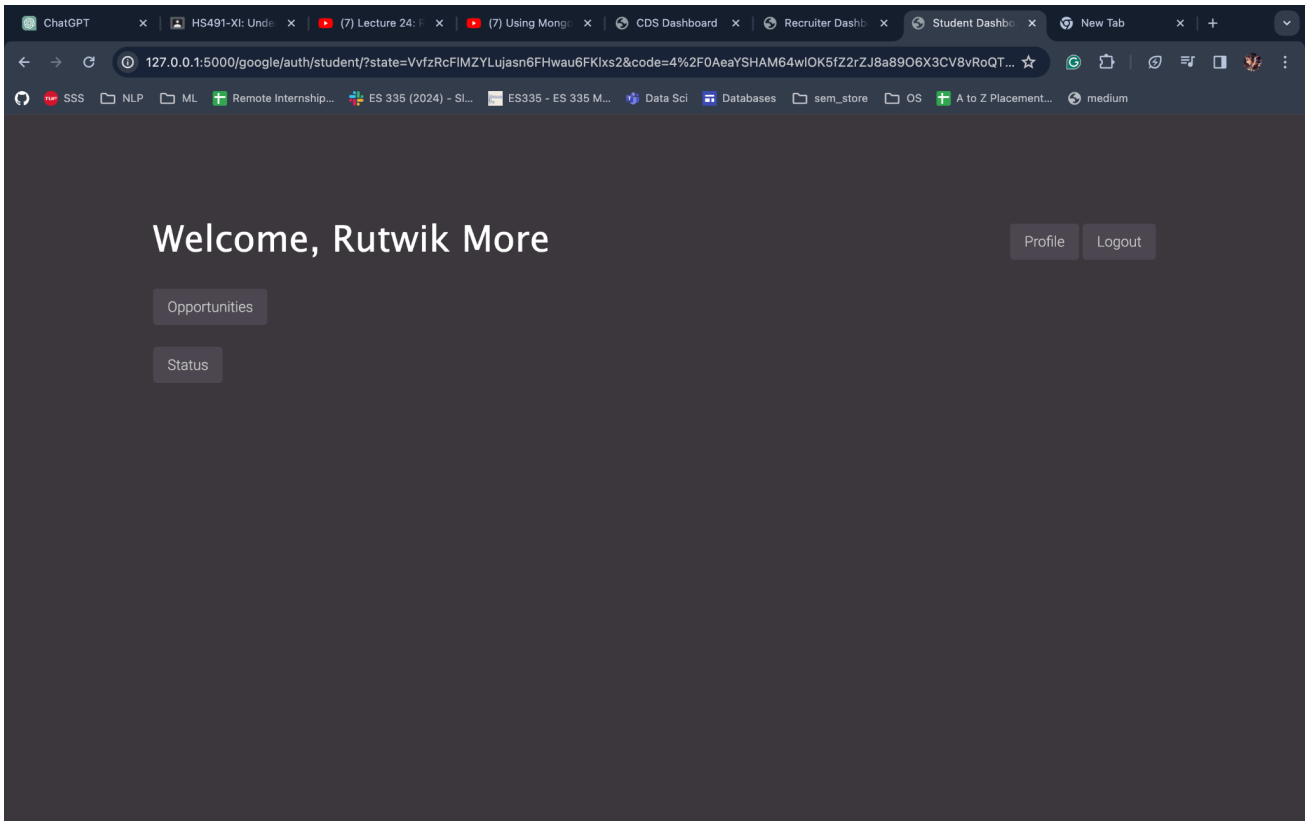
1. Student

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

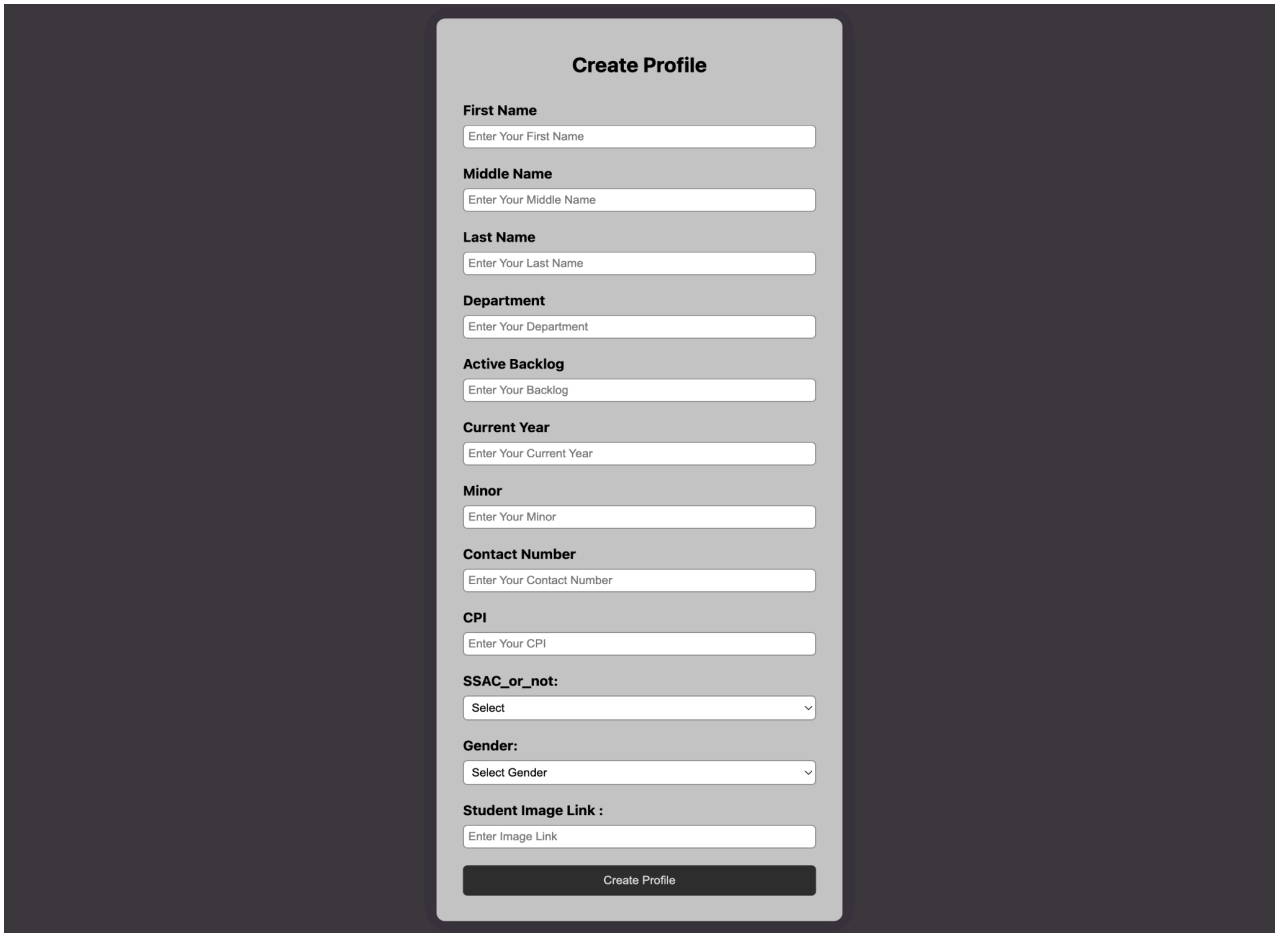
- Login Page



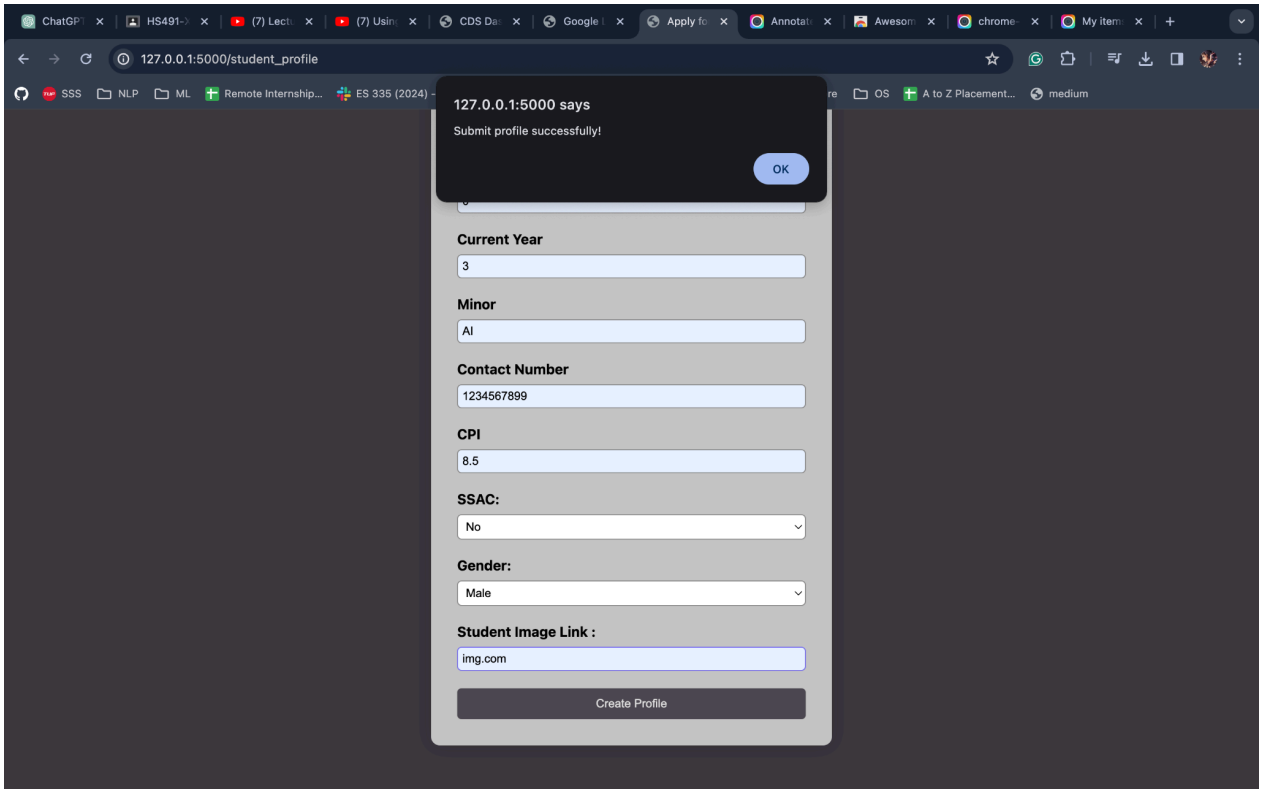
- Student Dashboard



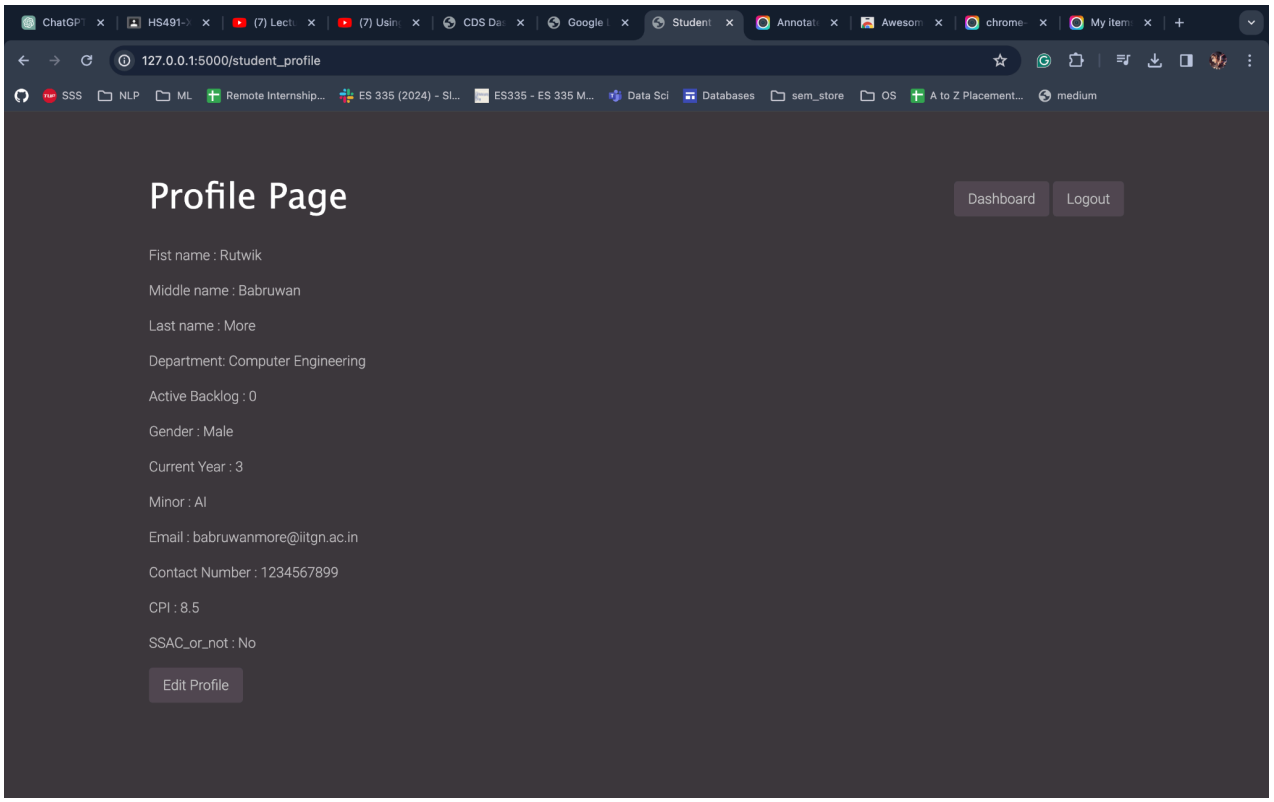
- Student Profile



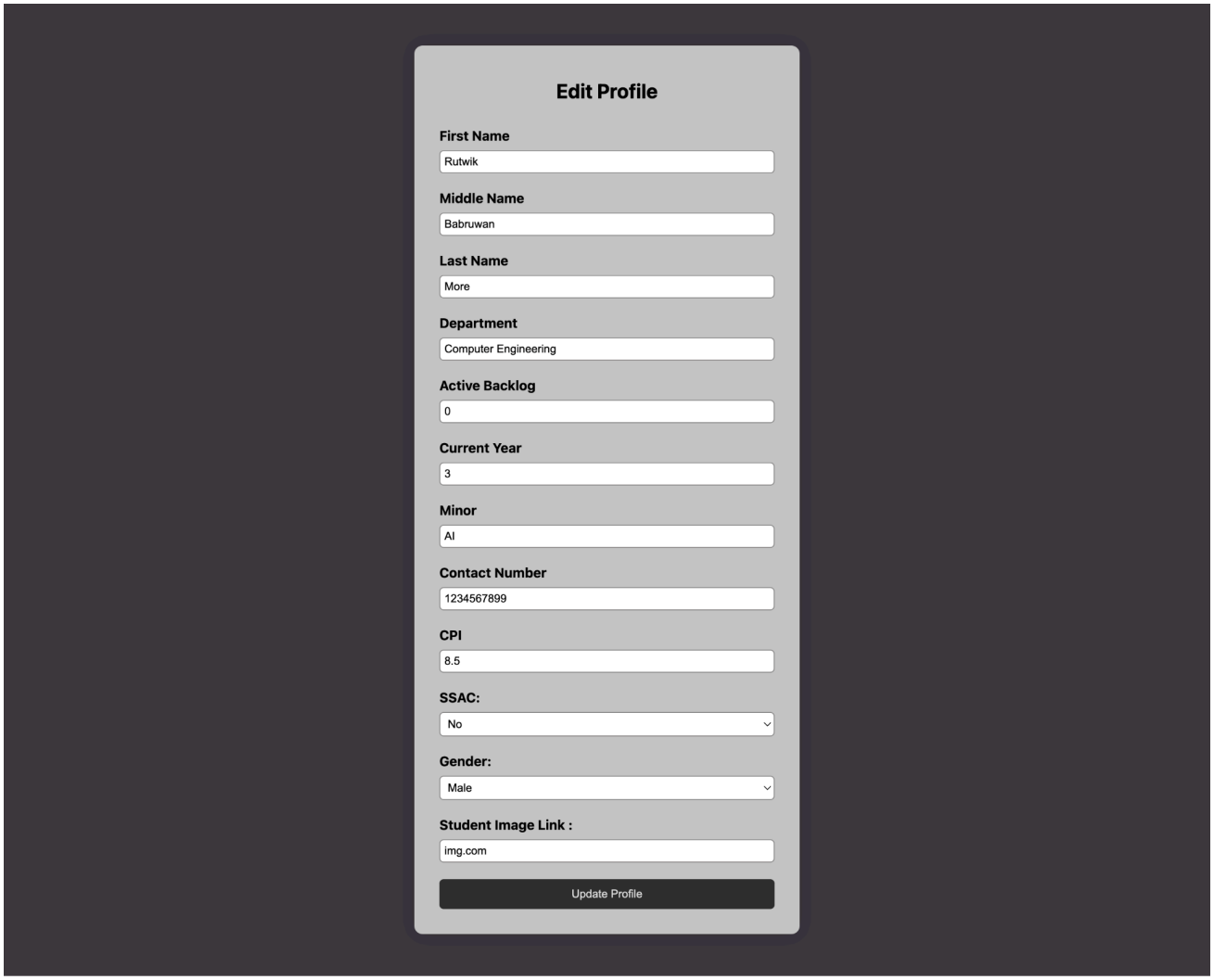
- Pop Up on Creating Profile successfully



