CS 432: Databases
Assignment 3: Implementing a Web App using MySQL

Topic: Placement Management

Group Members

G1:
Mihir Sutariya (20110208)
Pushpendra Pratap Singh (20110151)
Sanskar Sharma (20110185)
Shruhrid Banthia (20110198)
Tanvi Dixit (20110212)
Utkarsh Mishra (20110218)

G2:
Dhyey Kumar Thummar (20110059)
Ksheer Sagar Agrawal (20110098)
R Yeeshu Dhurandhar (20110152)
Saatvik Rao (20110175)
Sahil Agrawal (20110178)

3.1 Responsibility of G1:

1. Changes Made After First Feedback:

Stakeholders: Mr. Saumil Shah (CDS officer), Aditya Bhujbal (Student Placement Officer)

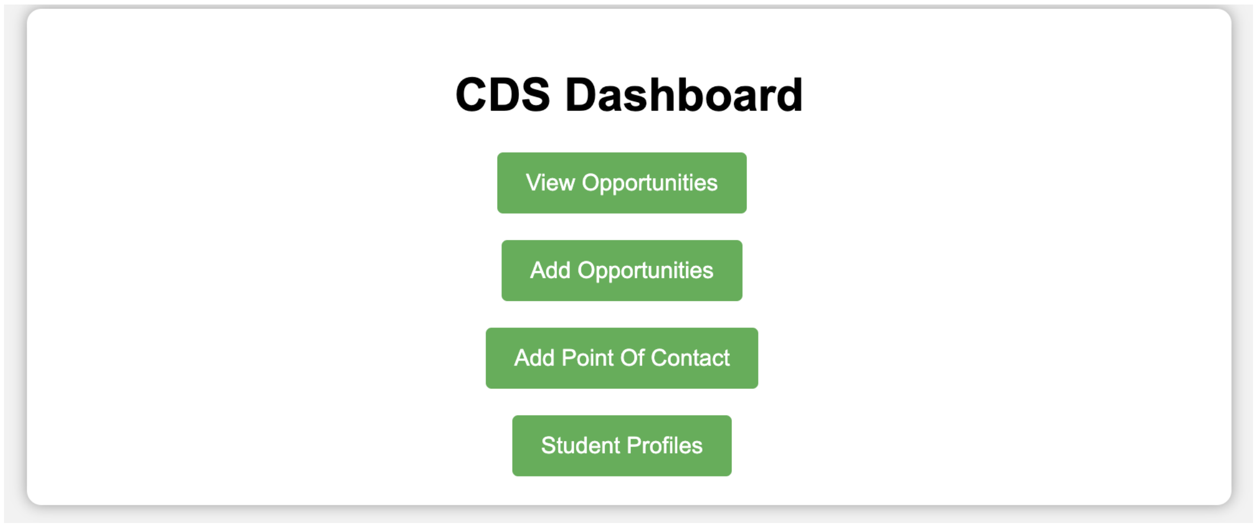
Feedback Received From Stakeholders:

1. *Round details updating feature is missing to notify students about the details of the current round that they are in. The CDS team should be able to update the round details too.*

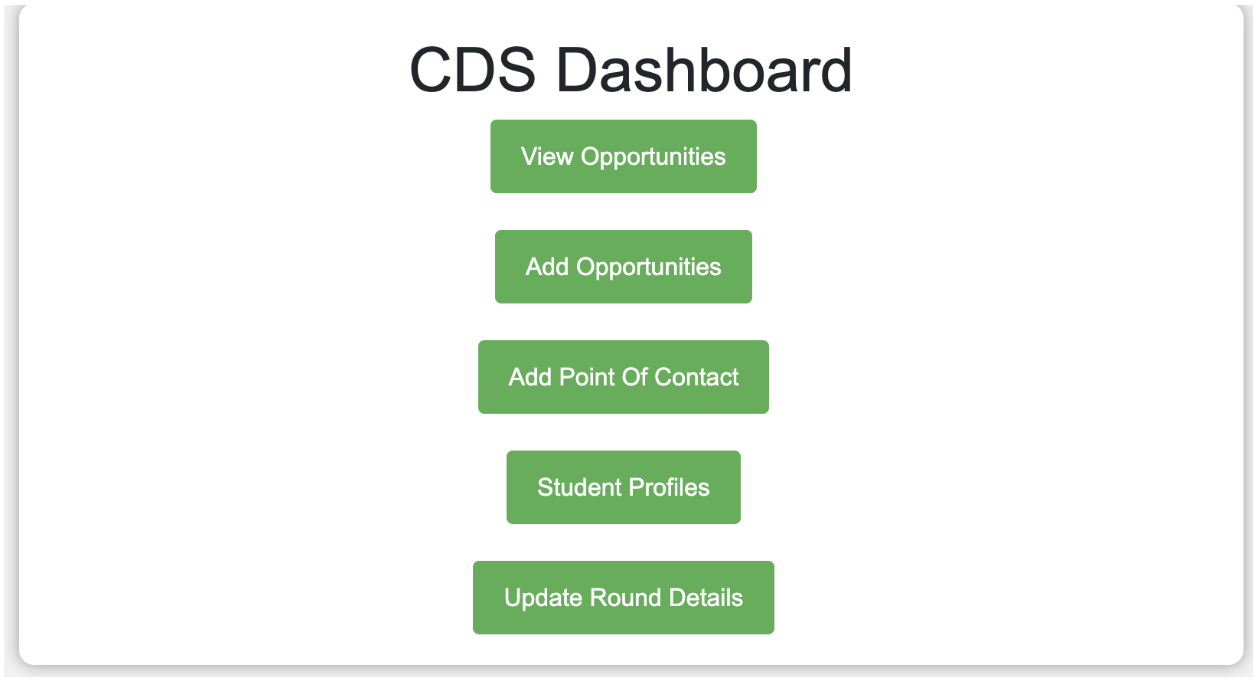
Solution:

- We have developed a system that allows students to easily check the details of each round and provides the CDS team with the flexibility to manage and update the details of each round quickly and efficiently. Students can view the round details for all opportunities they have applied to, which helps ensure a smooth and convenient experience. Our platform is designed to streamline the process for everyone involved, providing an efficient and user-friendly solution.