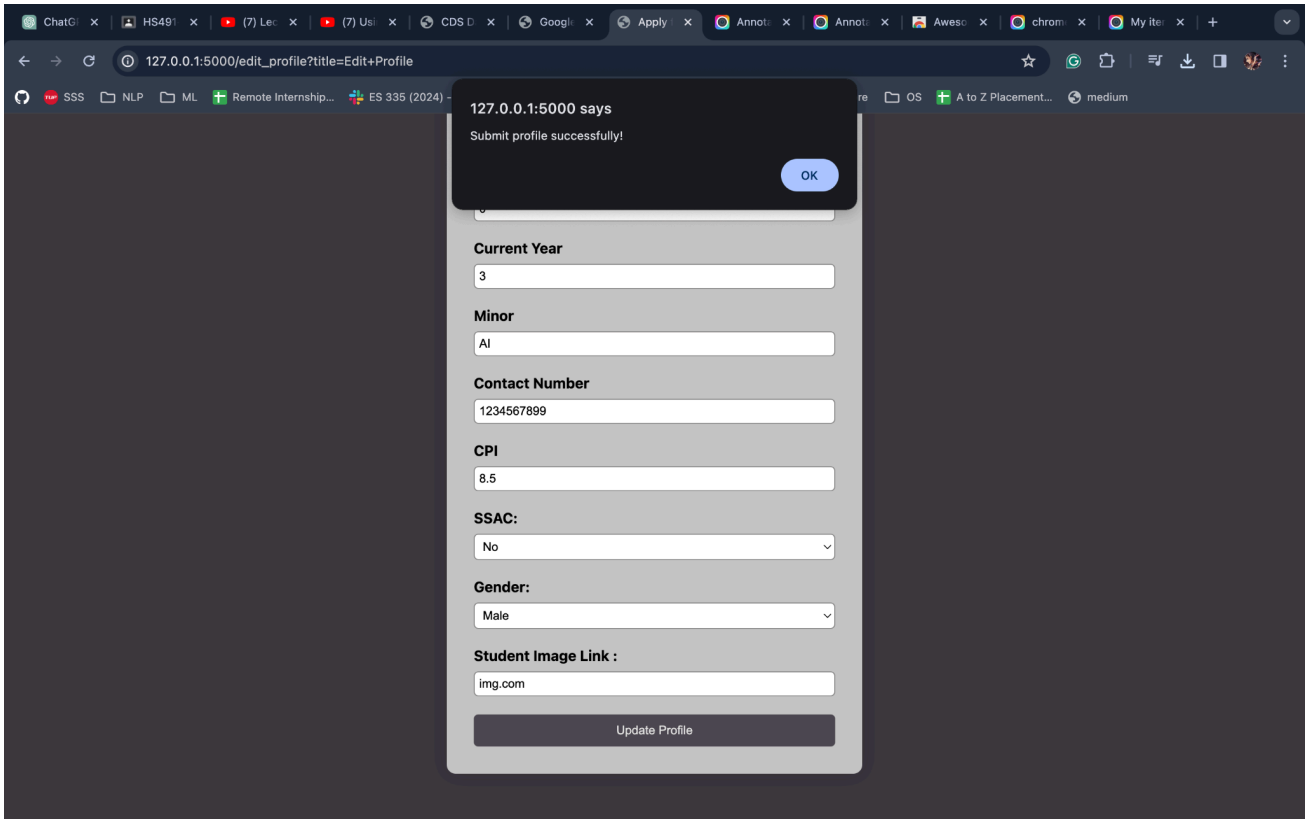
- Profile Page of Student



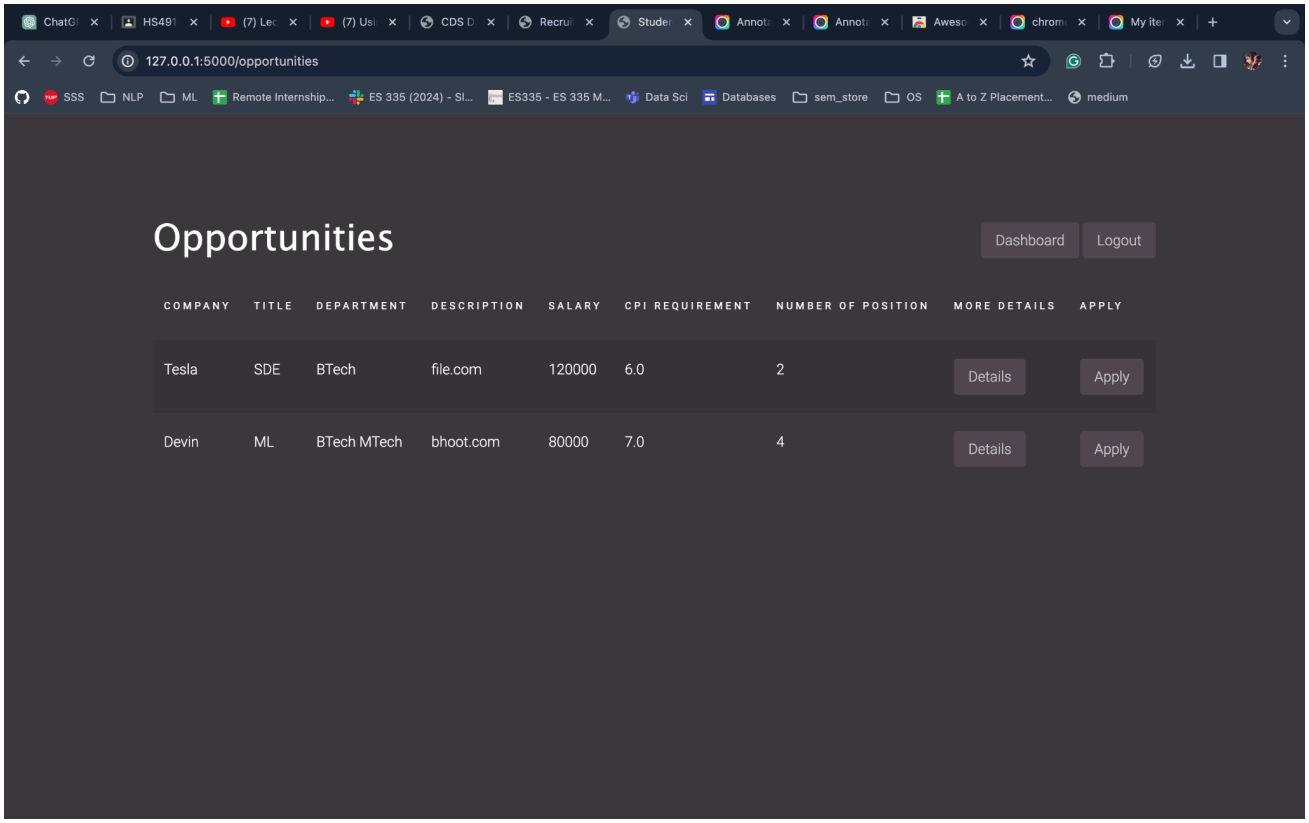
- Student can also edit their profile



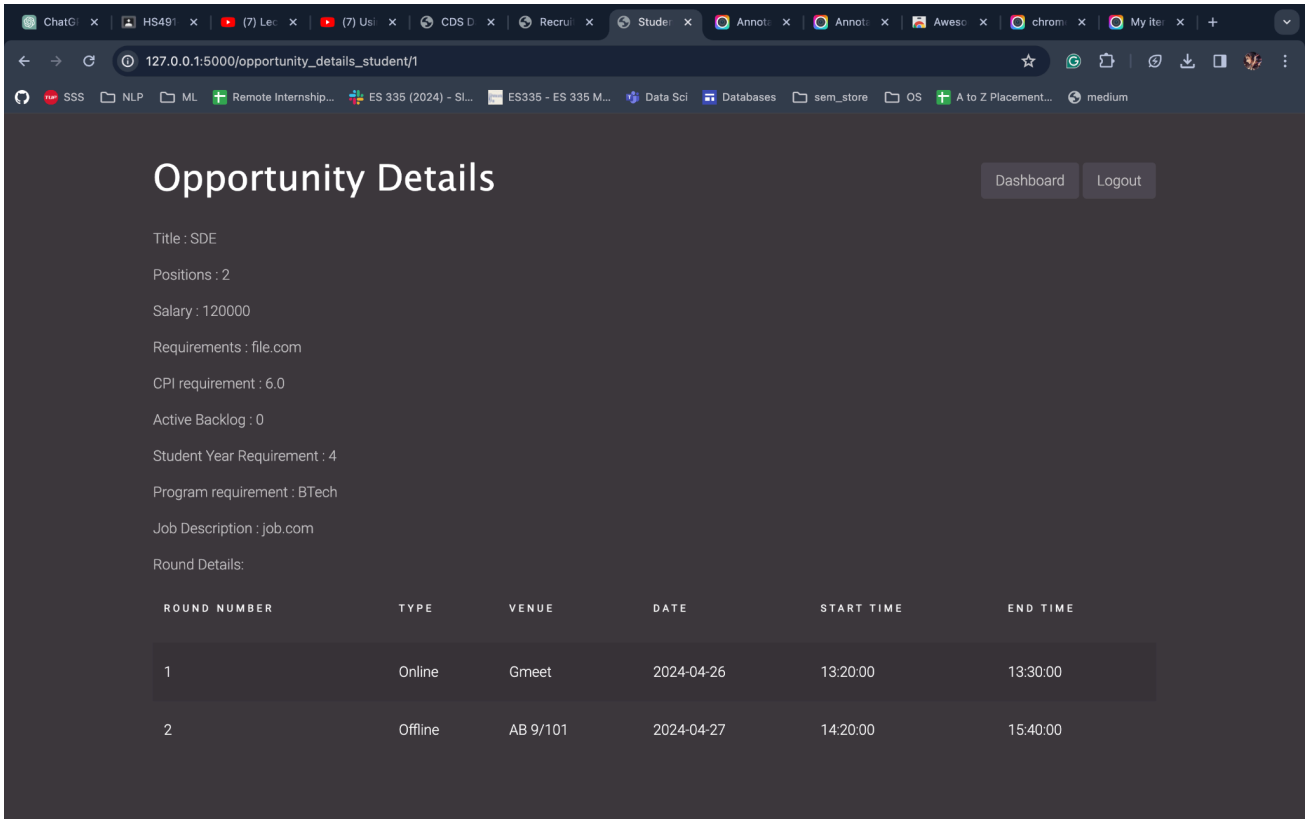
- Pop Up after editing the profile successfully.



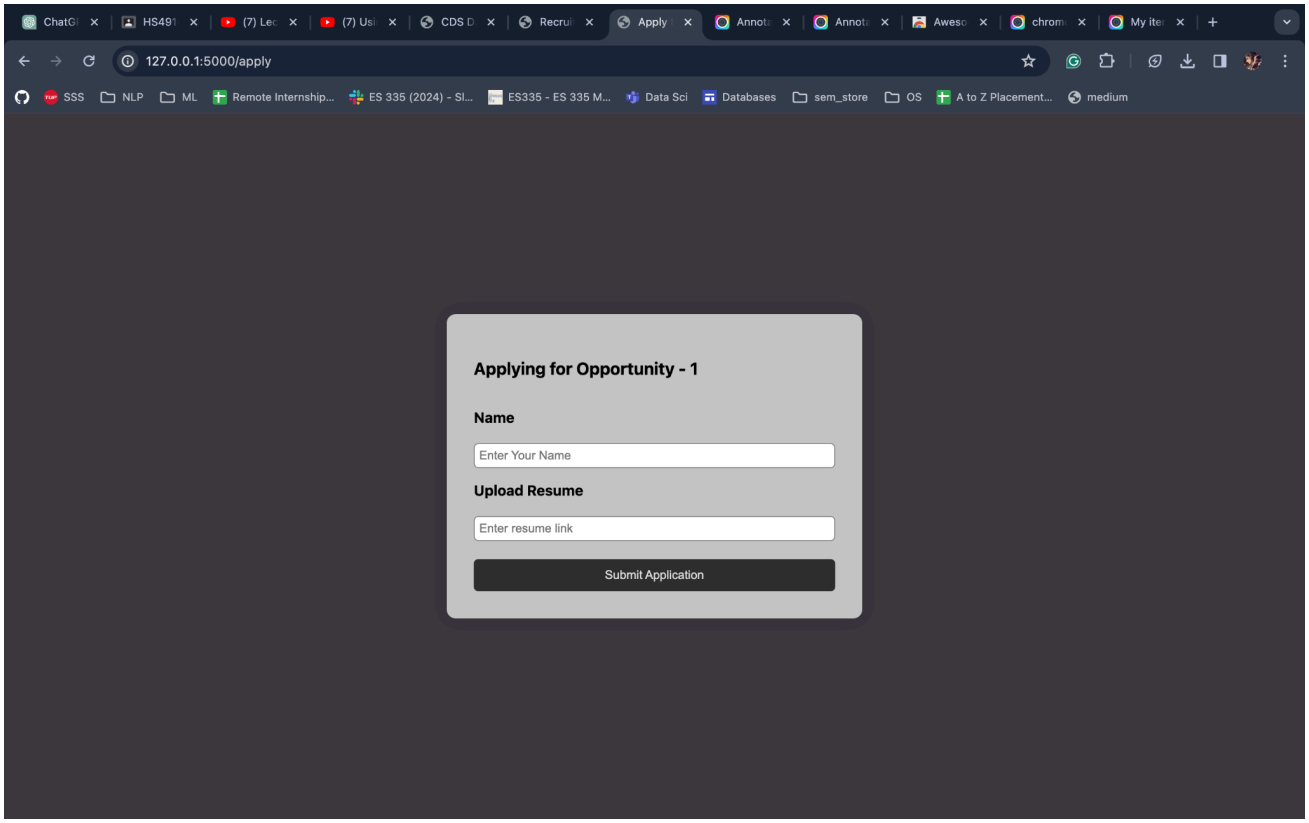
- Students can see the listed opportunities.



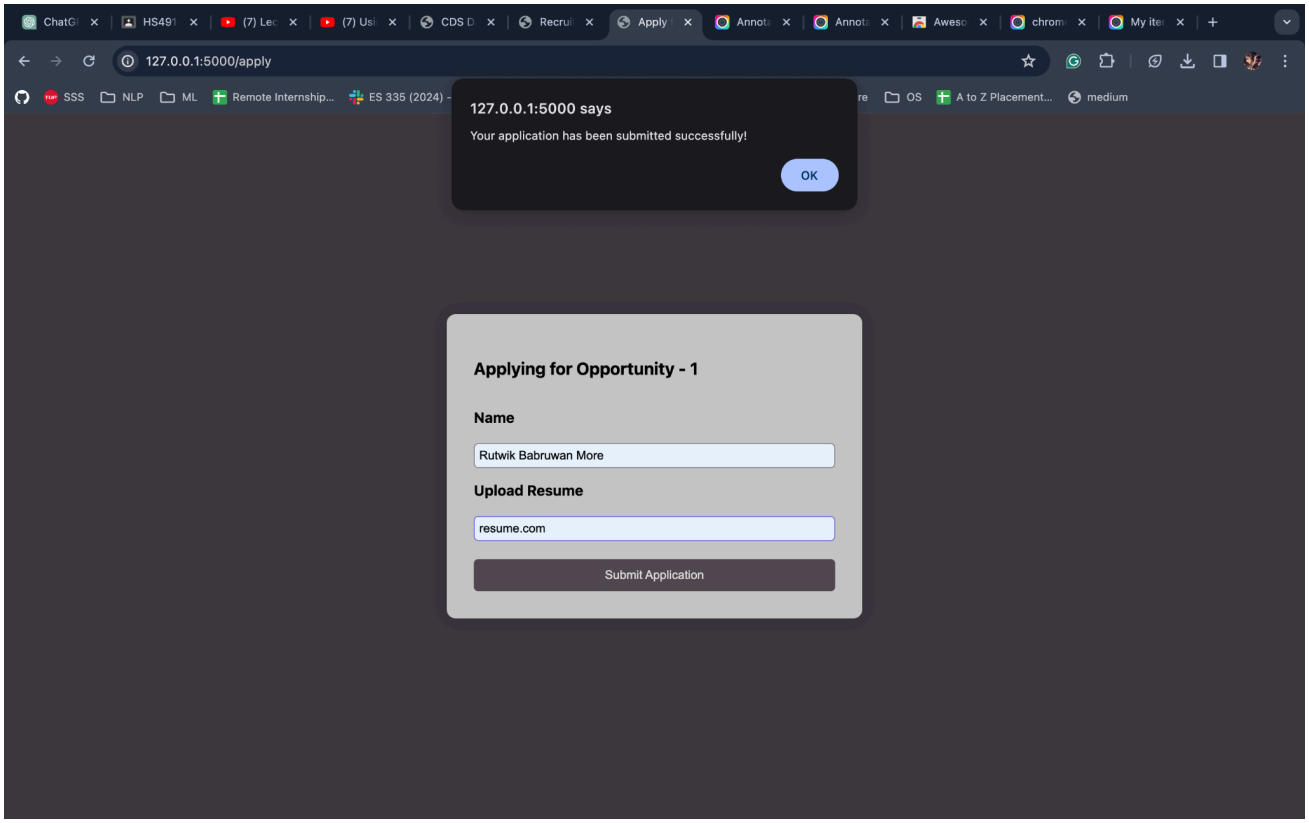
- Student can see each opportunity in detail after clicking Details button



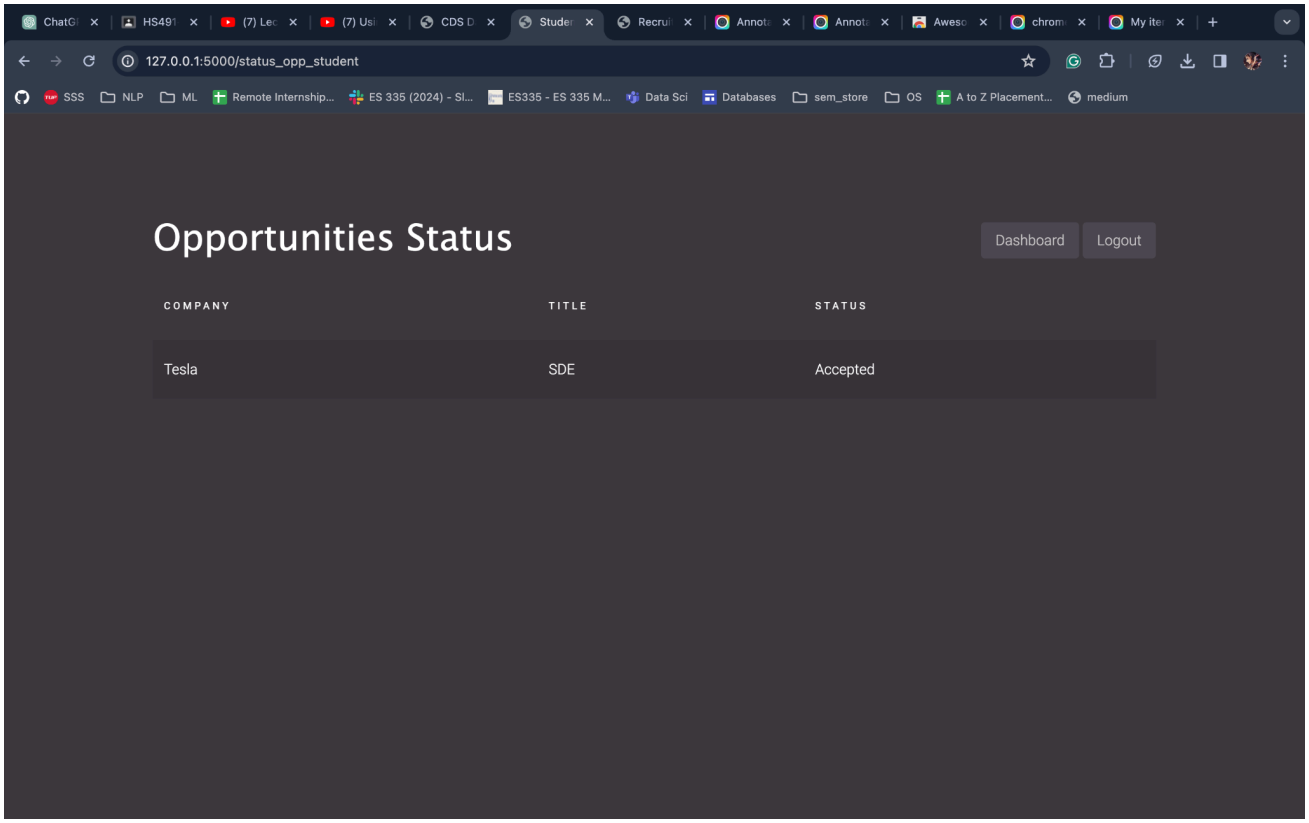
- Student can also apply for the opportunity by clicking on Apply button



- Student get pop up after successful application



- Students can see the status of applied opportunities.

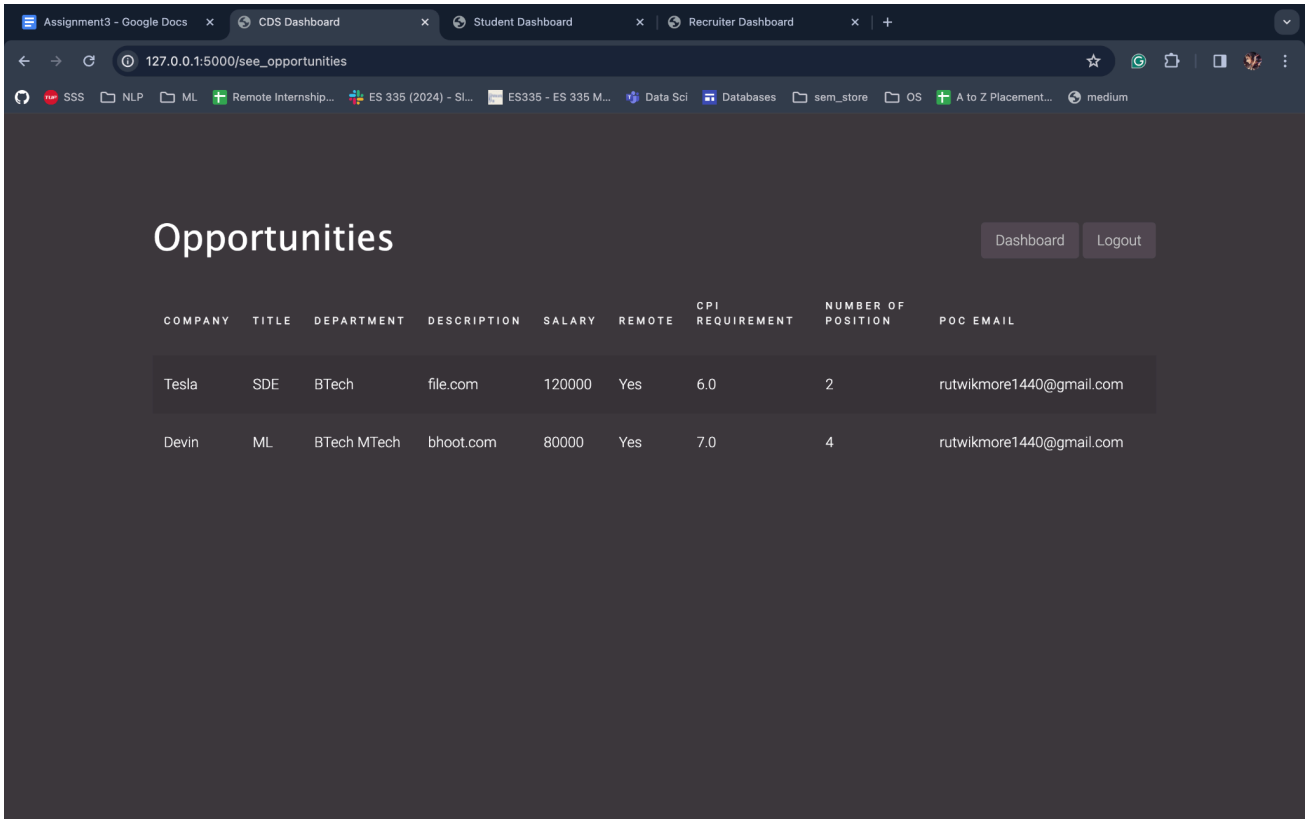


2. CDS Employee:

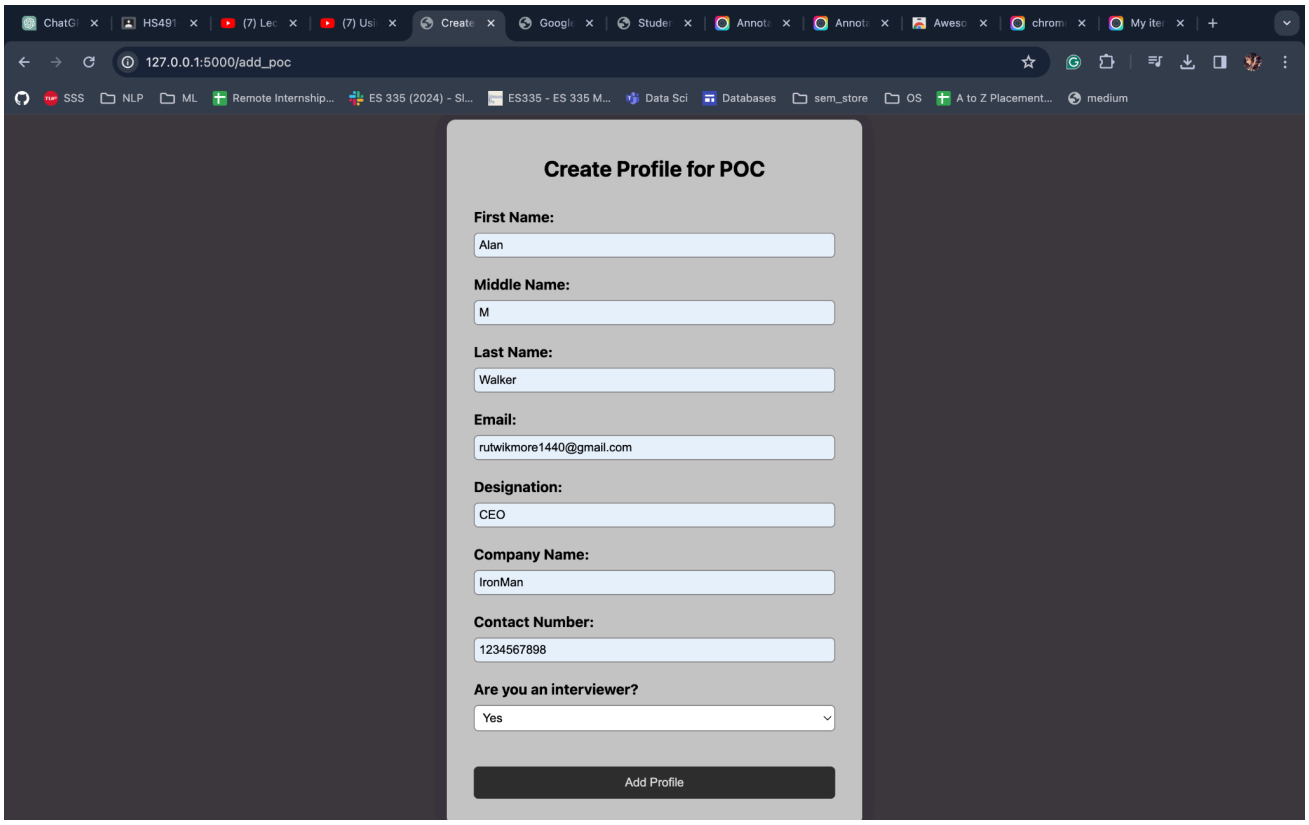
- Privileges
    - CDS have authority to add the person of contact. Only CDS authorised POC have opportunity to host the opportunity
    - CDS also can change the details of the students
    - They have the view access of the opportunity, placement.

- Dashboard of CDS

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

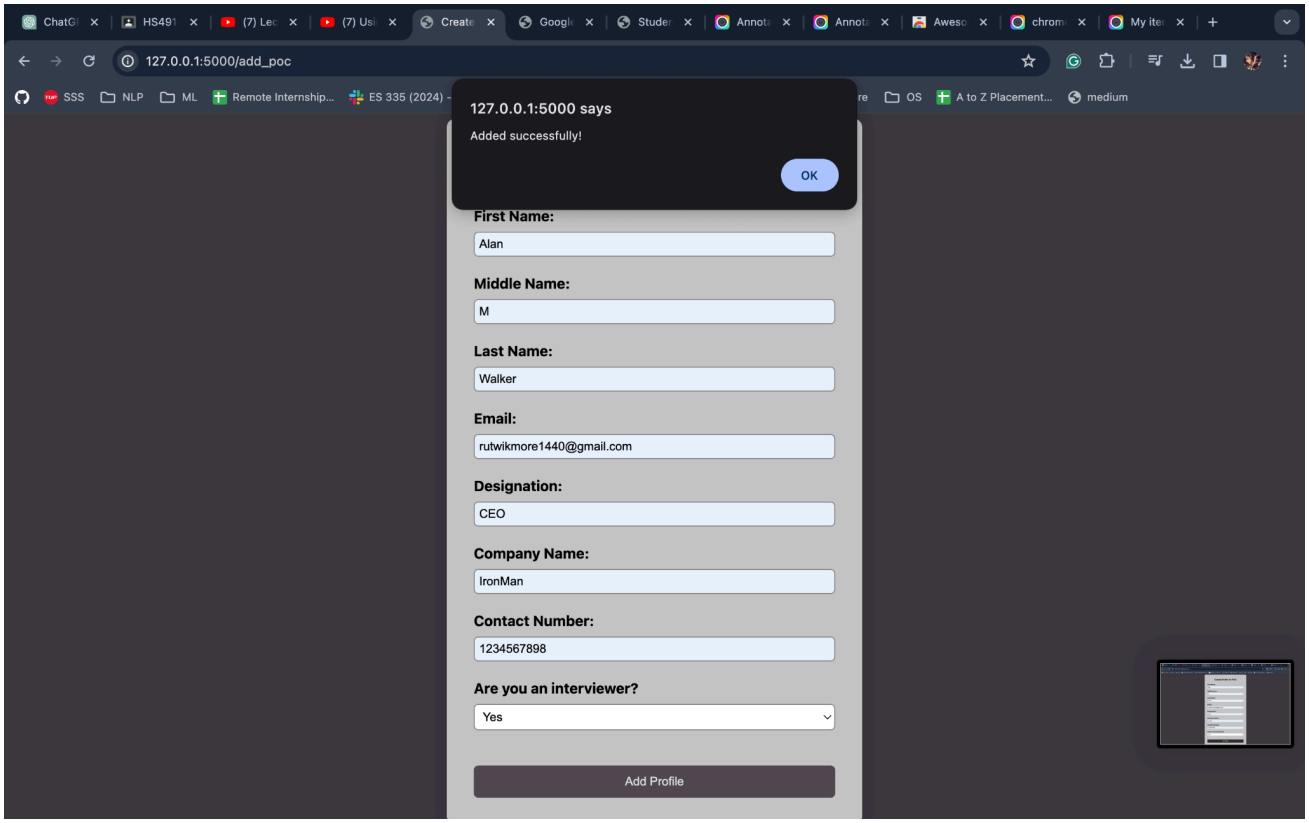


- CDS can add Person of Contact details by clicking on Add POC button.

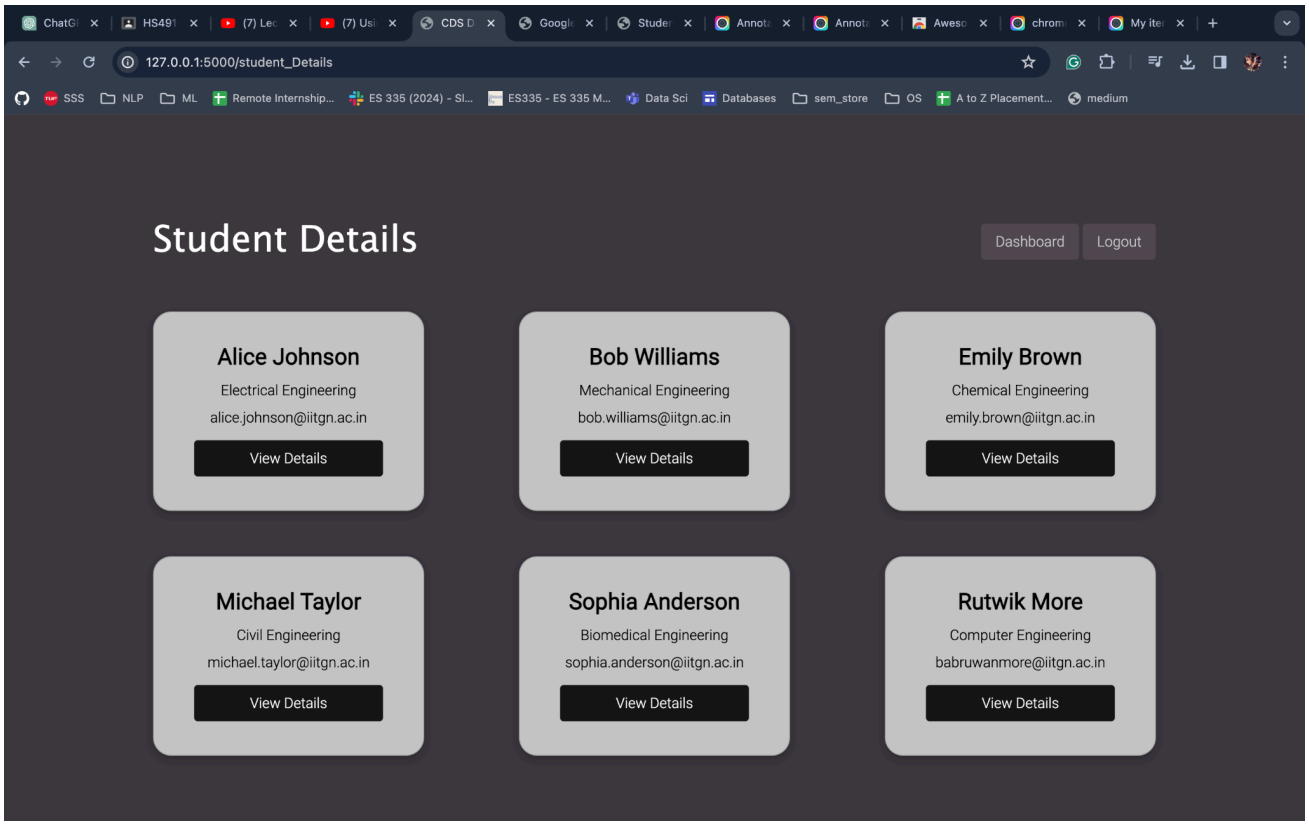


- CDS will get pop up after successfully adding the POC

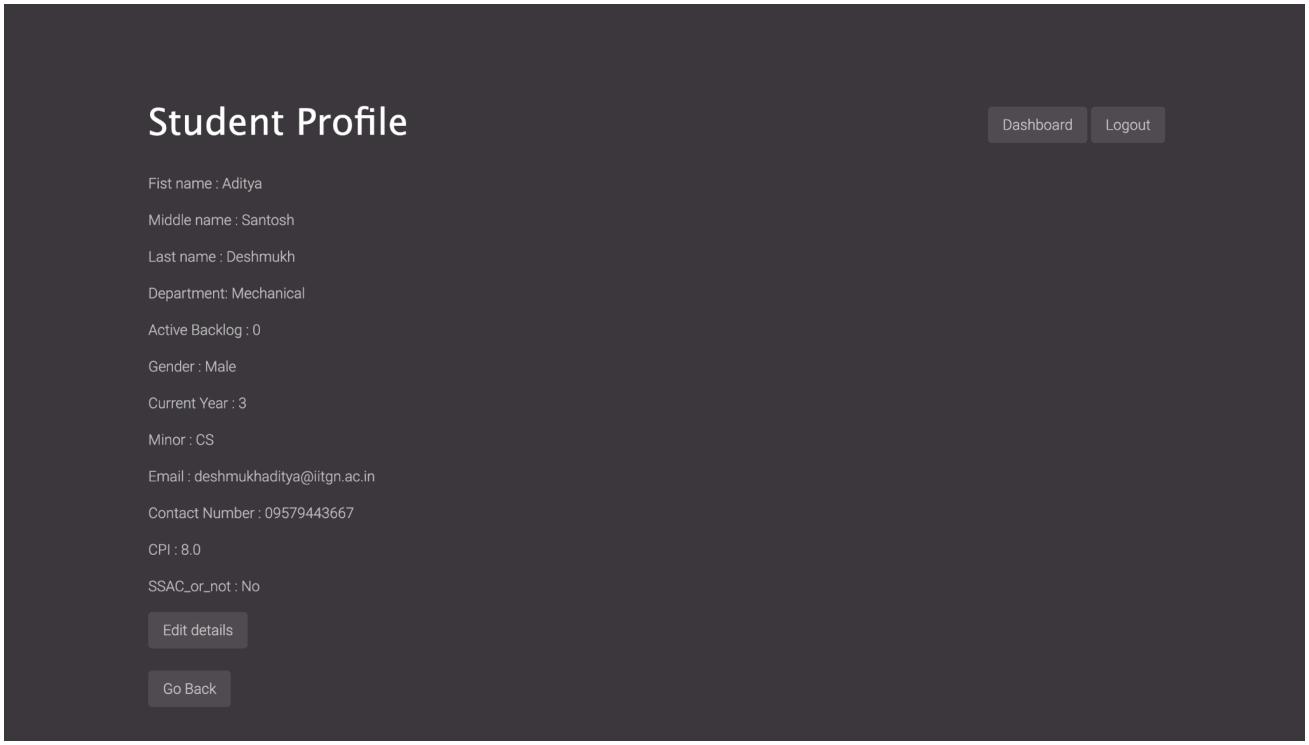
-


127.0.0.1:5000/add_poc
127.0.0.1:5000 says
Added successfully!
OK
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
Add Profile

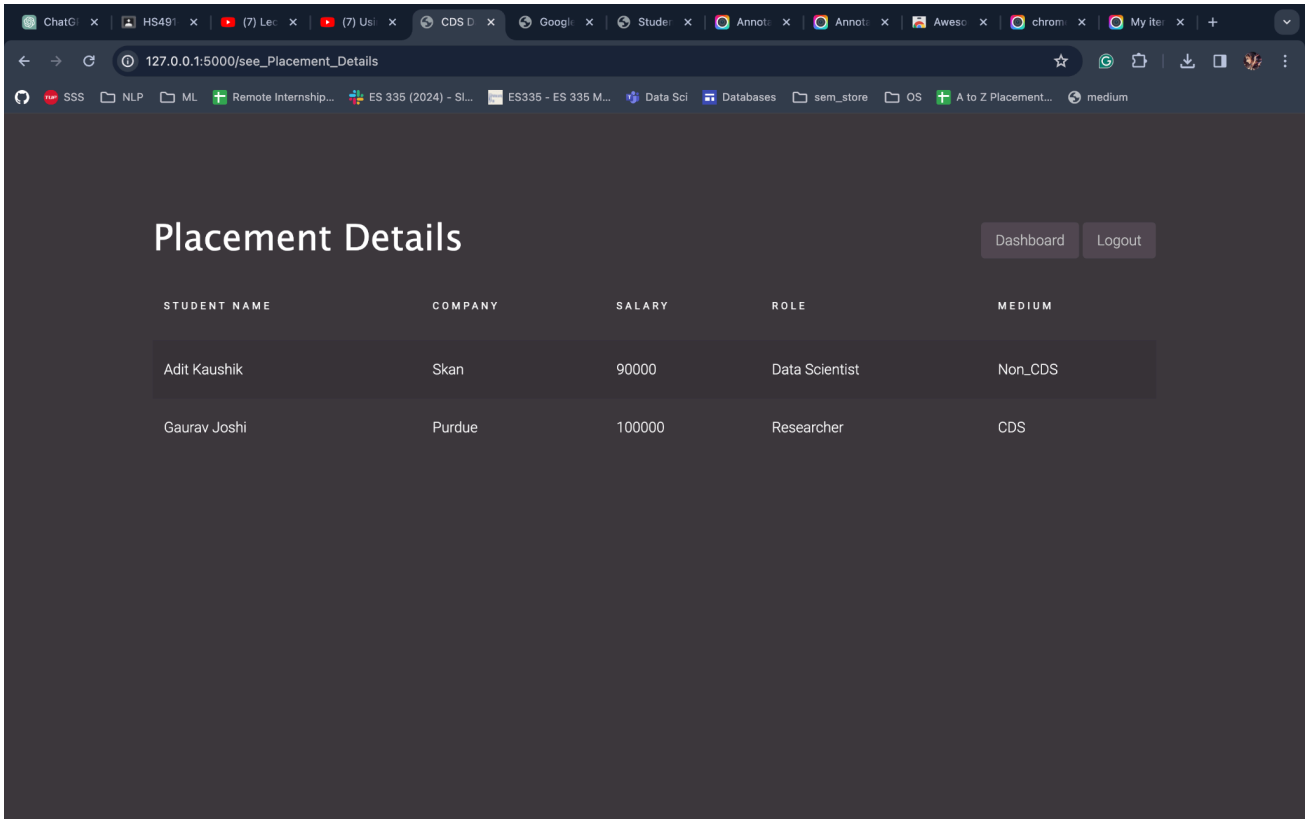
- CDS can see the Details of each student.



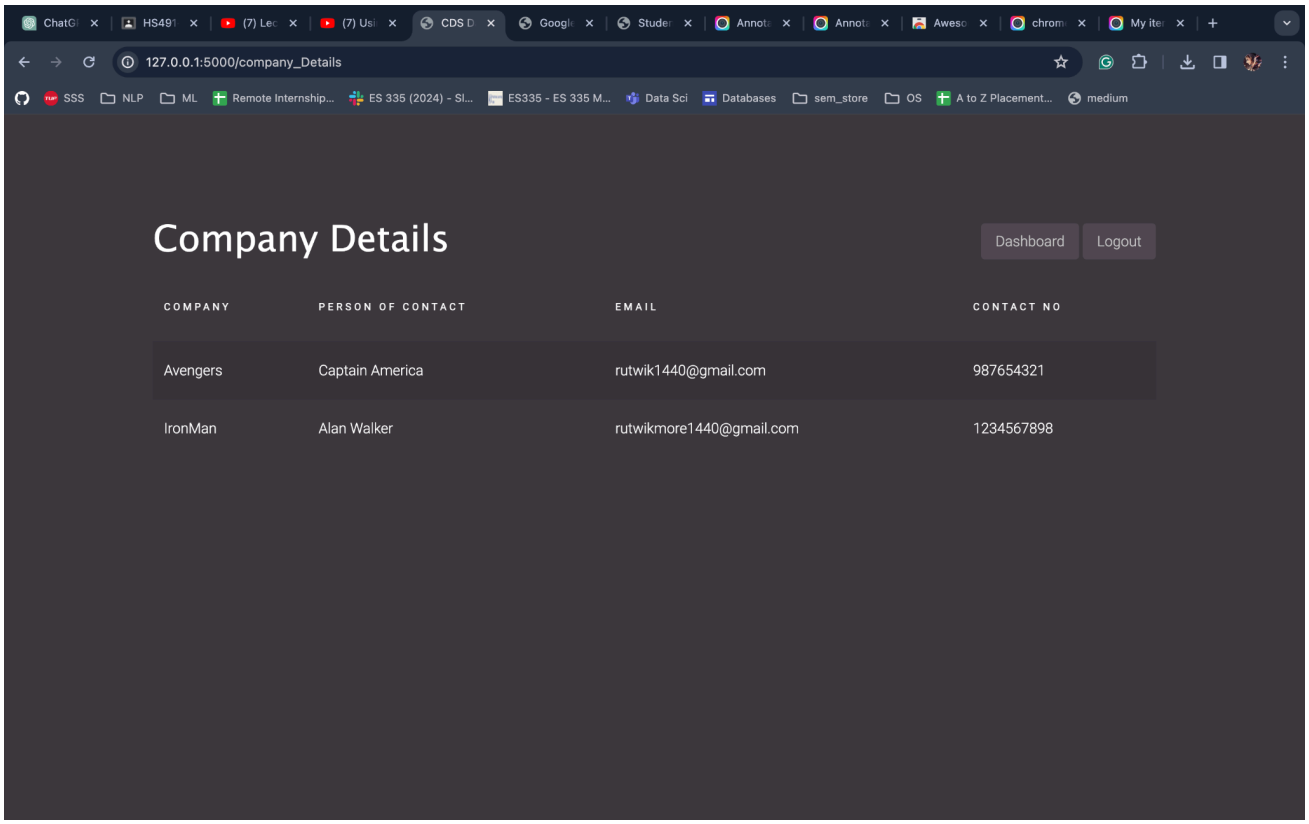
- CDS can See student details in detail by clicking on the button ‘View Details’.



- CDS can see placement details of each student.



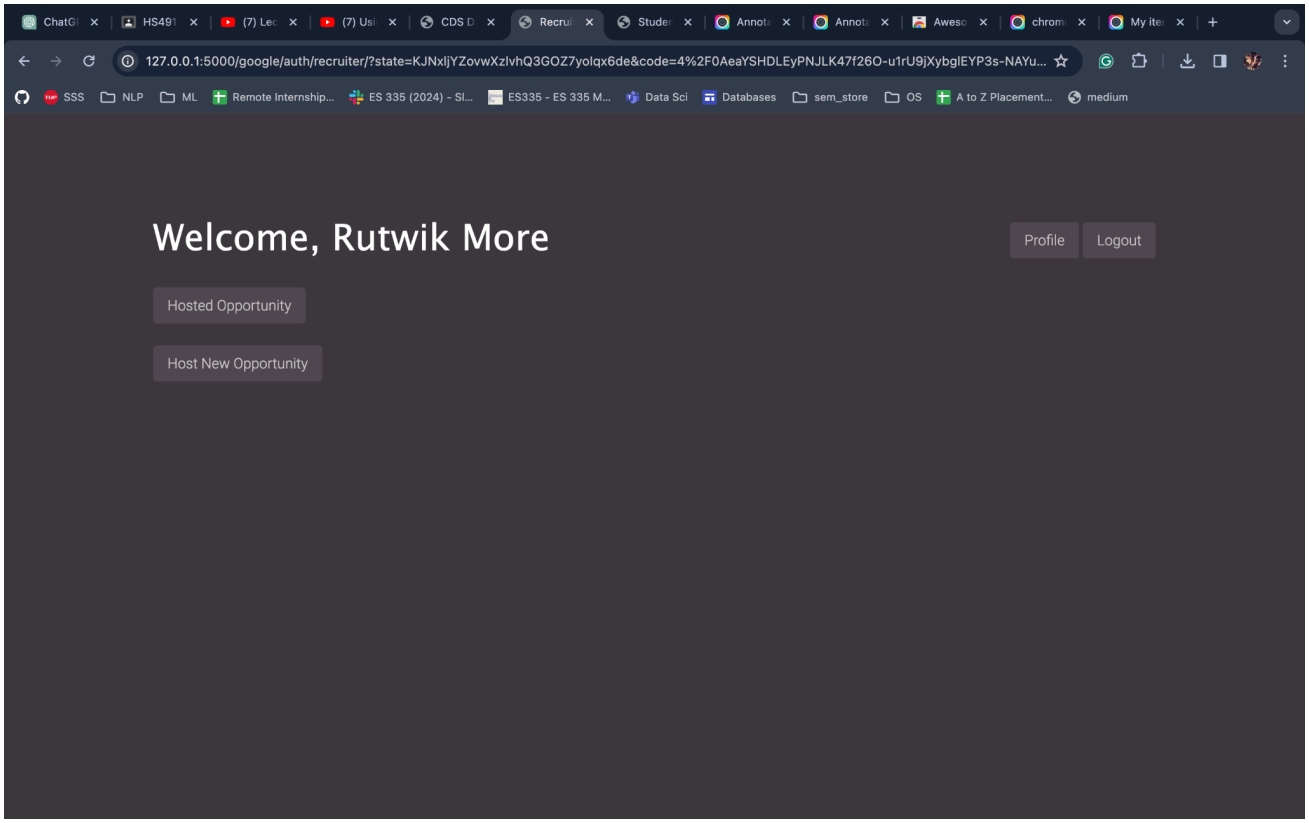
- CDS can see details of all companies.