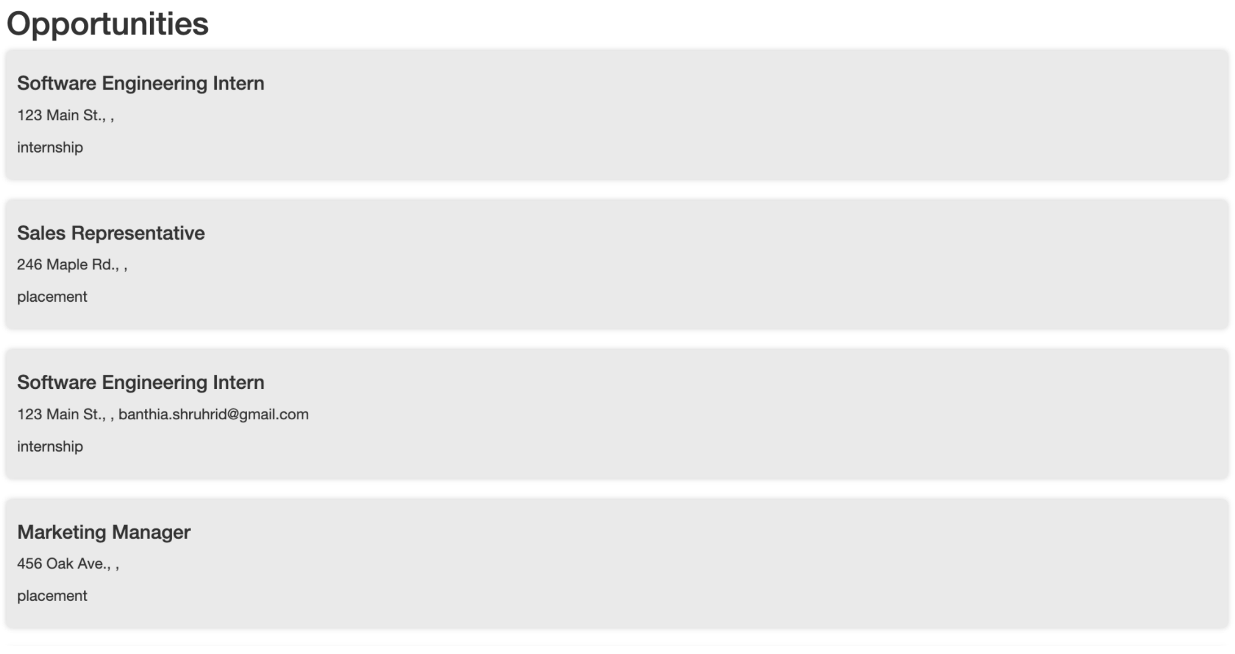
Before:



After:



New page to update round details has been added. When someone from the CDS team clicks on the Update Round Details Button, they can see the following view:


Opportunities

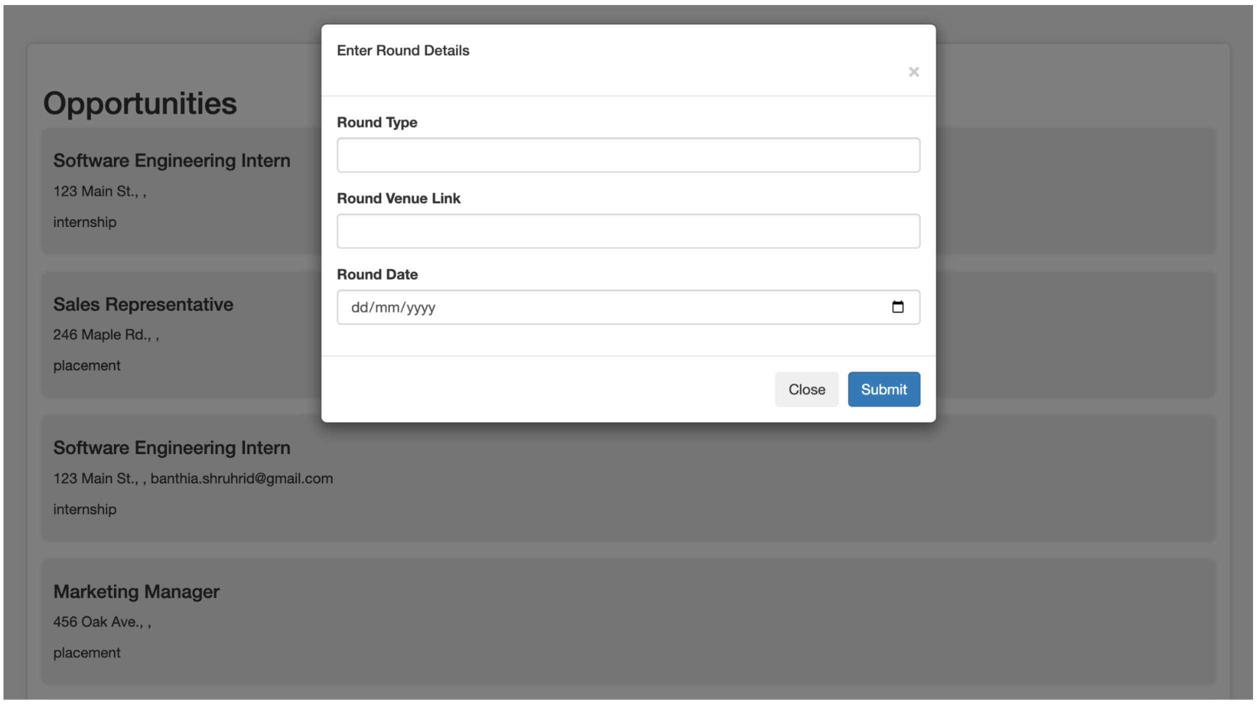
Software Engineering Intern
123 Main St., ,
internship

Sales Representative
246 Maple Rd., ,
placement

Software Engineering Intern
123 Main St., , banthia.shruhrid@gmail.com
internship