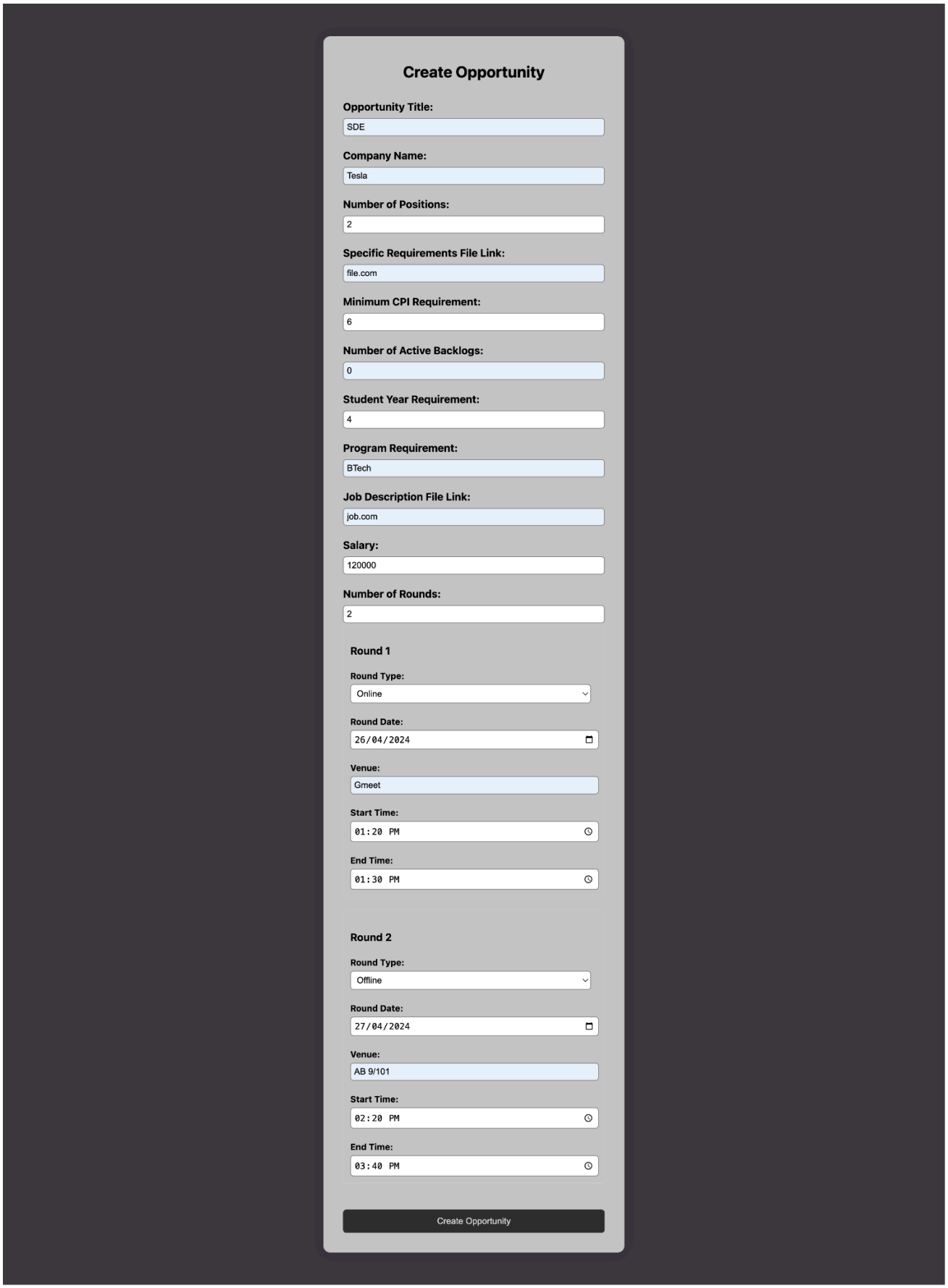
1. **Person of Contact (POC):**

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

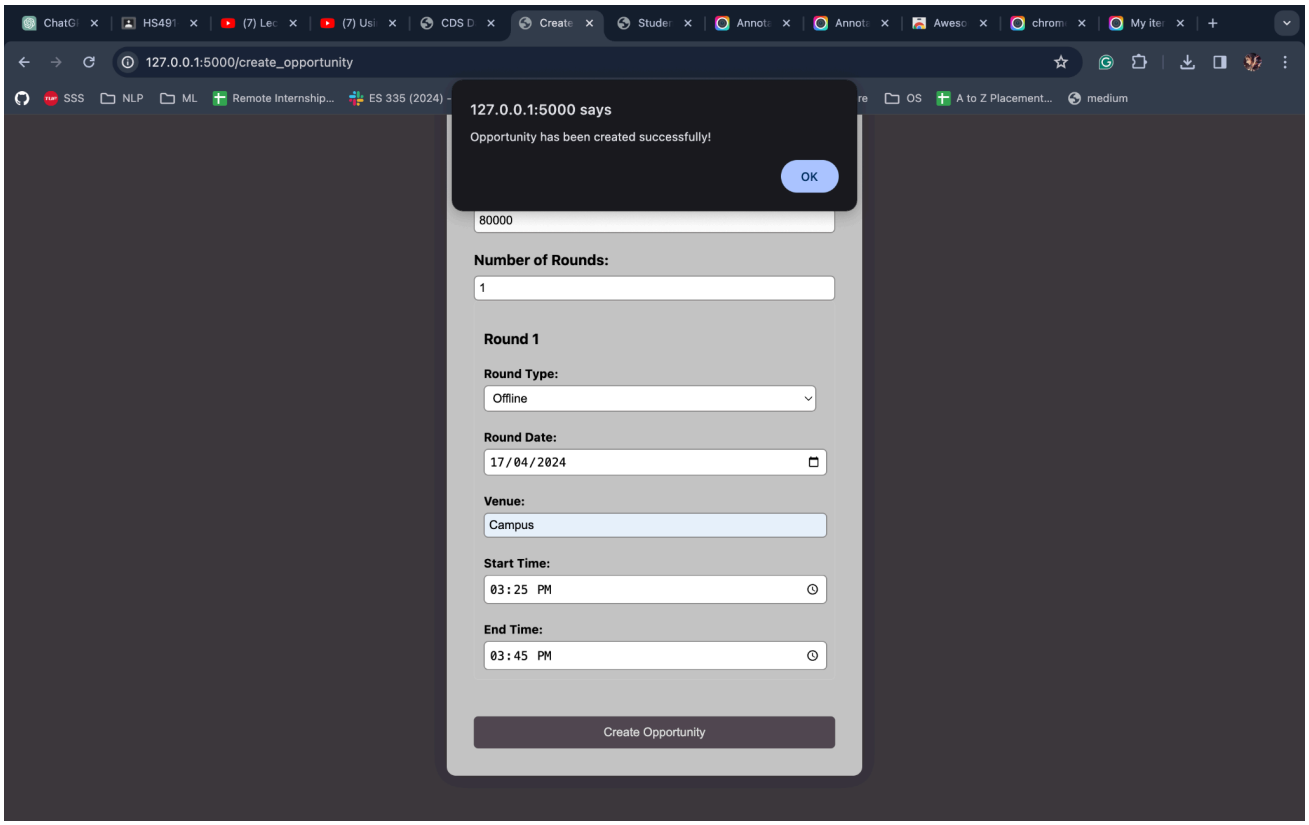
- Dashboard of Recruiter



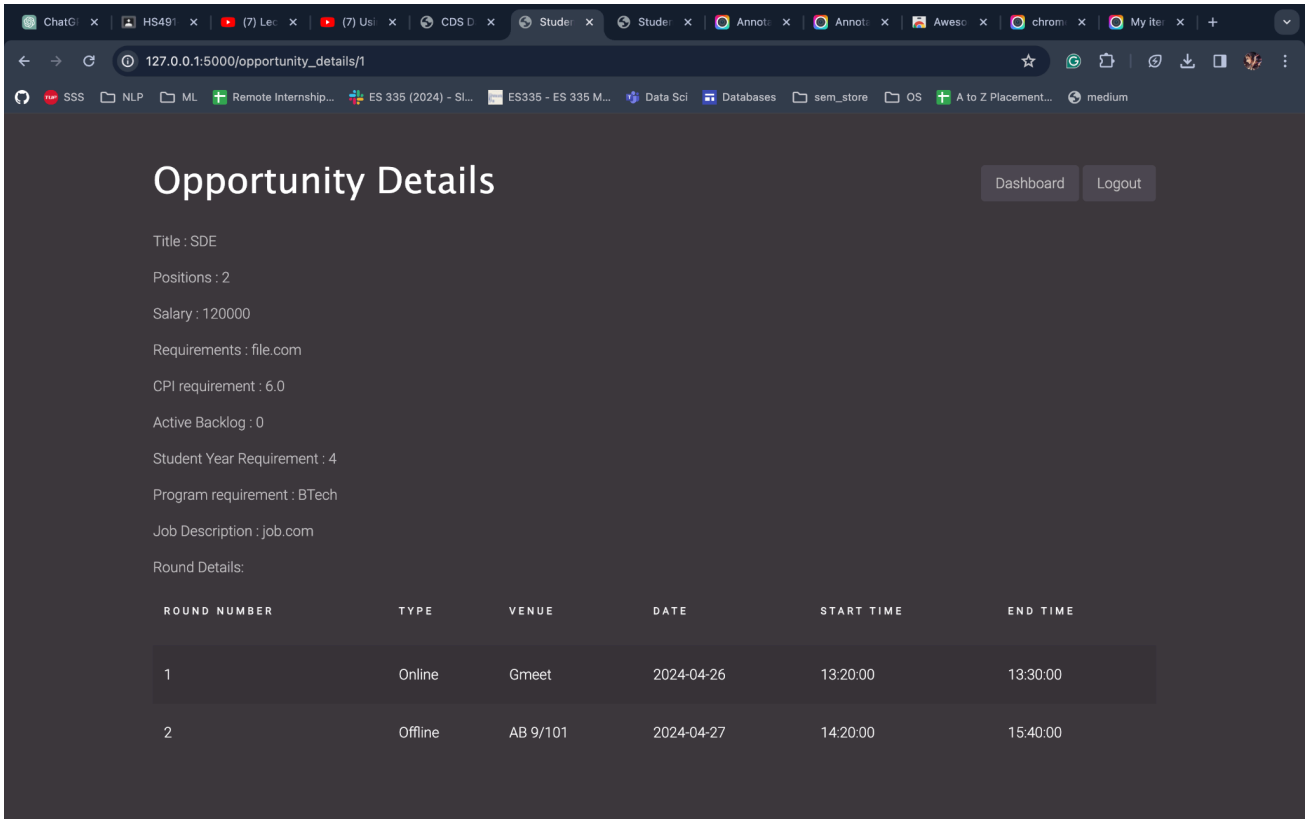
- Recruiter can host new Opportunities.



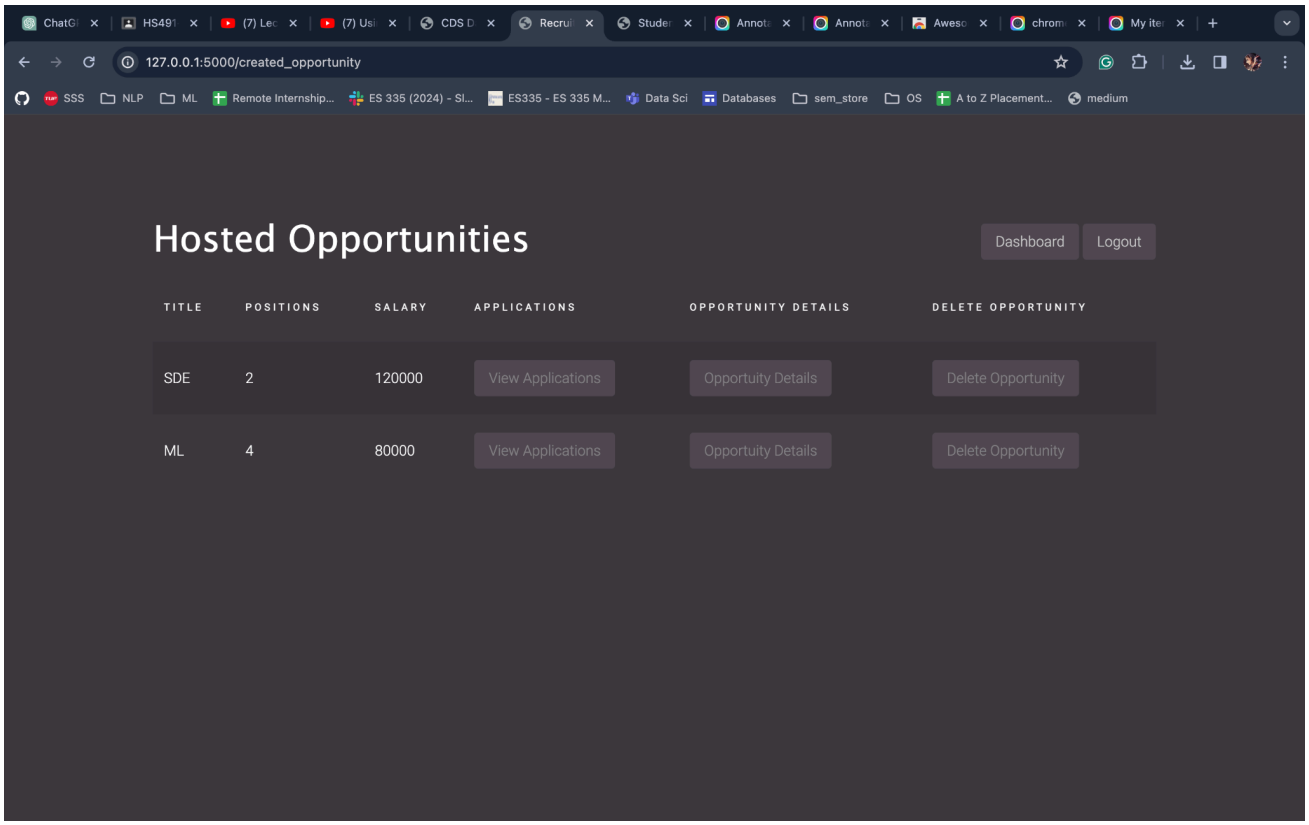
- Recruiter will get pop after successfully adding the opportunity.



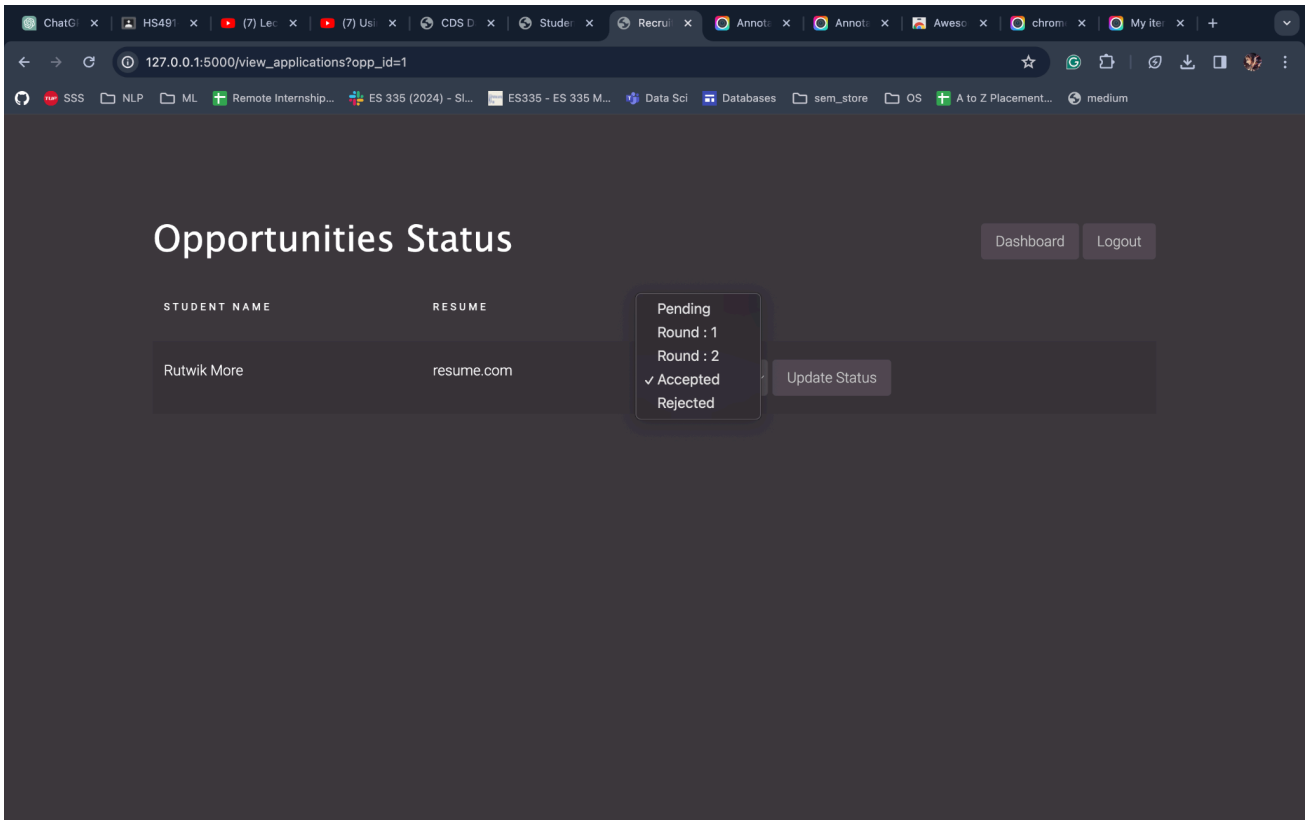
- Recruiter can see opportunities details of hosted opportunities.



- Recruiter can view applications submitted by student.

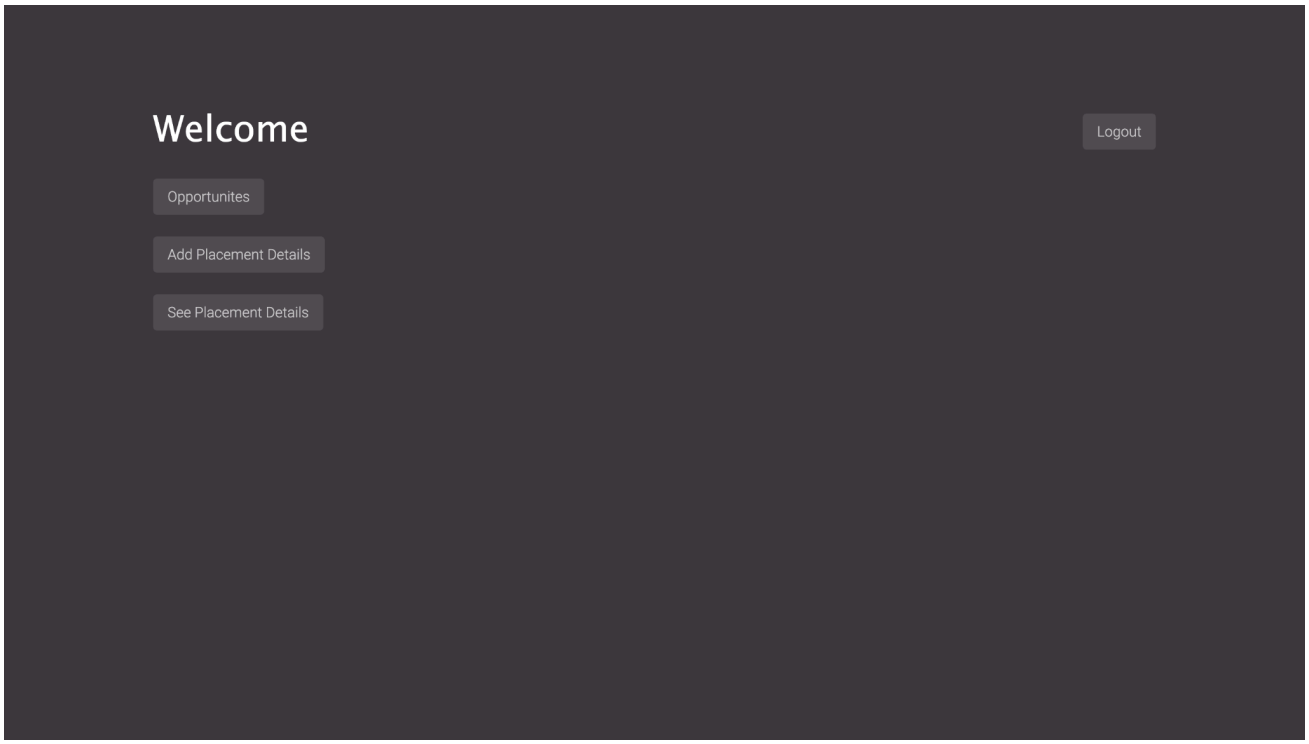


- Recruiter can change the status of opportunity of student.