Marketing Manager
456 Oak Ave., ,
placement

Now, they can see all the opportunities that are currently floated and are active. Now, if they want to update the round details of a particular opportunity, they can click that particular opportunity card and the following view appears:



Enter Round Details

×

Round Type

Round Venue Link

Round Date

dd/mm/yyyy

Close Submit

Opportunities

Software Engineering Intern
123 Main St., ,
internship

Sales Representative
246 Maple Rd., ,
placement

Software Engineering Intern
123 Main St., , banthia.shruhrid@gmail.com
internship

Marketing Manager
456 Oak Ave., ,
placement

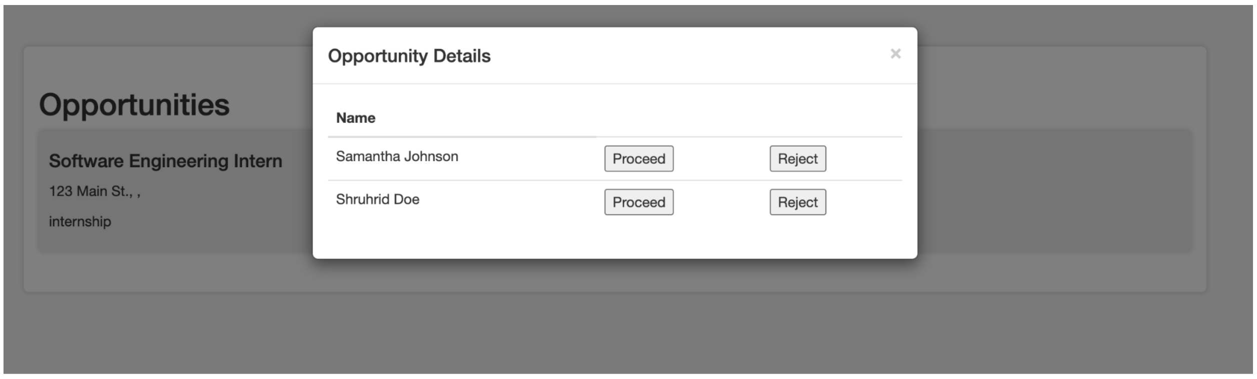
They can update the round details from this form that appears on the screen for any particular opportunity.

2. *The view where POC and select and reject students should be better with 3 options, "select", "reject" and "proceed". Currently, only 2 options are present "select" and "reject"*

Solution:

- We have enhanced the functionality of our web app by addressing an issue with the POC view for selecting and rejecting students. Previously, only two options were available, "select" and "reject." However, we have implemented a significant improvement by adding a third option, "proceed," which has greatly improved the user experience. This option proceeds the student into the next round and updates the round details for that particular student in their respective view. The point of contact can control the entire flow of the placement/internship selection process with little to no interference from the CDS team apart from certain round communications.

Before:



Opportunity Details

×

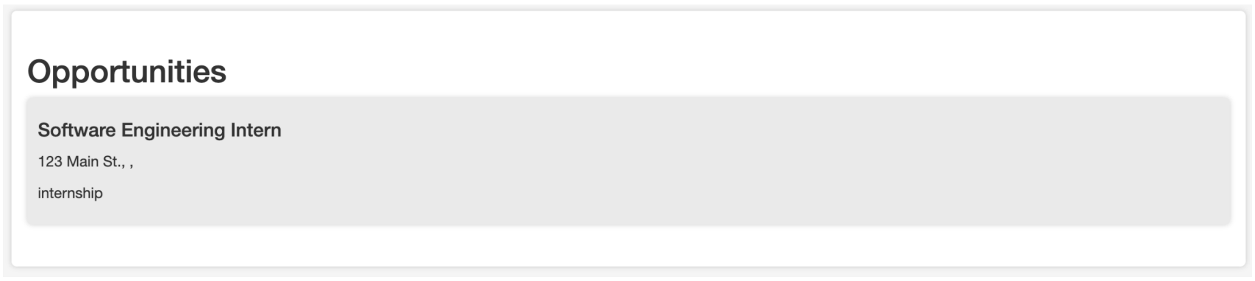
Name		
Samantha Johnson	Proceed	Reject
Shruhrid Doe	Proceed	Reject

Opportunities

Software Engineering Intern
123 Main St., ,
internship

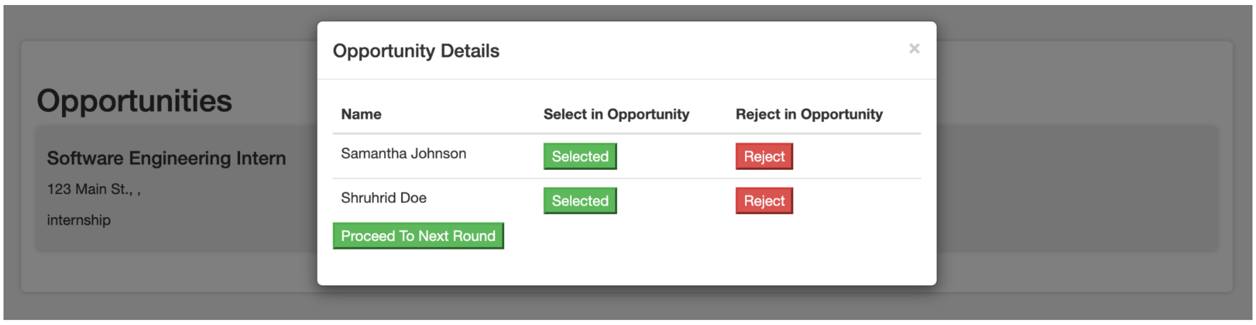
After:

The point of contact sees the opportunities floated by them as before:


Opportunities

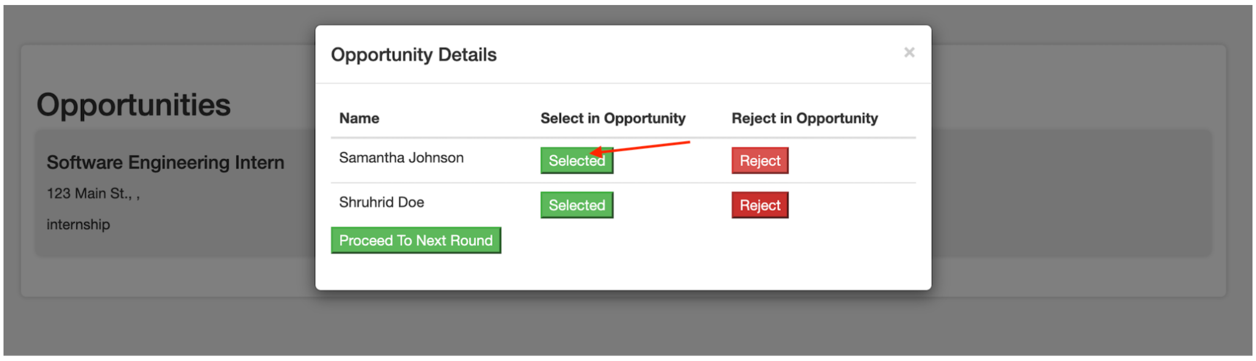
Software Engineering Intern
123 Main St., ,
internship

If they want to accept, reject or proceed a student from the current round to the next round, they can click on the particular opportunity for which they wish to do this for. After clicking on a particular opportunity card, they get the following view:


Opportunity Details

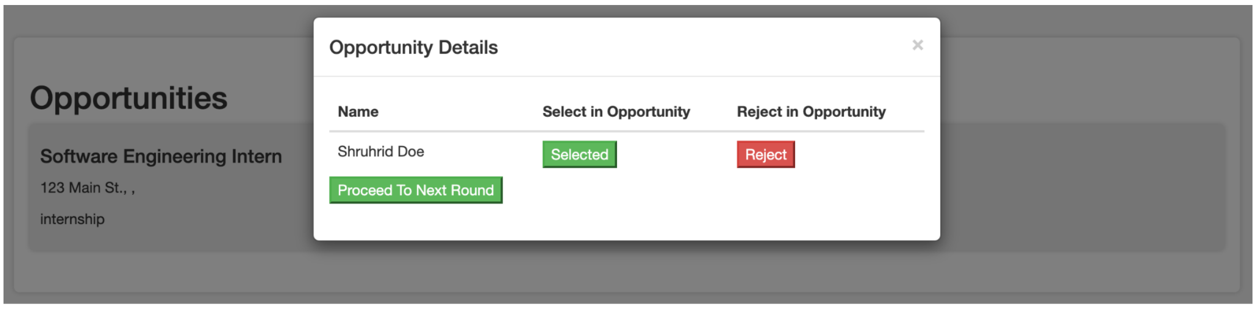
Name	Select in Opportunity	Reject in Opportunity
Samantha Johnson	Selected	Reject
Shruhrid Doe	Selected	Reject

Currently, we can see that 2 people are remaining in the current round of the process. There might be some students who have been accepted(selected in this opportunity after a lesser number of rounds) or rejected(rejected from this opportunity) at some previous stage. This view gives the point of contact the ability to make decisions on the students interviews using the options on the screen. The “Selected” button accepts them in this opportunity. The “Reject” button rejects them from this opportunity. The “Proceed to Next Round” button should be pressed after the point of contact has made the acceptance and rejection decisions for some of the students and wants the other students to go through another round of interviews or tests. After pressing this button, someone from the CDS team can use the update round details feature to add new round details for the students that have proceeded to the further rounds.
If we click on this button:


Opportunity Details

Name	Select in Opportunity	Reject in Opportunity
Samantha Johnson	Selected	Reject
Shruhrid Doe	Selected	Reject

We can see that “Samantha Johnson” does not appear in the current rounds anymore (because she has been accepted in this opportunity):


Opportunity Details

Name	Select in Opportunity	Reject in Opportunity
Shruhrid Doe	Selected	Reject

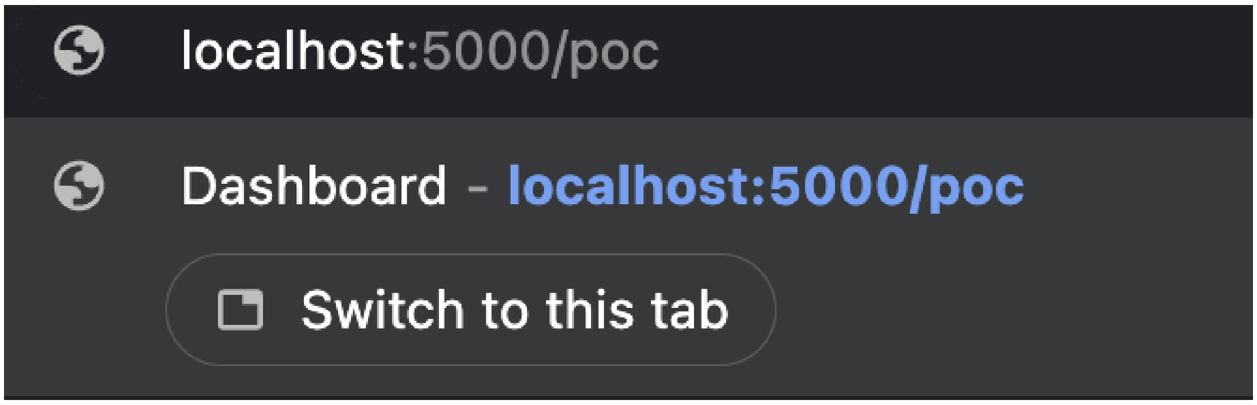
Similar case would be if we press the “Reject” button.