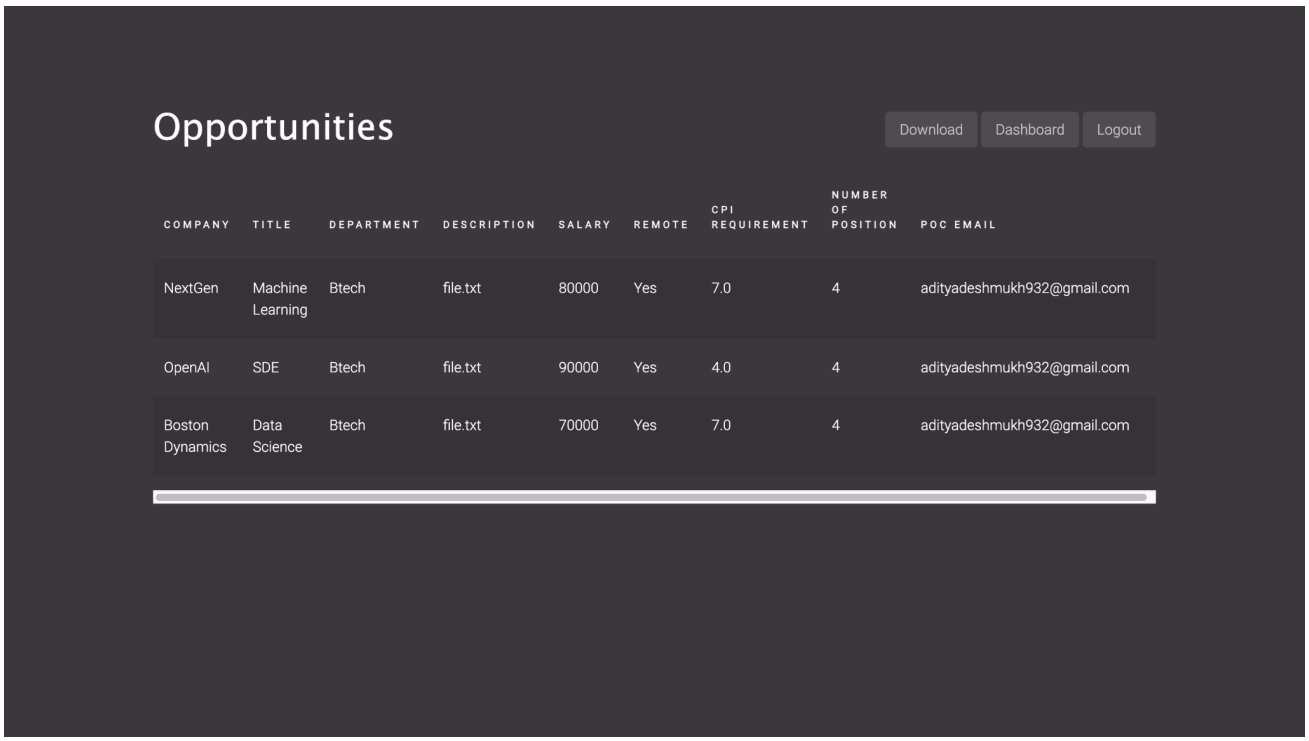


4. Placement manager view

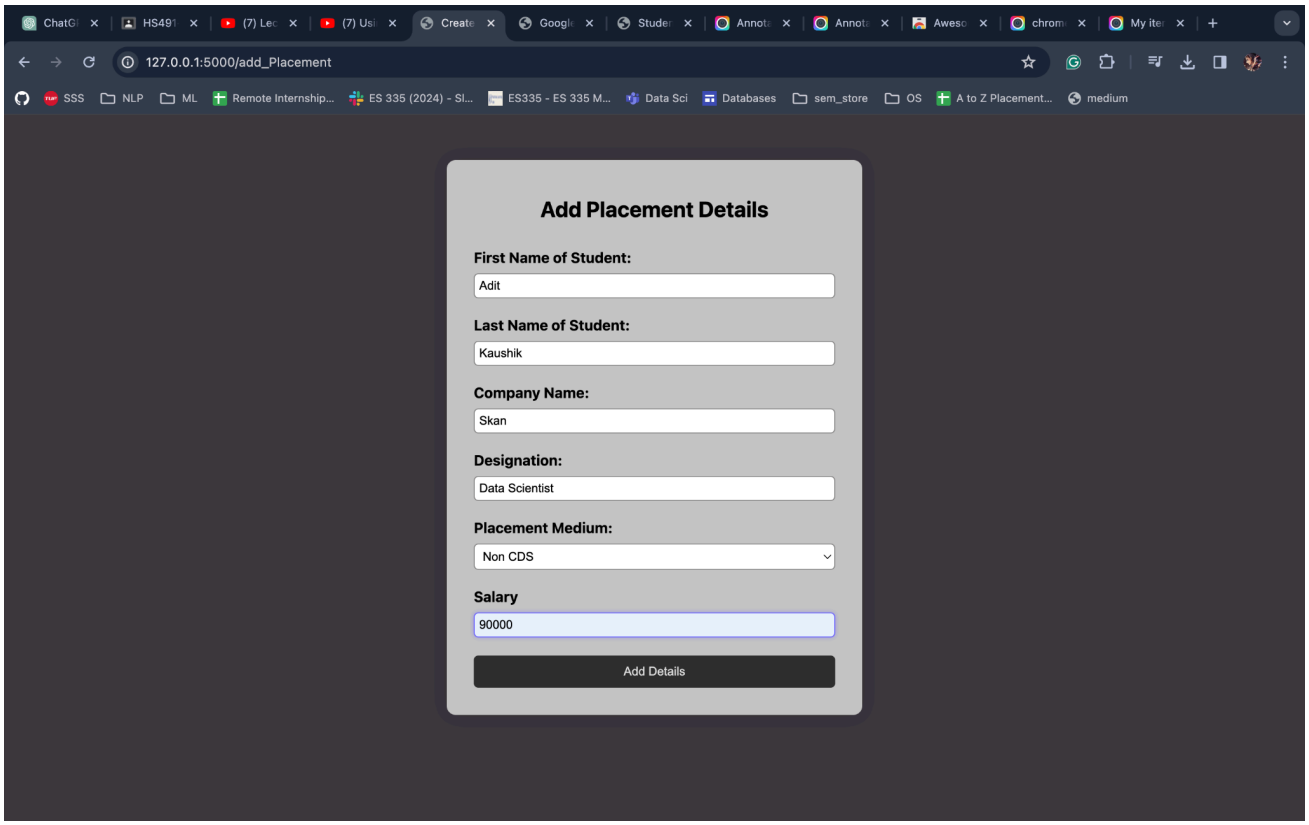
- Privileges:
    - Placement manager have access to add the placement details of the students
    - They can see the opportunities and also placement details add



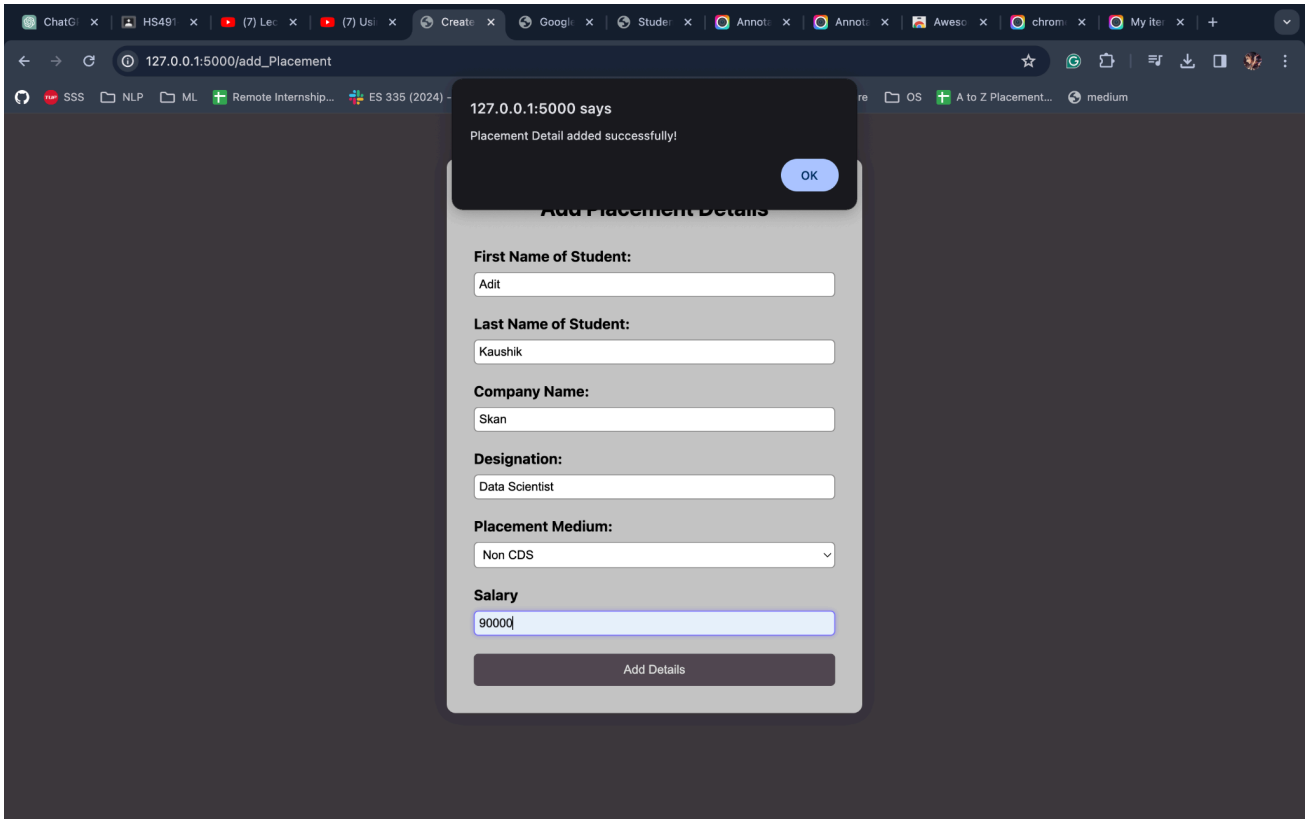
- Placement manager can view the opportunities available



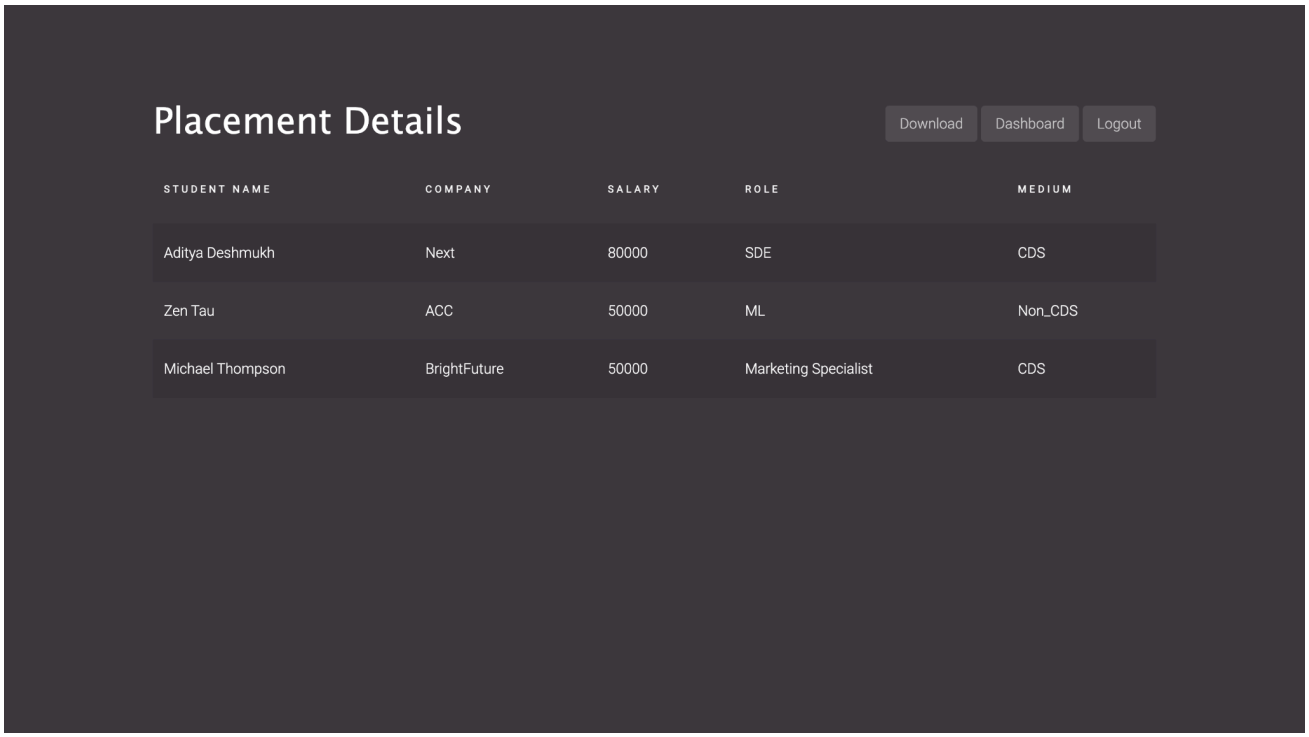
- The Placement Manager can add placement details of each student.



- CDS will pop up after adding the details successfully.



- Placement manager can see the details of the placed students

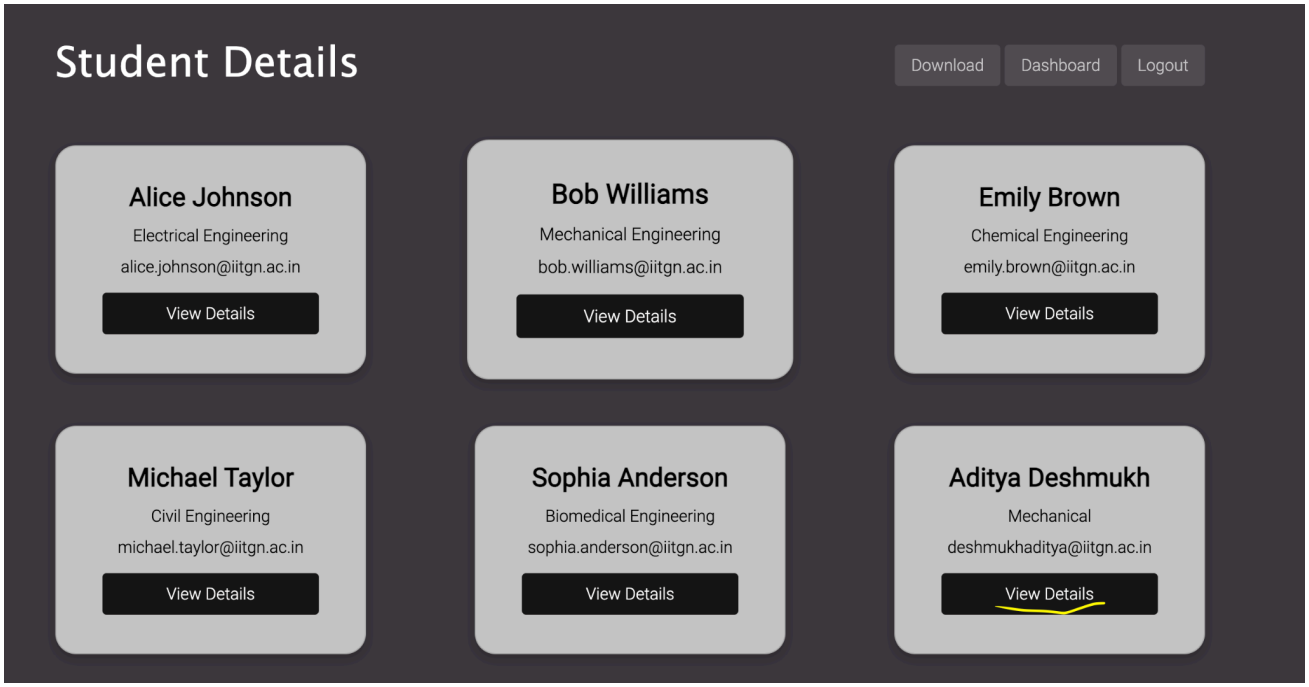


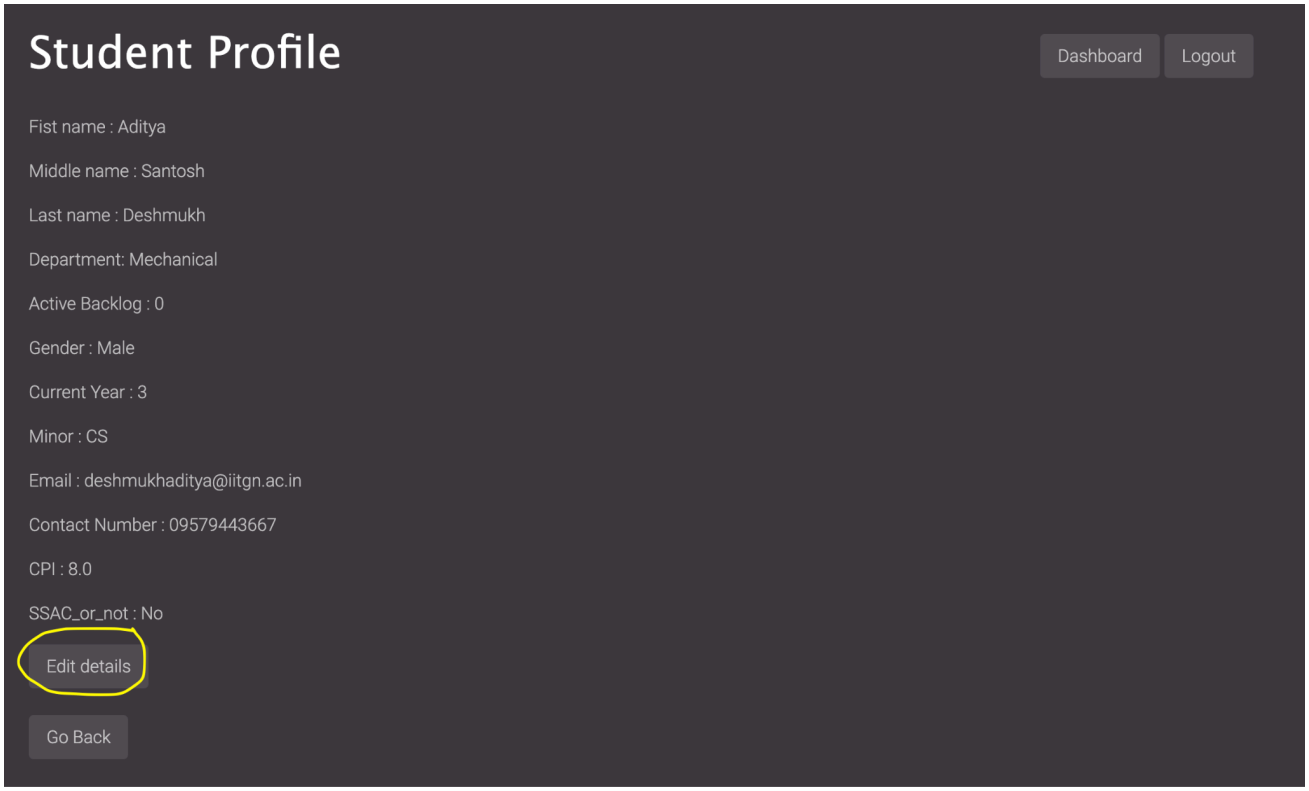
3.2 Responsibility of G2: **40 Pts.**

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.

- CDS editing student profile





- 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 =
values = (firstName, middleName, lastName, activeBacklog, department, gender, currentYear, studentImage, minors, contact
cur.execute(query, values)
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.**

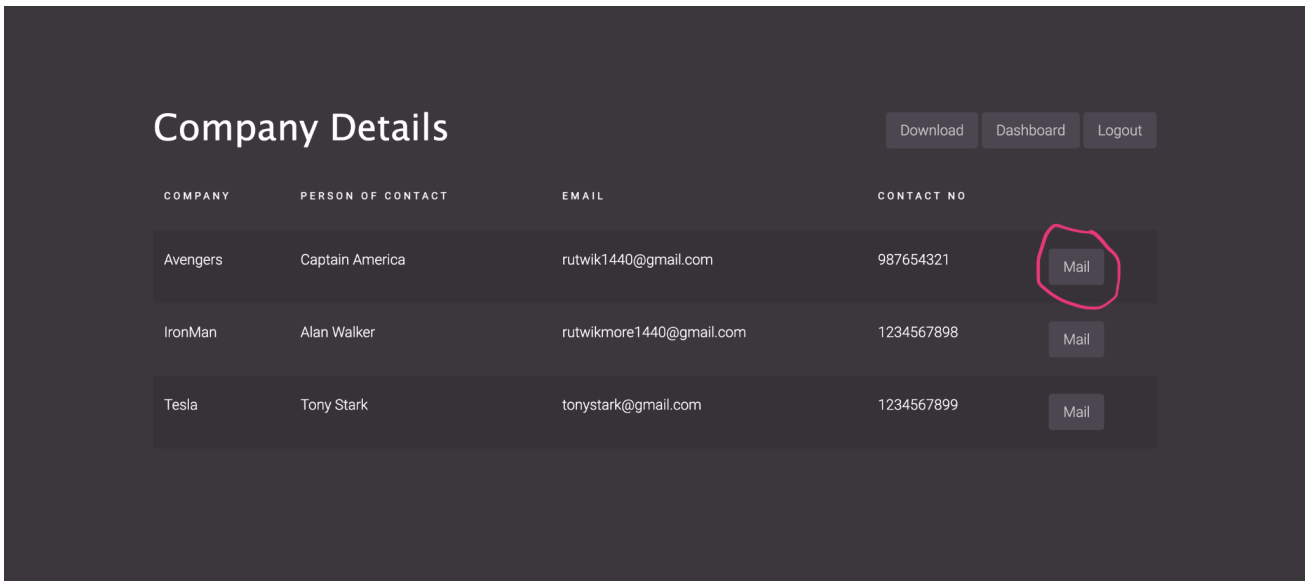
2. Implement the changes in the database as per the feedback received from stakeholders. **(20 Points)**

Changes made after the first feedback-

1. Mailing Feature

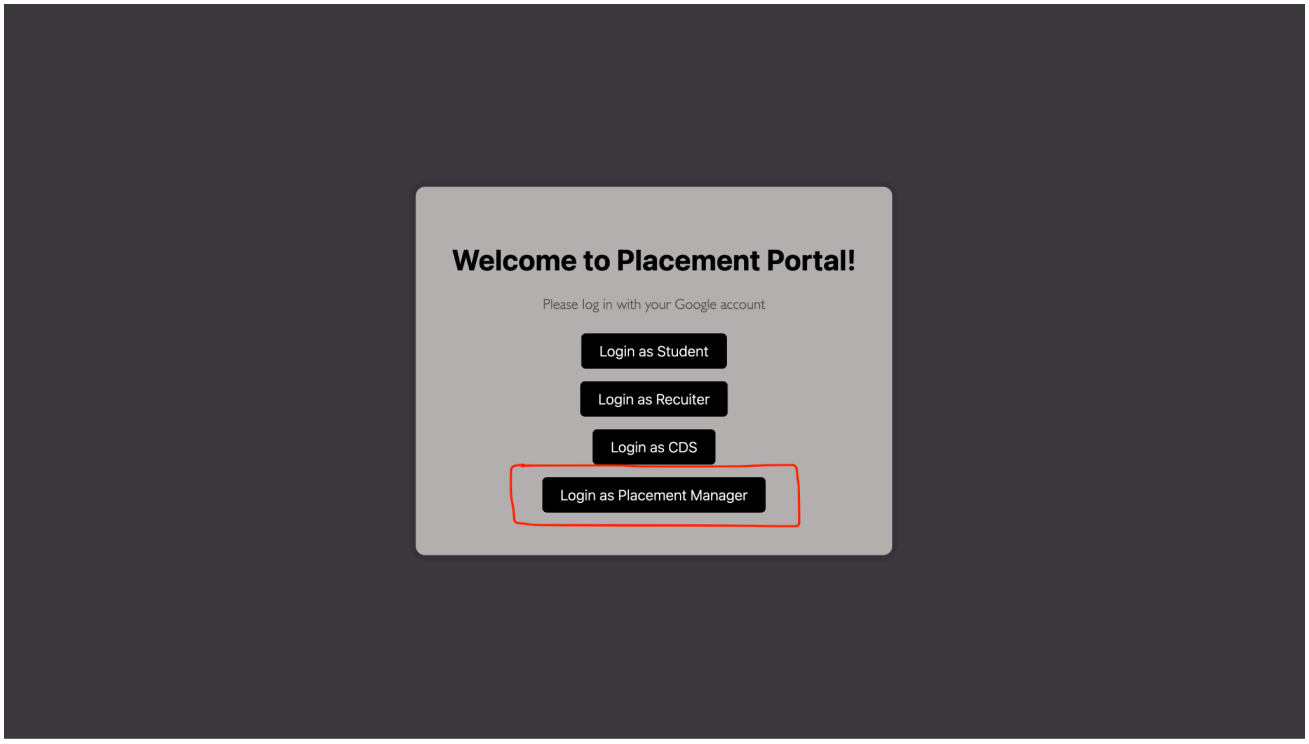
In our discussion with stakeholders, they suggested that it would be very inconvenient for CDS members to communicate with the recruiters through the mail. They will have to search for the email id of the recruiter and go to the Gmail app and then send the mail to them.