3. *A common login page would be nice.*

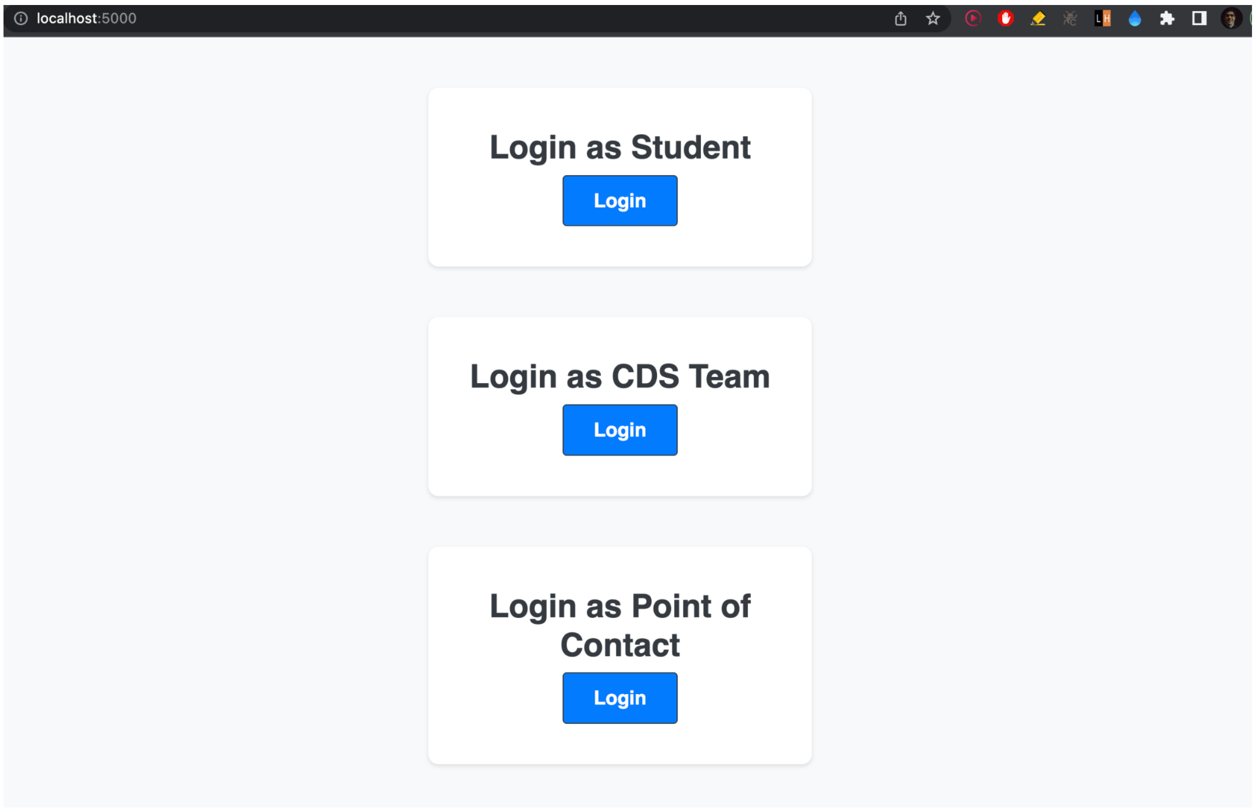
Solution:

- We have made a common login page which prompts the user the option to log in as the following:
    - Student
    - CDS Team
    - Point of Contact
- This removes the initial roadblock in which we had to make sure to use the correct links for the correct user. Now after the login page has been accessed, corresponding redirection has been made so that the user would be able to access their pages without any roadblocks.

Before:



We had to manually choose the view



After:

The default page is the login page now. Each button is configured to communicate with the backend server using a post request which tells the server which user is trying to login. The server then redirects the user to their corresponding dashboard. The authentication process used is Google Authentication.

Changes Made After Second Feedback:

In the second round of feedback, the stakeholders expressed their admiration for our web application. We proactively discussed potential security concerns with our database and assured them that we would take necessary measures to address them. Following the meeting, we promptly resolved all identified security issues to ensure the safety and confidentiality of our data.

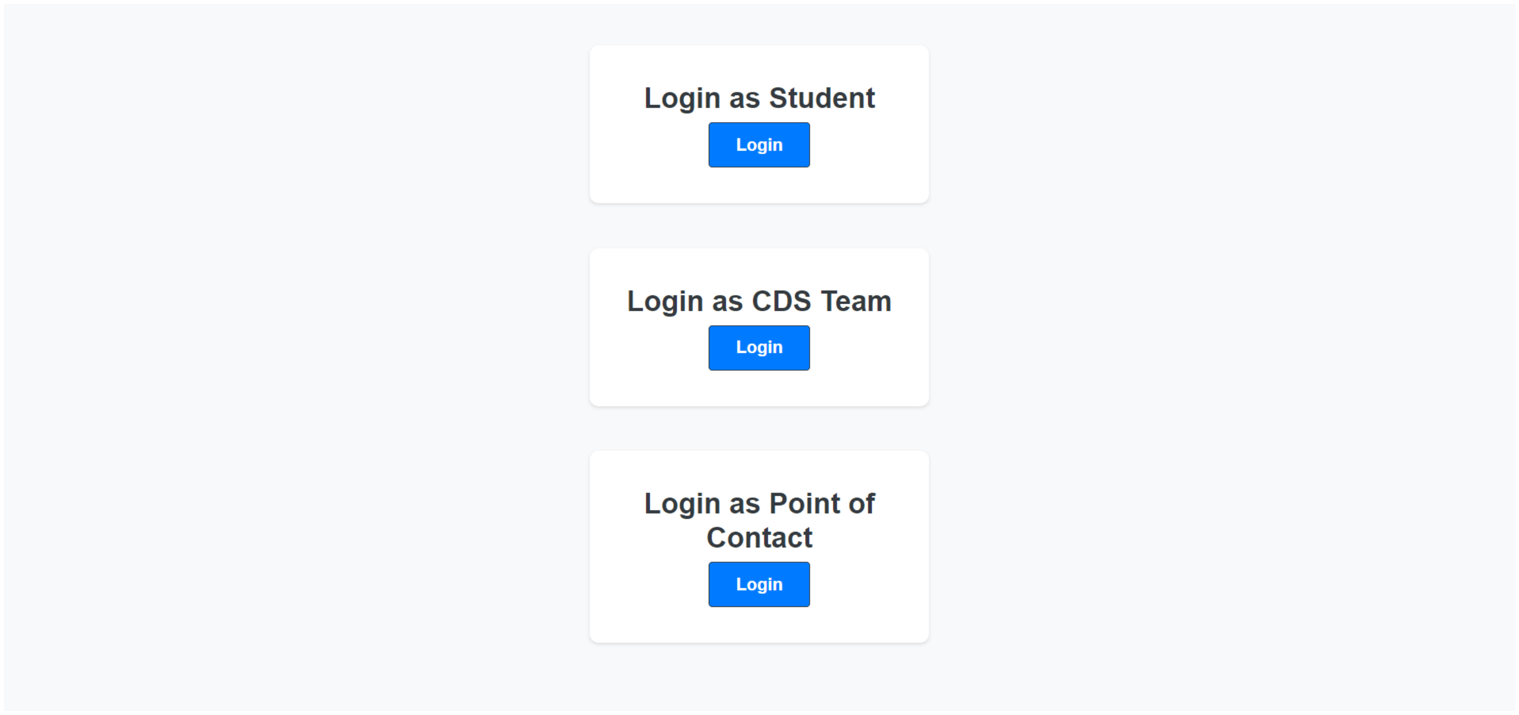
The web app had vulnerabilities to SQL injection and Cross-Site Scripting (XSS) attacks. An attacker could exploit the SQL injection vulnerability in the upload_resume function of the Python backend by inserting malicious SQL statements into the resume_file_name or resume_file fields. This would allow them to view, modify, or delete data from the application's database. The attacker could also exploit the XSS vulnerability by injecting malicious scripts into the resume_file_name or resume_file fields, which could steal sensitive information or perform actions on the user's behalf. The defense against SQL injection attacks is to use prepared statements or parameterized queries instead of dynamically generating SQL queries. Additionally, input validation and sanitization can also help prevent SQL injection attacks. To prevent XSS attacks, input fields should be sanitized using a library like bleach to remove any characters that could be used in XSS attacks.

Note: All the images regarding this are included in the portion of G1 and G2.

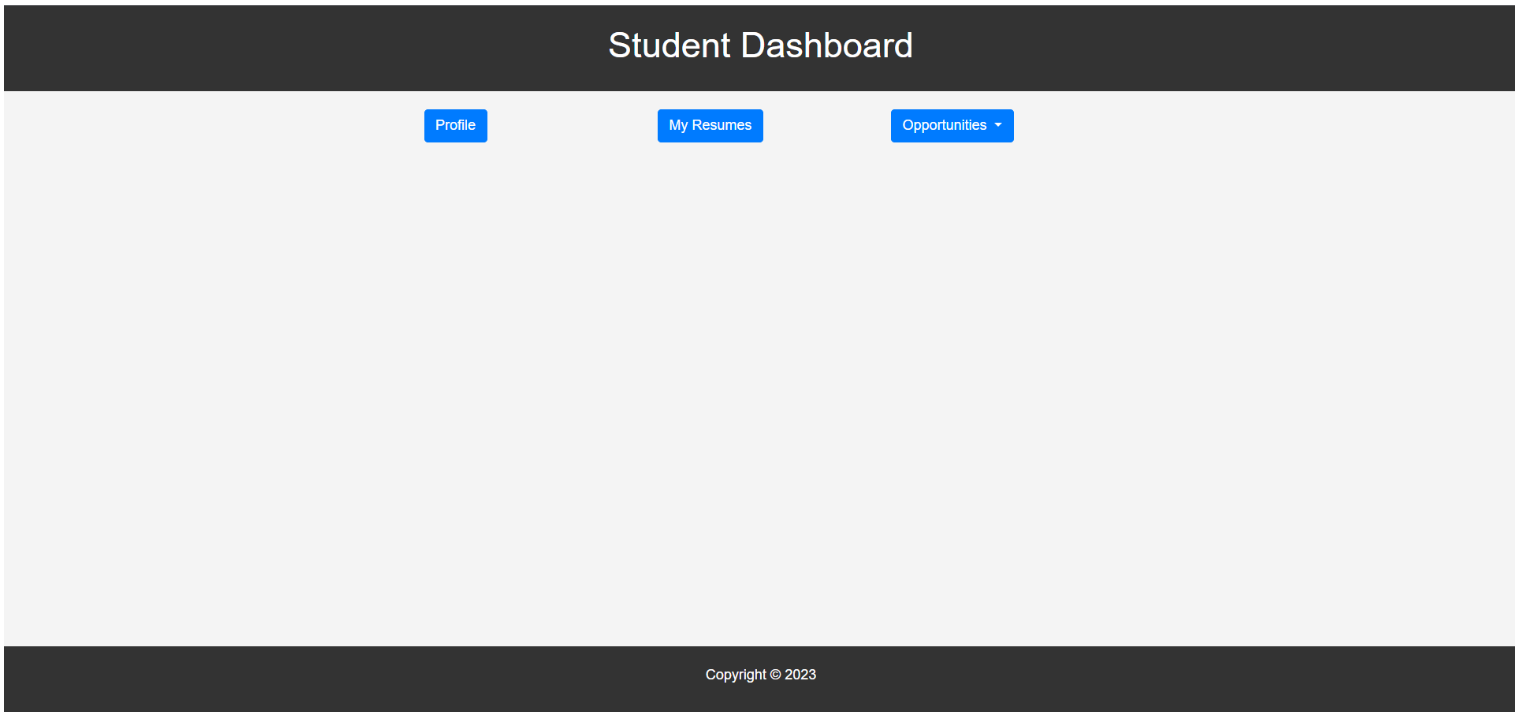
2. Attach screenshots of different views [along with a write-up on their privileges] of the database as seen by different classes of users.