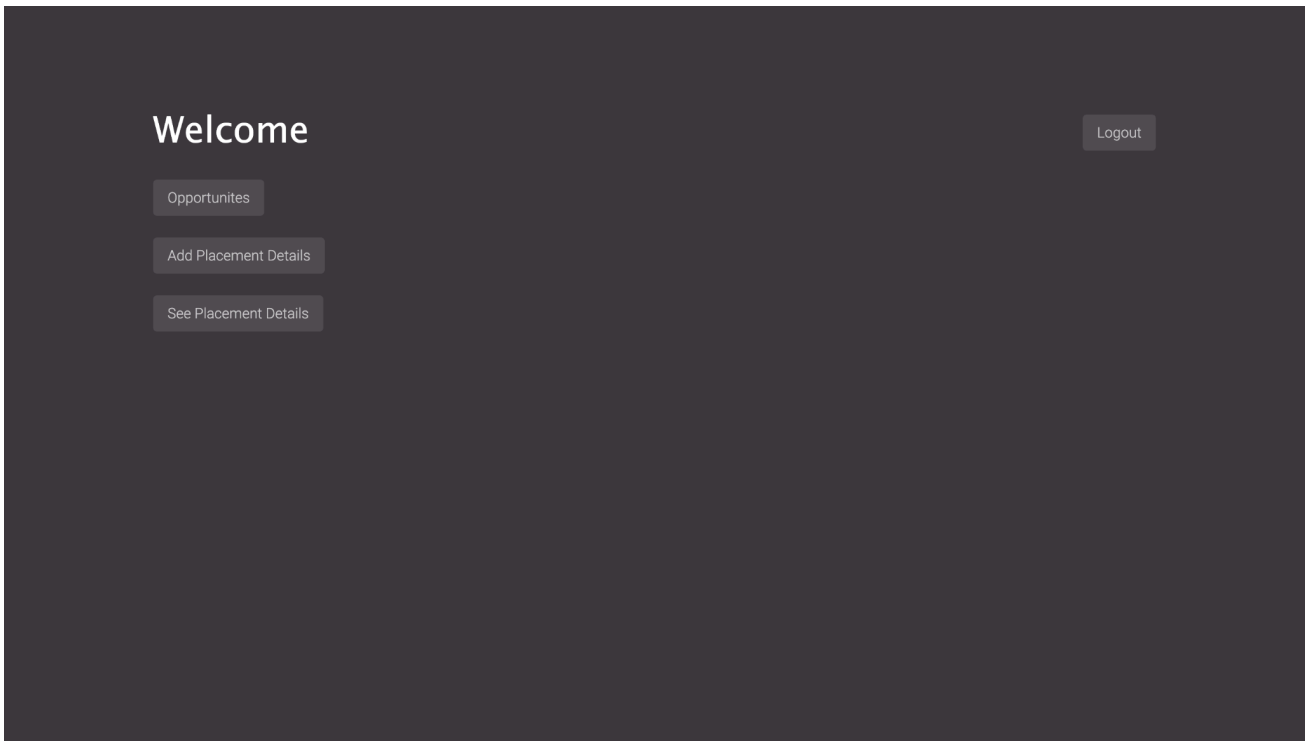
Instead of that, for the convenience of the CDS members, there should be a mail button alongside the name of the recruiter so that they can send the mail within our webapp.



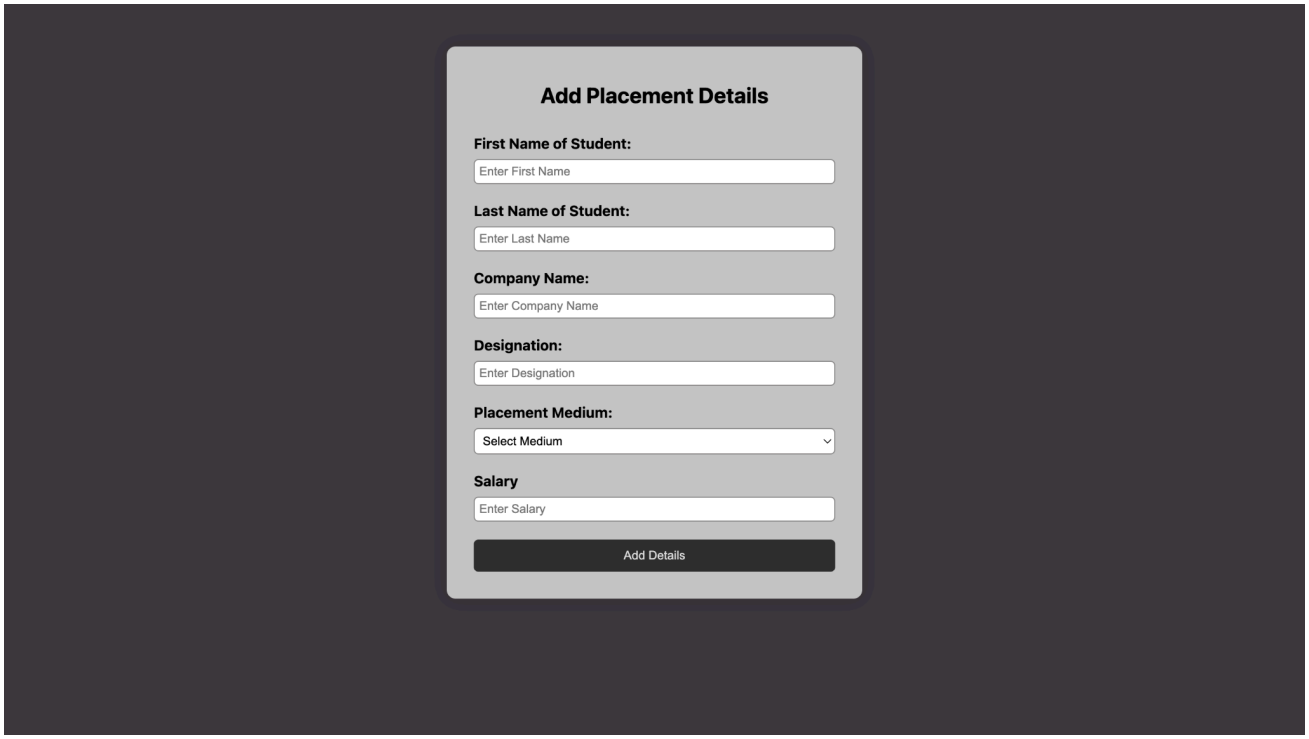
2. Add Placement manager - Give access of adding placement details

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.

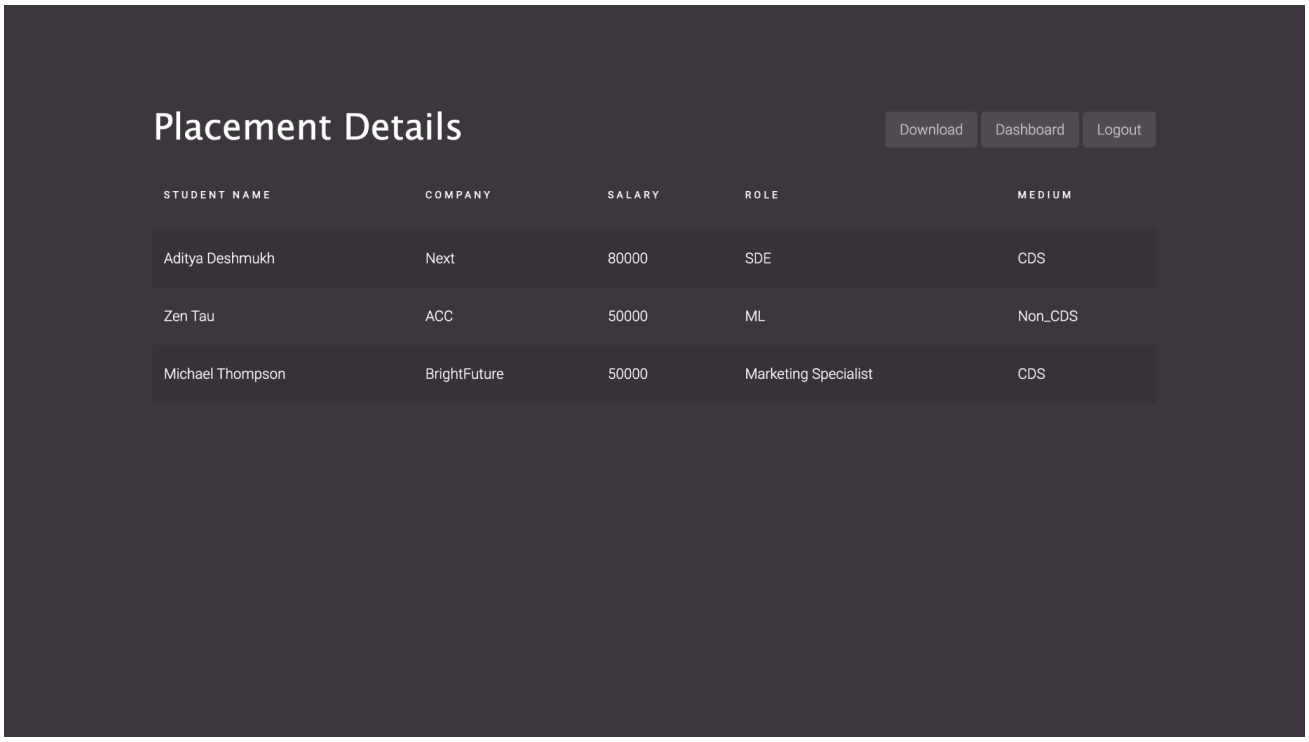




- Add Placement Details



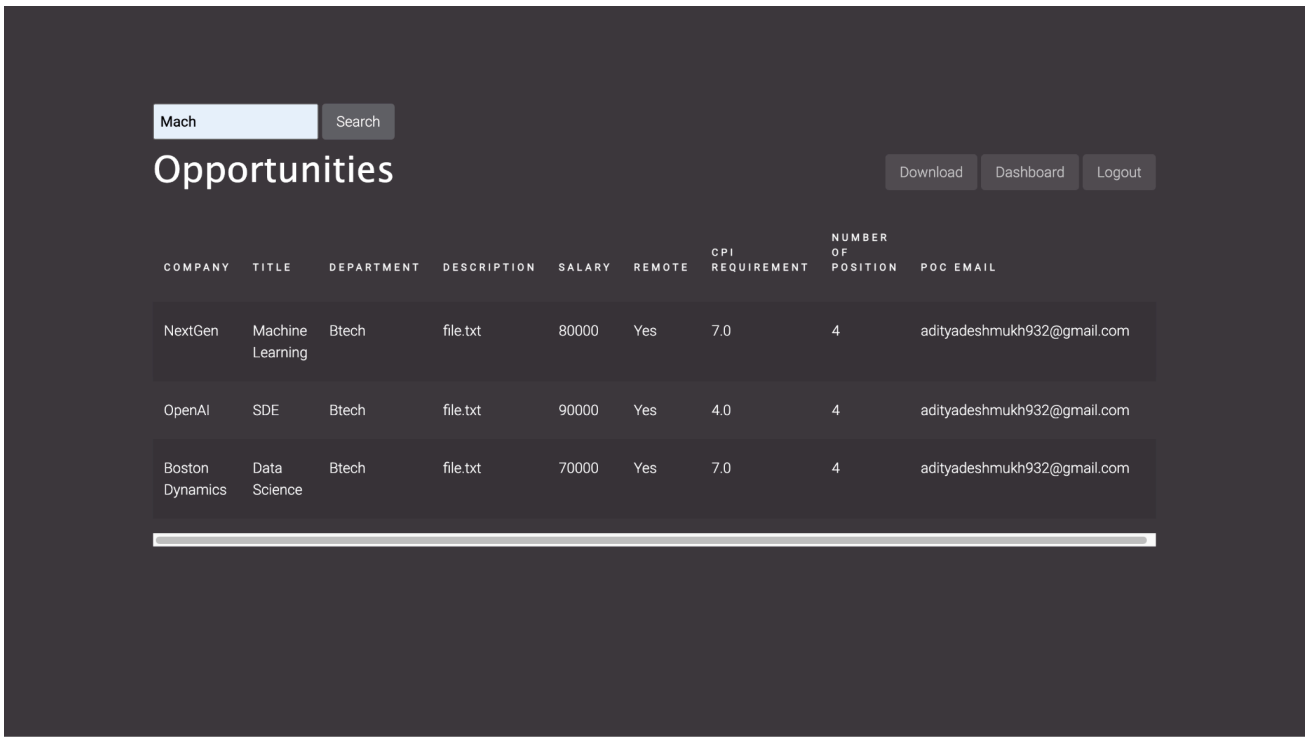
- See placement details

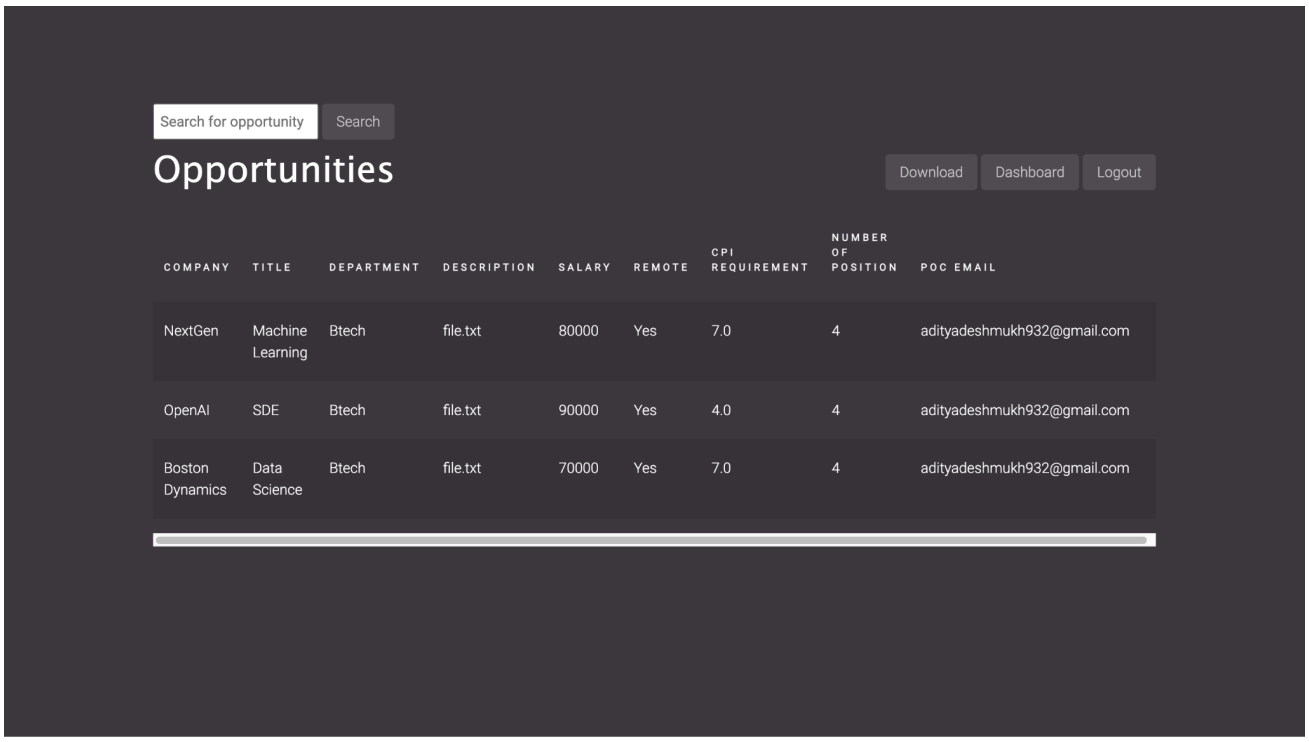


3. Search Feature

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.





If the query matches it give corresponding results else it gives

Search for opportunity | Search

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

Search for opportunity | Search

Opportunities

Download | Dashboard | Logout

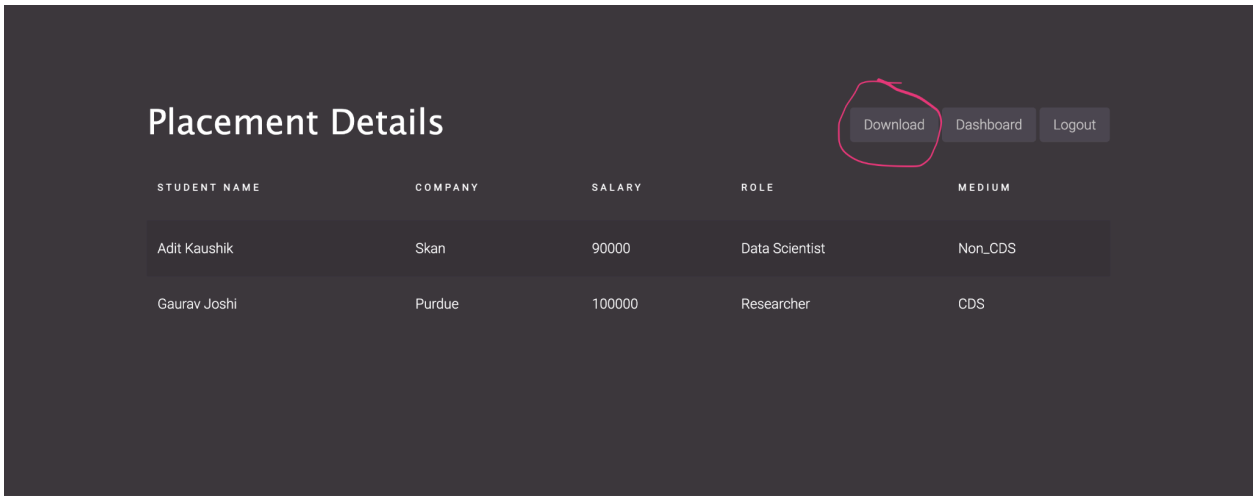
No opportunities available at the moment for the serached query : no

Changes made after the final feedback

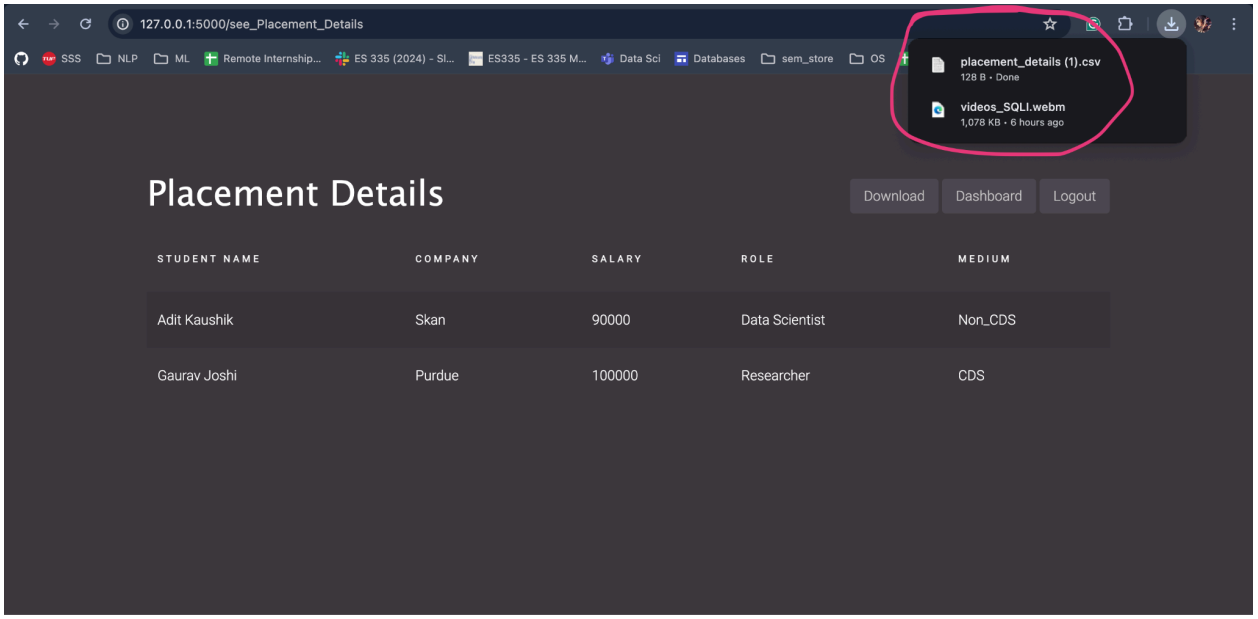
1. Download tables as CSV

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.