Common View:

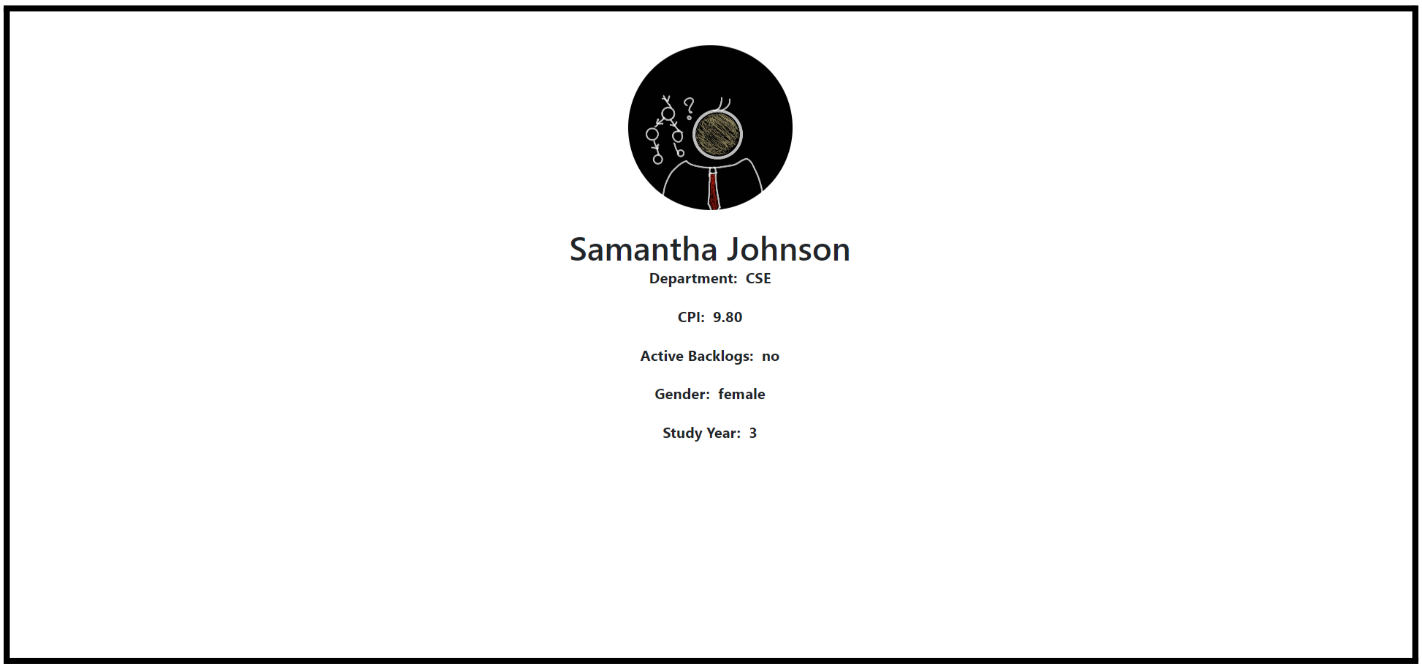
Students, CDS as well as point of contact can see this common page where they can login.



Student View:



Student has the access to view their profile, view the opportunities and view and update resume.


Samantha Johnson

Department: CSE

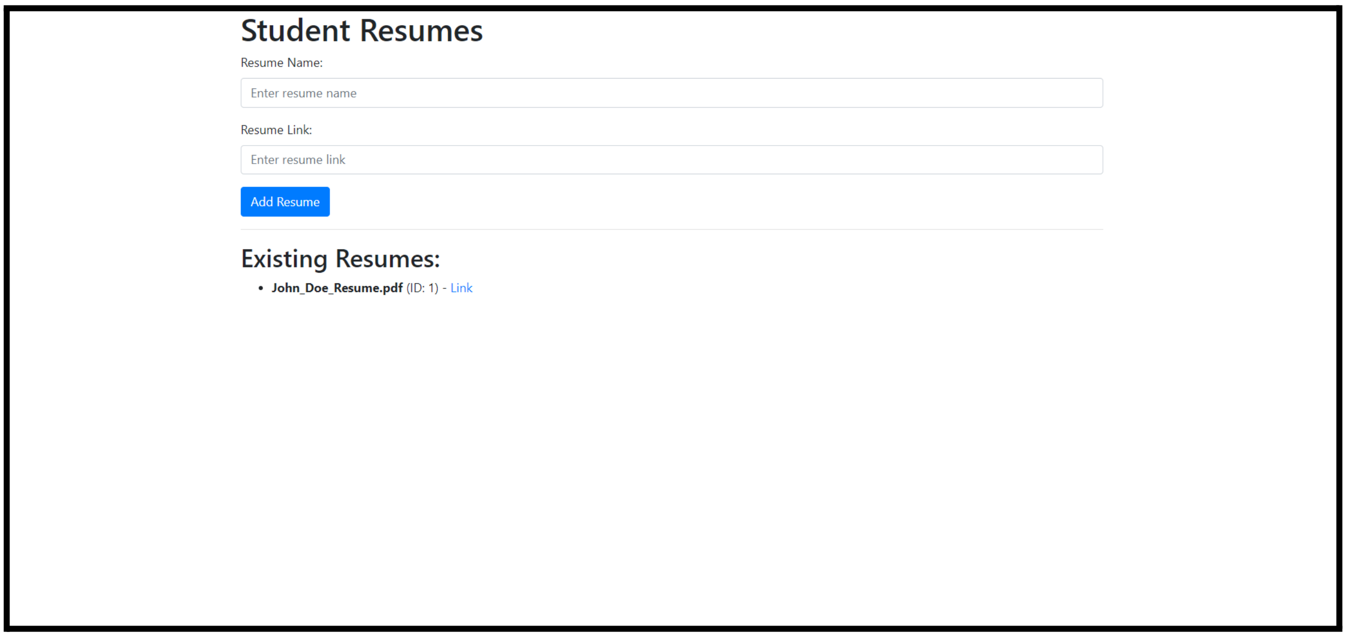
CPI: 9.80

Active Backlogs: no

Gender: female

Study Year: 3

Student only has the privilege to view the profile and not edit any component.