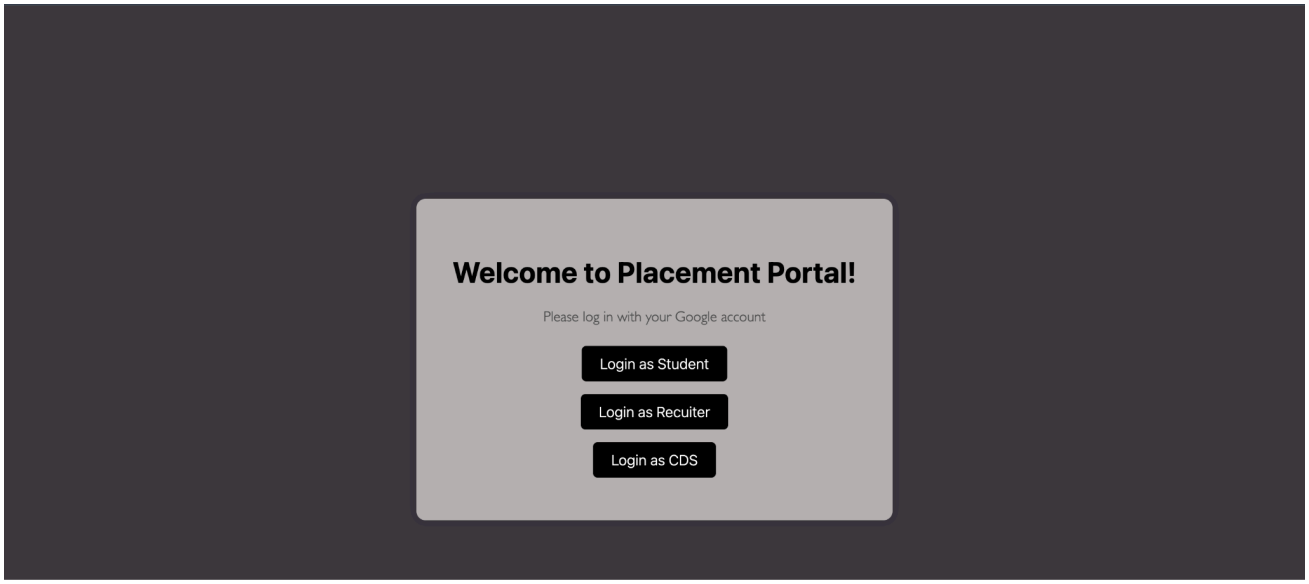
Here we can see that CSV has been downloaded.

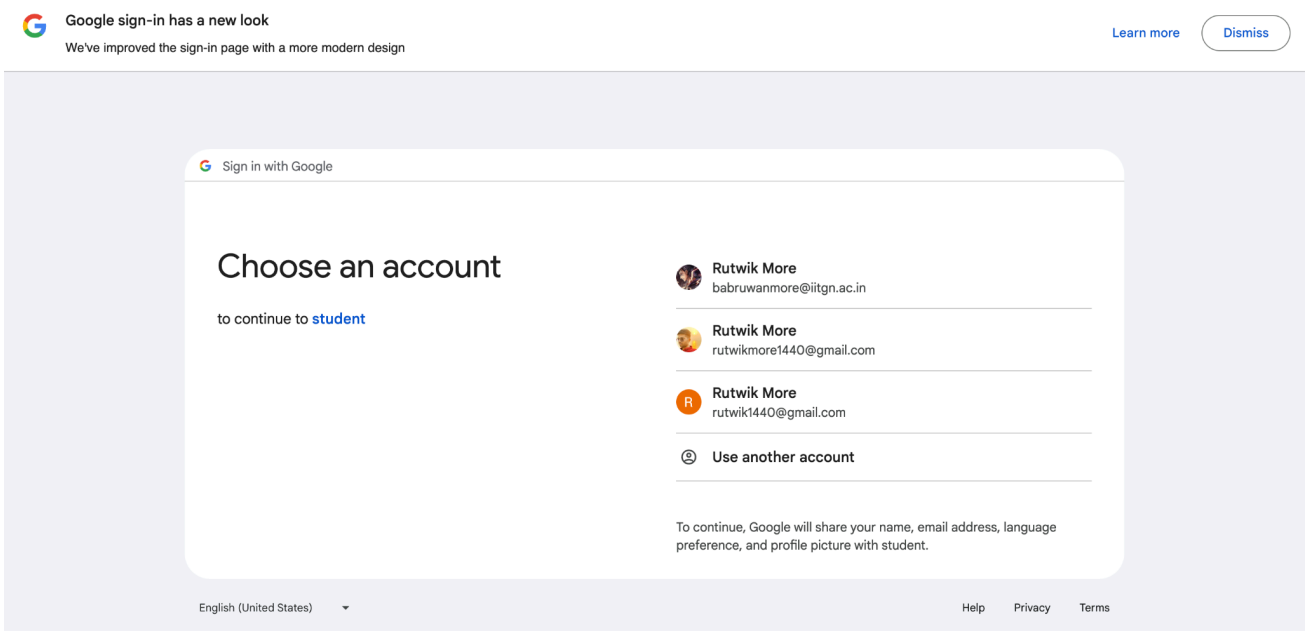


3.2 Responsibility of G2:

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.





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

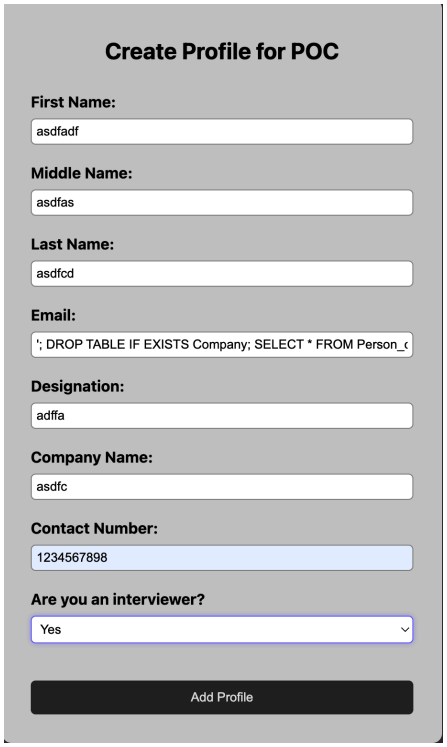
SQL injection is a common type of web attack in which an attacker inserts malicious SQL statements into an entry field, such as a login form or a search bar. This can result in the attacker being able to view, modify, or delete data from the application's database.

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

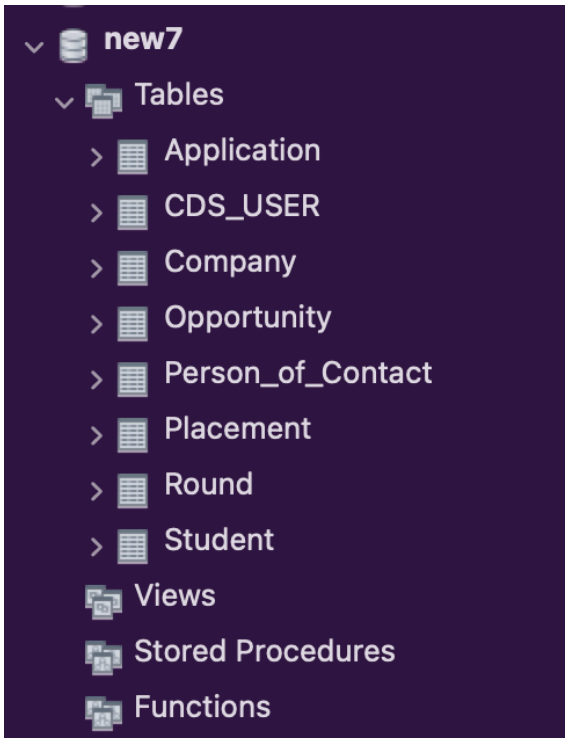
2) Then after creating the profile we create another new profile and while adding other entries we add following sql query in Email box input

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

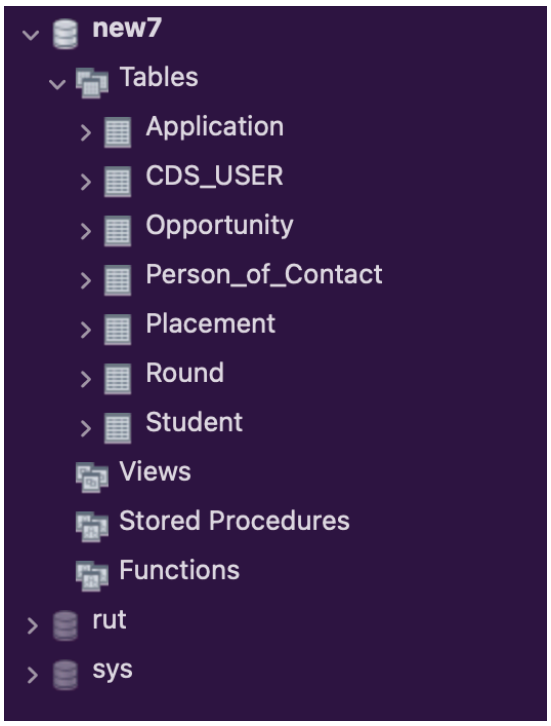


3) Here you can see email Id with email rutwikmore1440@gmail.com is present.

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, Employe
    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()
```

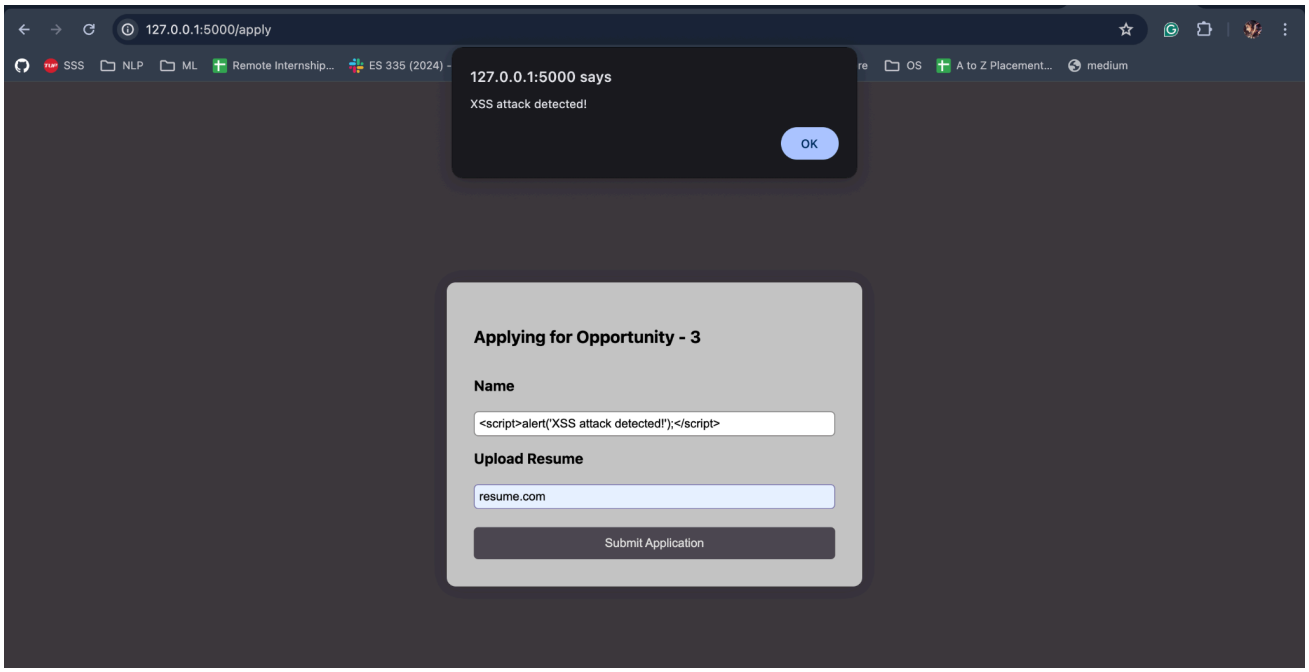
By using a parameterized query, you prevent any SQL injection attacks because the user's input is treated as a value rather than part of the SQL query.

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:

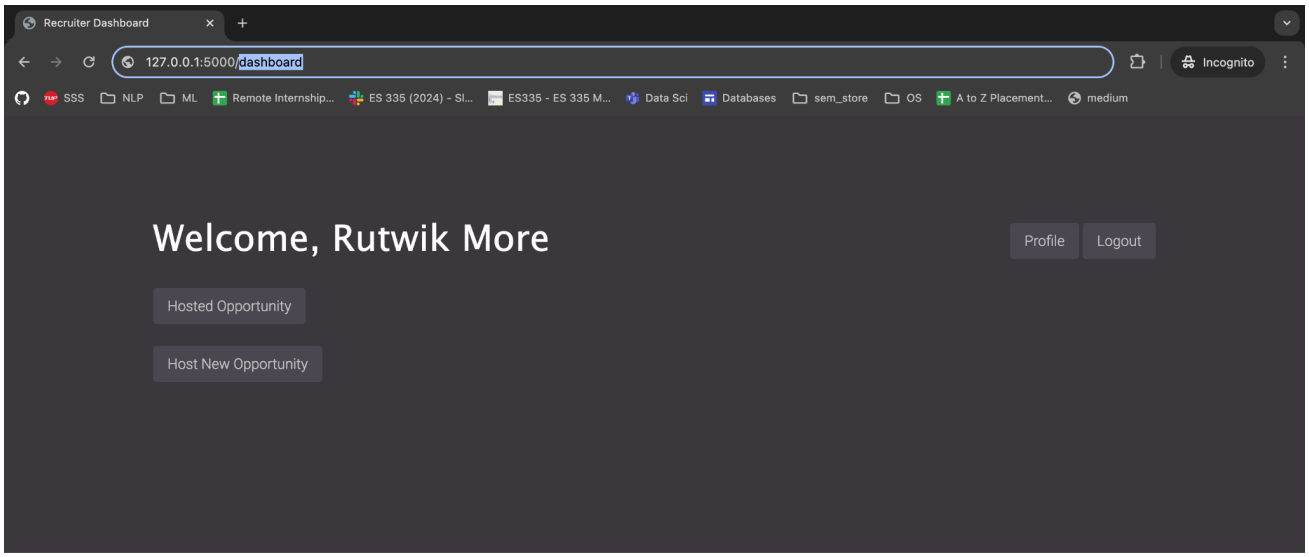
Use the encodeURIComponent() method to encode the input data before sending it to the server to prevent special characters from being interpreted as HTML code.

```
$('#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)

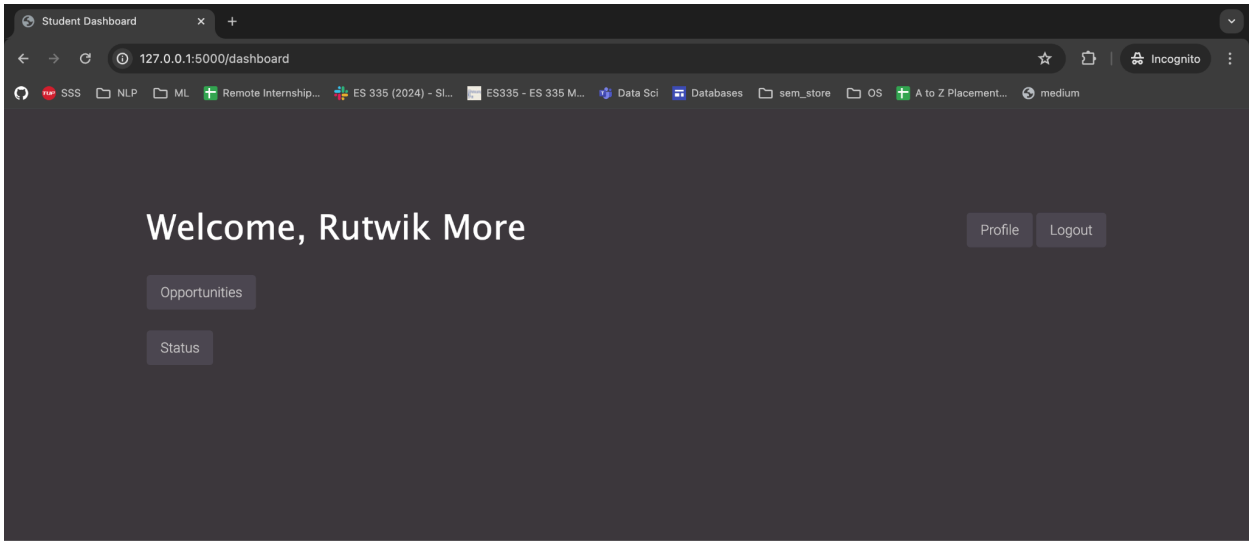
3. Insecure direct Object references (IDOR) (URL Attack)

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.

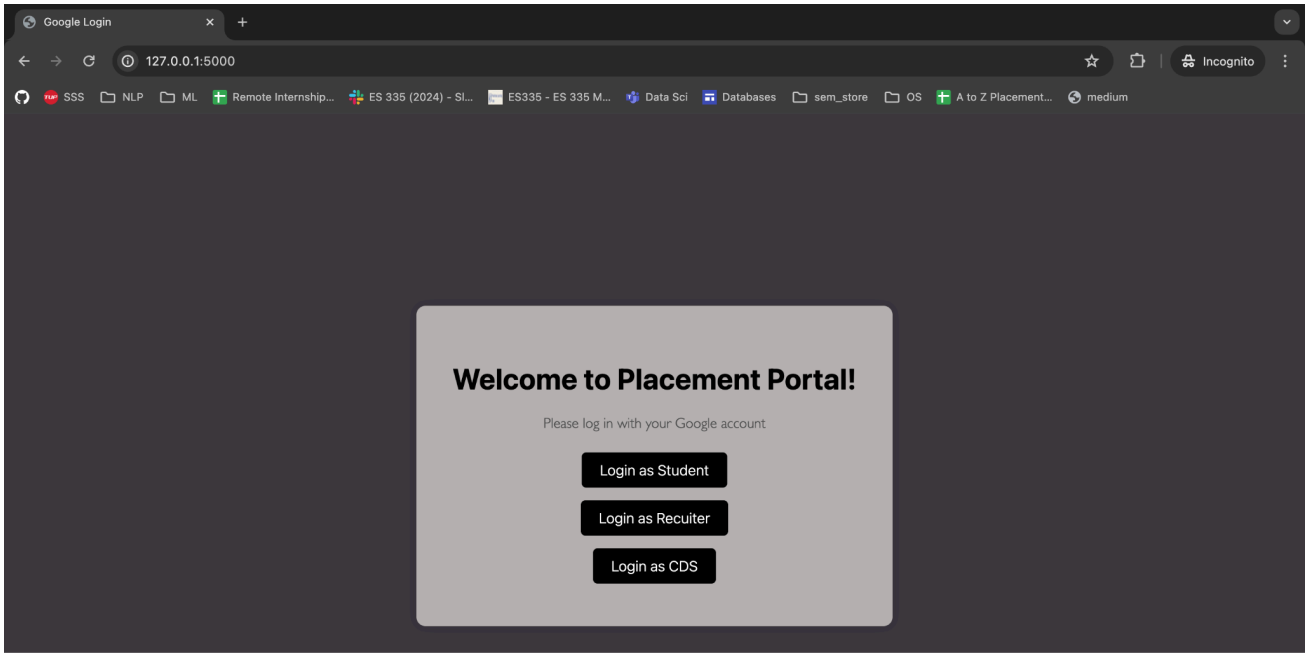


2) The user inspects the HTML source code of the page they are currently on and finds a link or reference to the admin page URL. Such as the Recruiter can see the URL for student dashboard and manually edits the URL in the browser's address bar to replace the current page's URL with the student dashboard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

```python
@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.

```python
@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

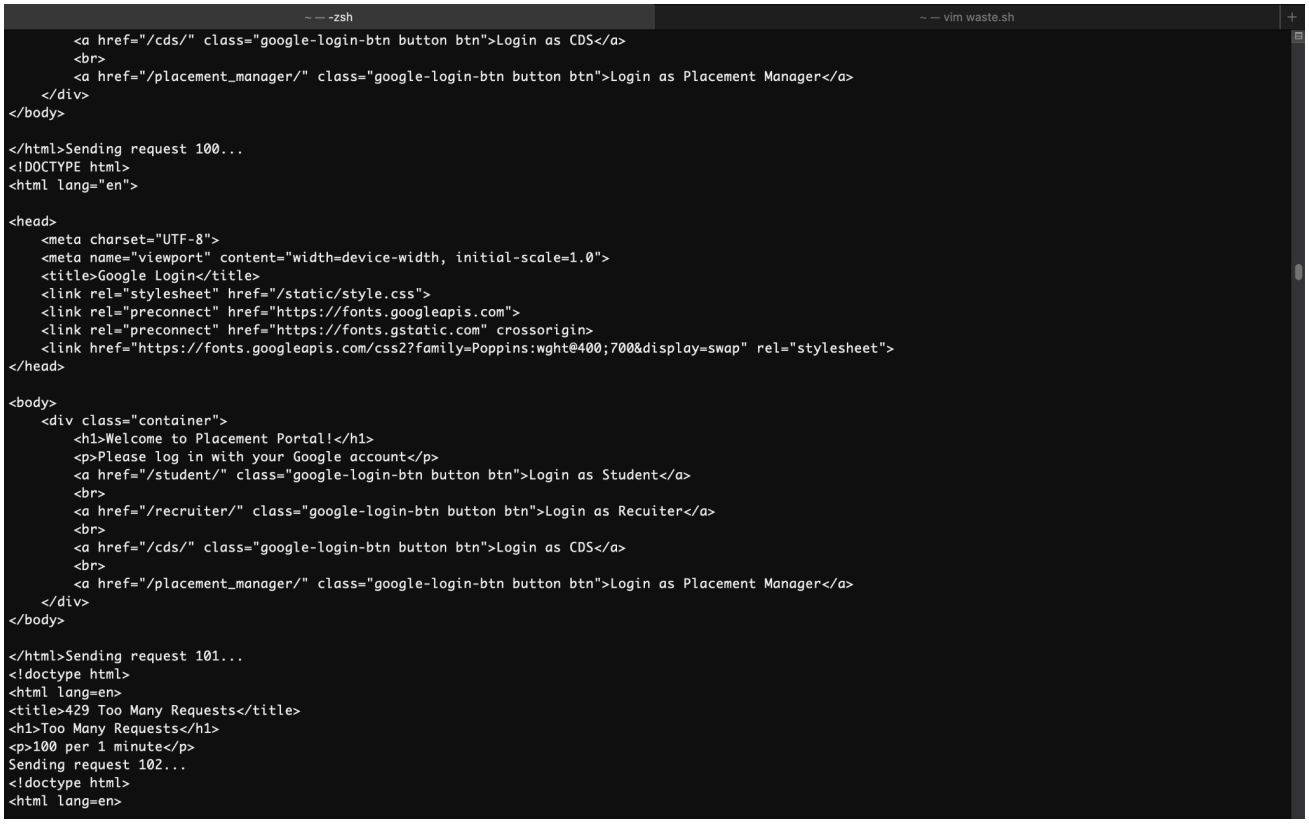
- A denial-of-service (DoS) attack is a method used to render a machine or network inaccessible by overwhelming it with an excessive amount of traffic. This results in the target system failing to function properly, thereby preventing legitimate users from accessing its services.
- To carry out the attack, 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

```bash
#!/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

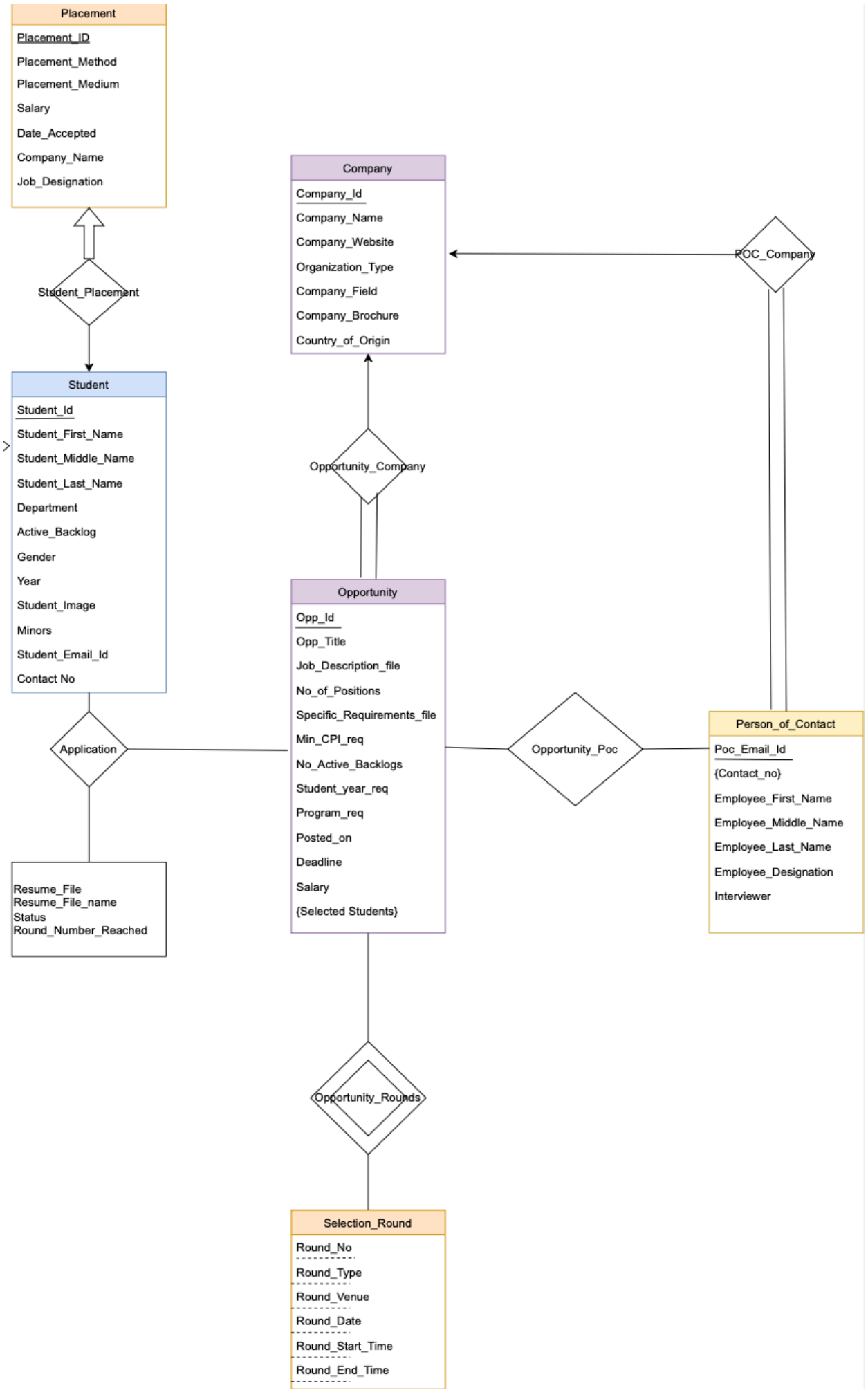
```python
from flask import Flask, render_template
from flask_limiter import Limiter
from flask_limiter.util import get_remote_address


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



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)**


Placement
Placement_ID
Placement_Method
Placement_Medium
Salary
Date_Accepted
Company_Name
Job_Designation
Student_Placement
Student
Student_Id
Student_First_Name
Student_Middle_Name
Student_Last_Name
Department
Active_Backlog
Gender
Year
Student_Image
Minors
Student_Email_Id
Contact No
Application
Resume_File
Resume_File_name
Status
Round_Number_Reached
Company
Company_Id
Company_Name
Company_Website
Organization_Type
Company_Field
Company_Brochure
Country_of_Origin
POC_Company
Opportunity_Company
Opportunity
Opp_Id
Opp_Title
Job_Description_file
No_of_Positions
Specific_Requirements_file
Min_CPI_req
No_Active_Backlogs
Student_year_req
Program_req
Posted_on
Deadline
Salary
{Selected Students}
Opportunity_Poc
Person_of_Contact
Poc_Email_Id
{Contact_no}
Employee_First_Name
Employee_Middle_Name
Employee_Last_Name
Employee_Designation
Interviewer
Opportunity_Rounds
Selection_Round
Round_No
Round_Type
Round_Venue
Round_Date
Round_Start_Time
Round_End_Time

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)

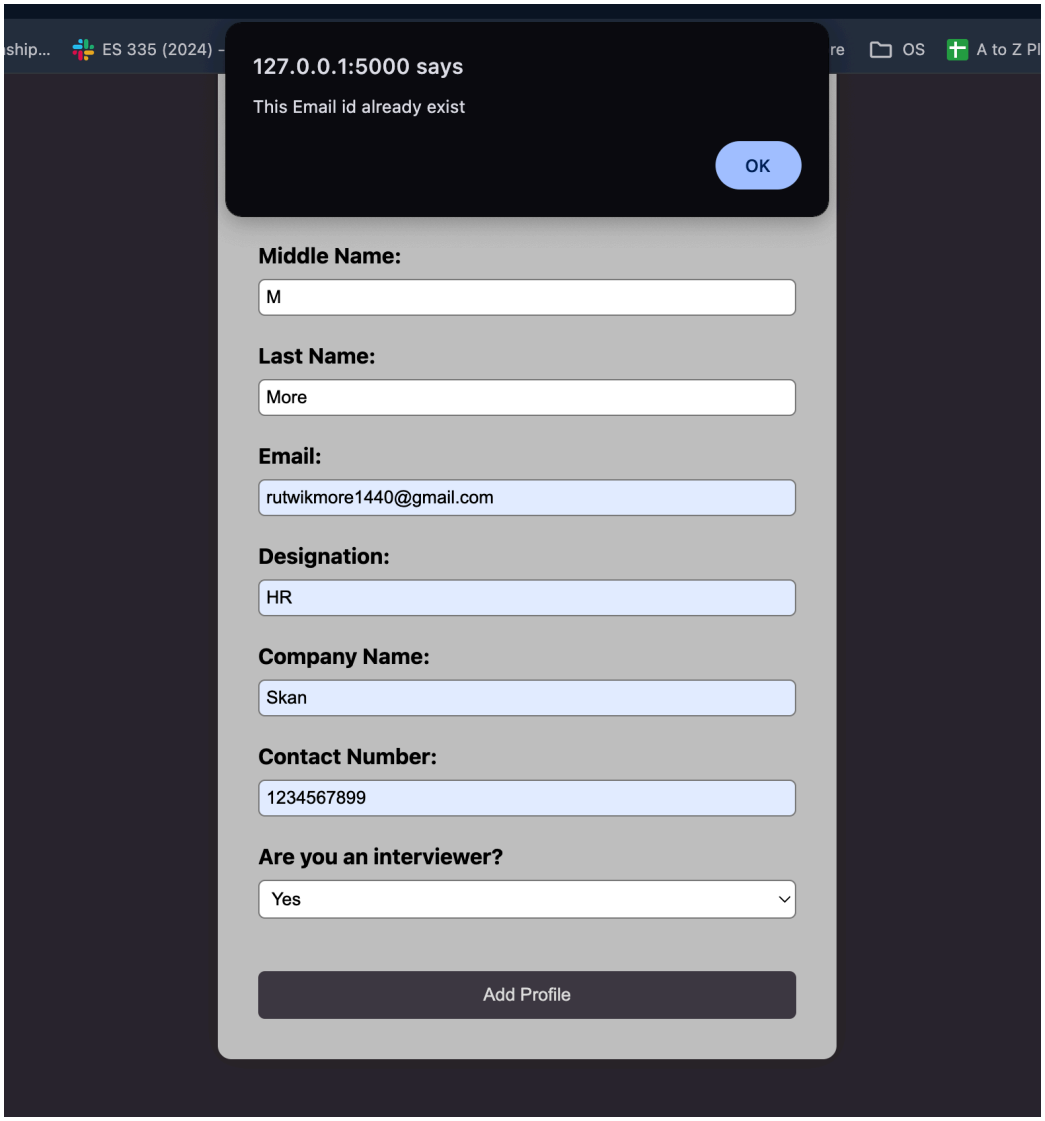
Opportunity (Opp_Id, Opp_Title, Company_Id, Company_Name, No_of_Positions, Specific_Requirements_file, Min_CPI_req, No_Active_Backlogs, Student_year_req, Program_req, Job_Description_file, Posted_on, Deadline, Salary, {Selected Students})

Person_of_Contact (Poc_Email_Id, Company_Id, {Contact_no}, Employee_First_Name, Employee_Middle_Name, Employee_Last_Name, Employee_Designation, Interviewer, Company_Name)

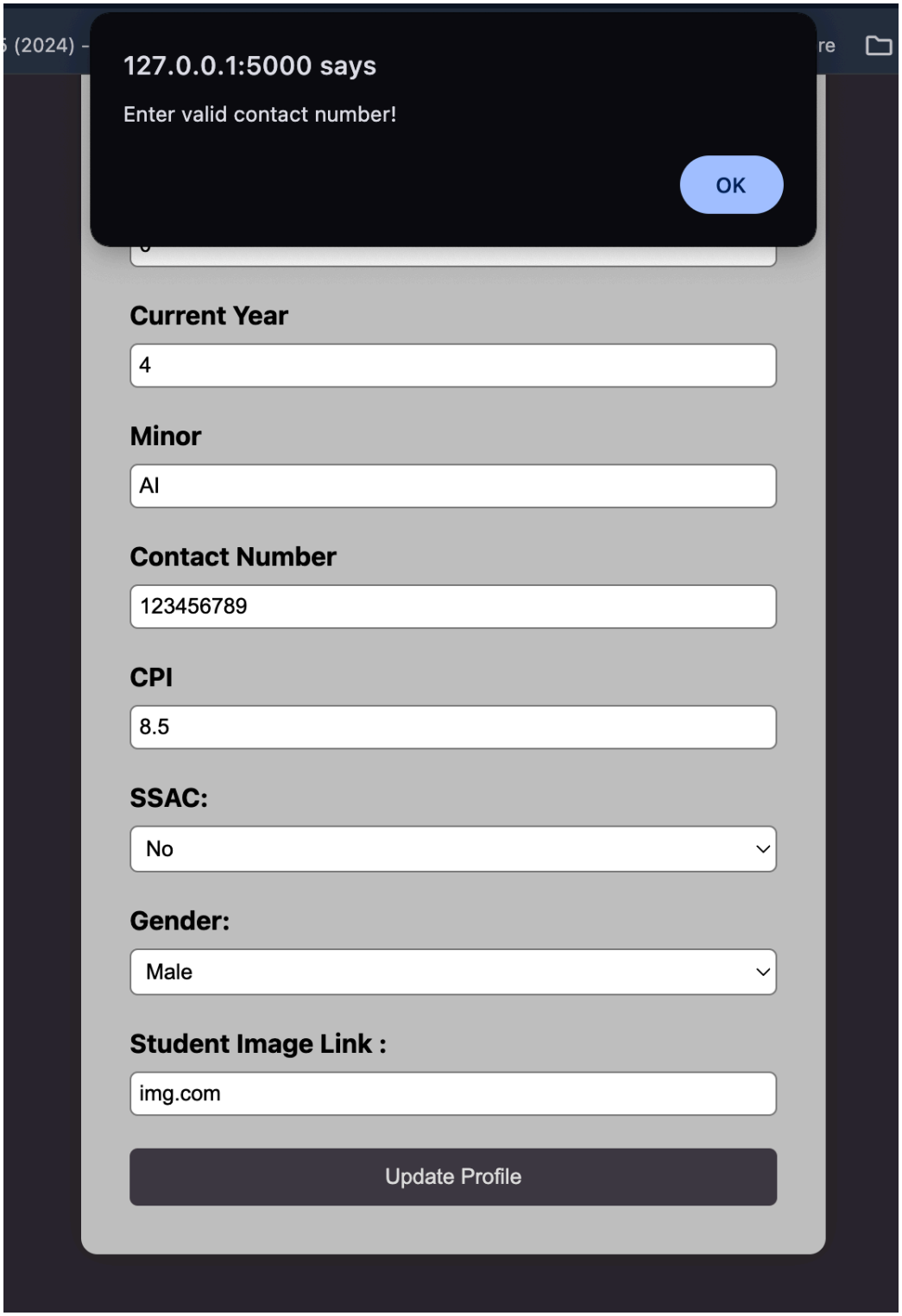
Placement (Placement_Id, Placement_Method, Placement_Medium, Salary, Date_Accepted, Company_Name, Job_Designation)

Selection_round (Round_No, Round_Type, Round_Venue, Round_Date, Round_Start_Time, Round_End_Time)

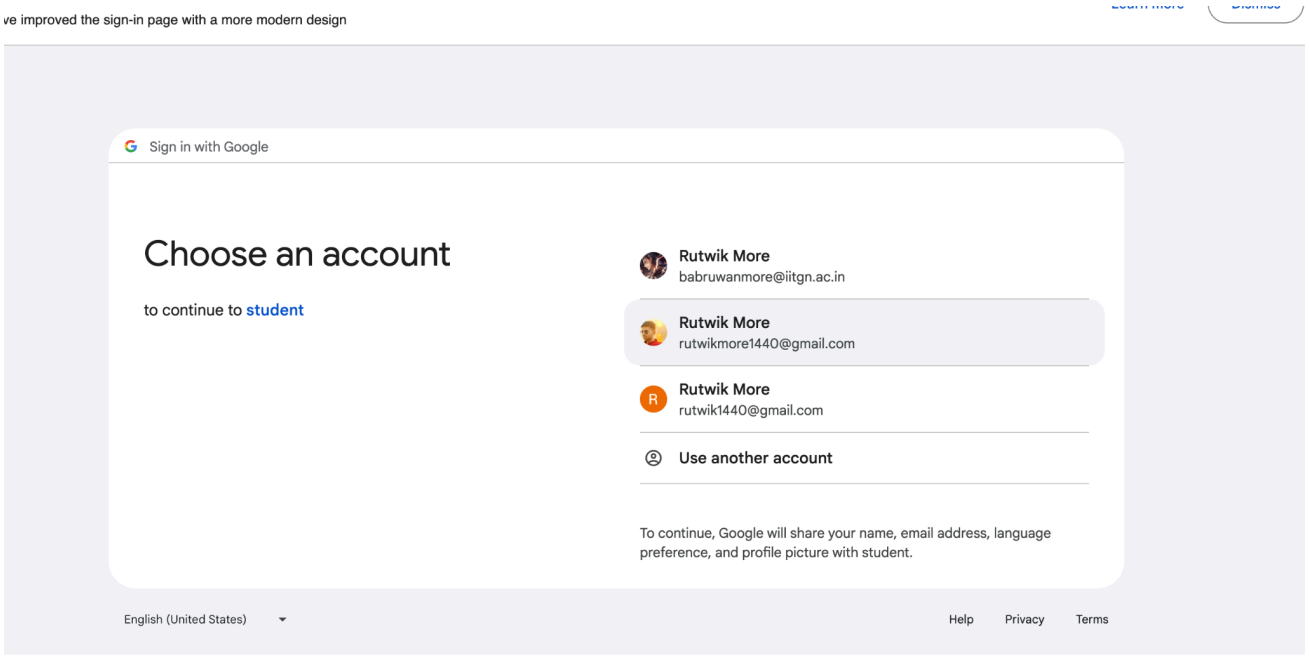
- When the person of contact email already exists in the database it gave the following pop up



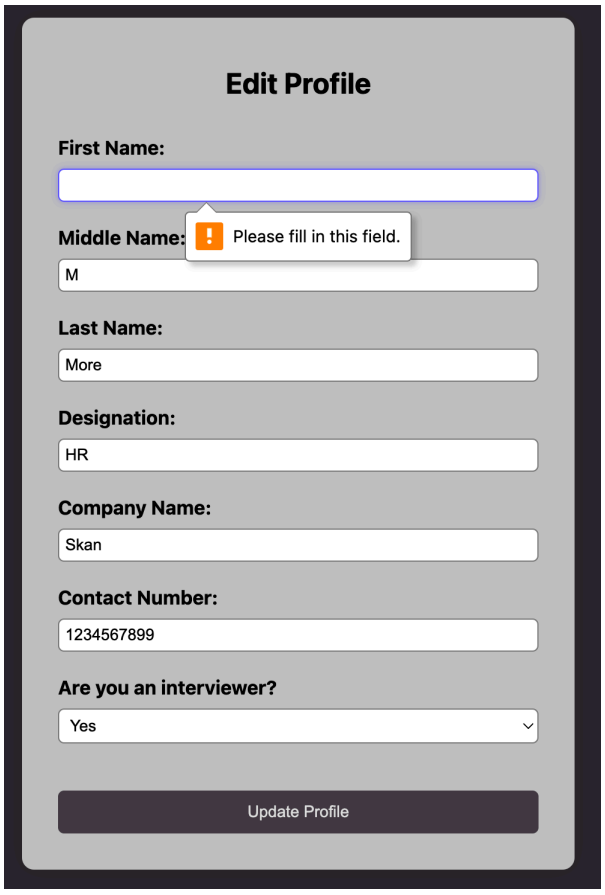
- When we entered invalid phone number i.e not of 10 digits it gave the below pop up



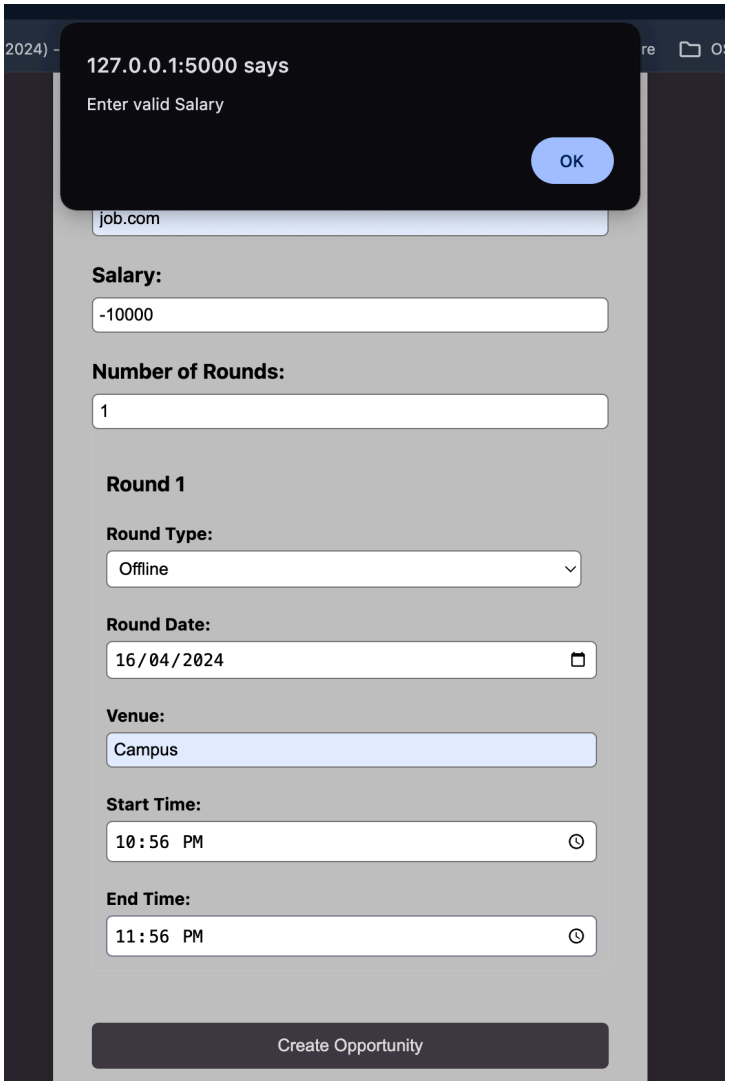
- Students cannot login without an IITGN email. If someone tries to login with a non IITGN email id then it redirects the user to the home page.



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



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/1thD8FtA0pBjFwbtnWxz85L4WhKXOa4s-

3) Generated Dataset for the Database: Google Drive:
   https://drive.google.com/drive/folders/1lr40jKZI_w58t7_DY7AVztuXKtXEc2A2

4) SQL Files, and data dump:
   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	• Contributed in making CDS hierarchy code in backend. • Debugged parts of the authentication code. • Contributed to documentation. • Interacted with the stakeholders and took the feedback.
Abhay Kumar Upparwal	G1	21110004	• Reviewed the initial feedback, assessed alterations, and brainstormed ideas for enhancements. • Contributed to the documentation.
Aaryan Darad	G1	21110001	• Worked on making the report of the project • Interacted with the stakeholders and understood the feedback, reviewed and communicated the suggested changes. • Incorporated the feedback and ensured that all required features are implemented.
Ahaan Giriya	G1	21110015	• Suggested some improvements • Added CSV download functionality in the Webapp after first feedback • Formatted the document
Gaurav Joshi	G2	21110065	• 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 • Documented the changes in the database as per the feedback received from stakeholders. • Working on building XSS attack

			and IDOR (URL) attack. Also built defenses against these attacks.
Adit Kaushik	G2	21110010	• Contributed for get requests and SQL commands for company POC and CDS Employee required for different requests • Wrote the setup requirements and steps to run the Web App. • Contributed to the DOS attack and XSS and the solution for it
More Rutwik	G2	21110133	• Working on building SQL injection and IDOR (URL) attack. Also built defenses against these attacks. • Added Mail functionality in the Webapp after first feedback • Added CSV download functionality in the Webapp after first feedback
Aditya Deshmukh	G2	21110014	• Contributed to the DOS attack and the solution for it • Contributed in making placement manager backend • Added search functionality