Student Resumes

Resume Name:

Enter resume name

Resume Link:

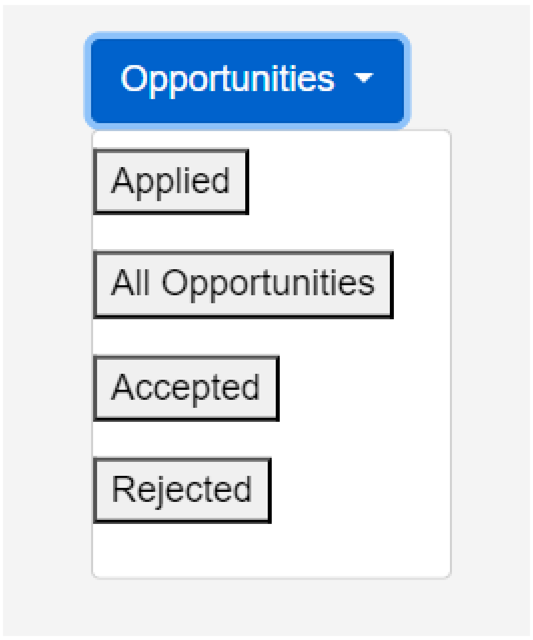
Enter resume link

Add Resume

Existing Resumes:

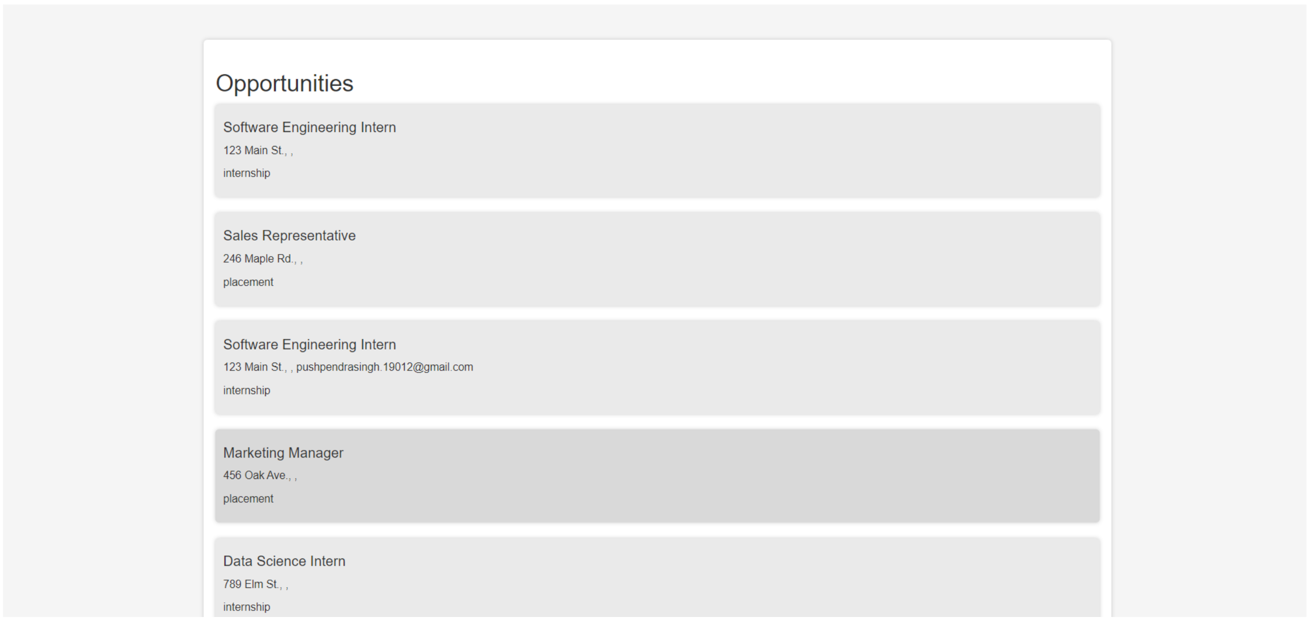
- **John_Doe_Resume.pdf** (ID: 1) - Link

Student has the privilege to view existing resumes and they can also add new resume for various other opportunities.



Student has the privilege to view opportunities classified under these four categories.

- In the *All Opportunites*, student can view all the opportunities for which they are eligible to.


Opportunities

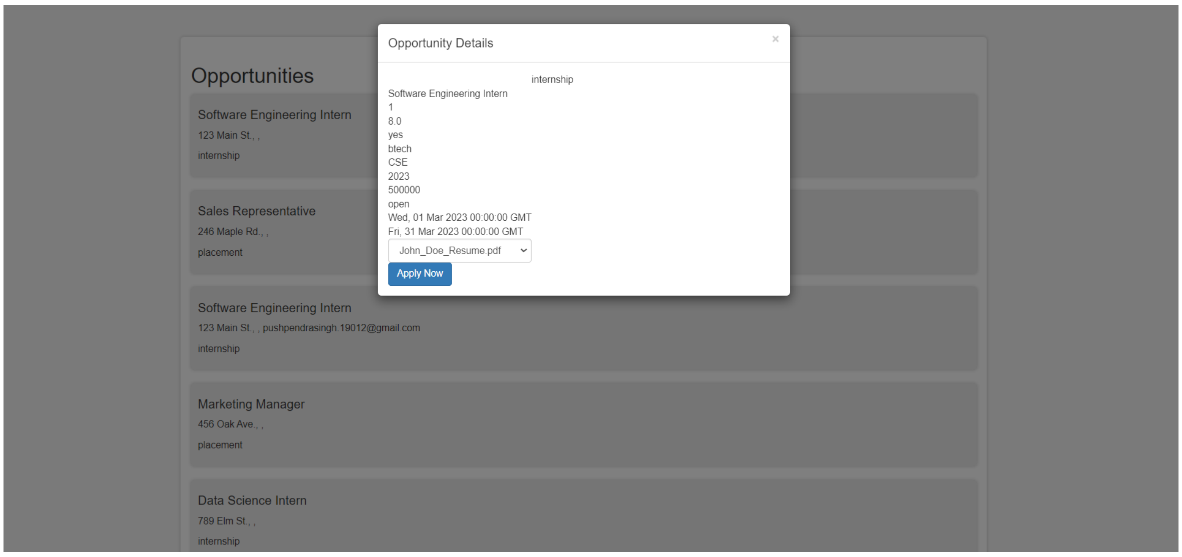
Software Engineering Intern
123 Main St., ,
internship

Sales Representative
246 Maple Rd., ,
placement

Software Engineering Intern
123 Main St., , pushpendrasingh.19012@gmail.com
internship

Marketing Manager
456 Oak Ave., ,
placement

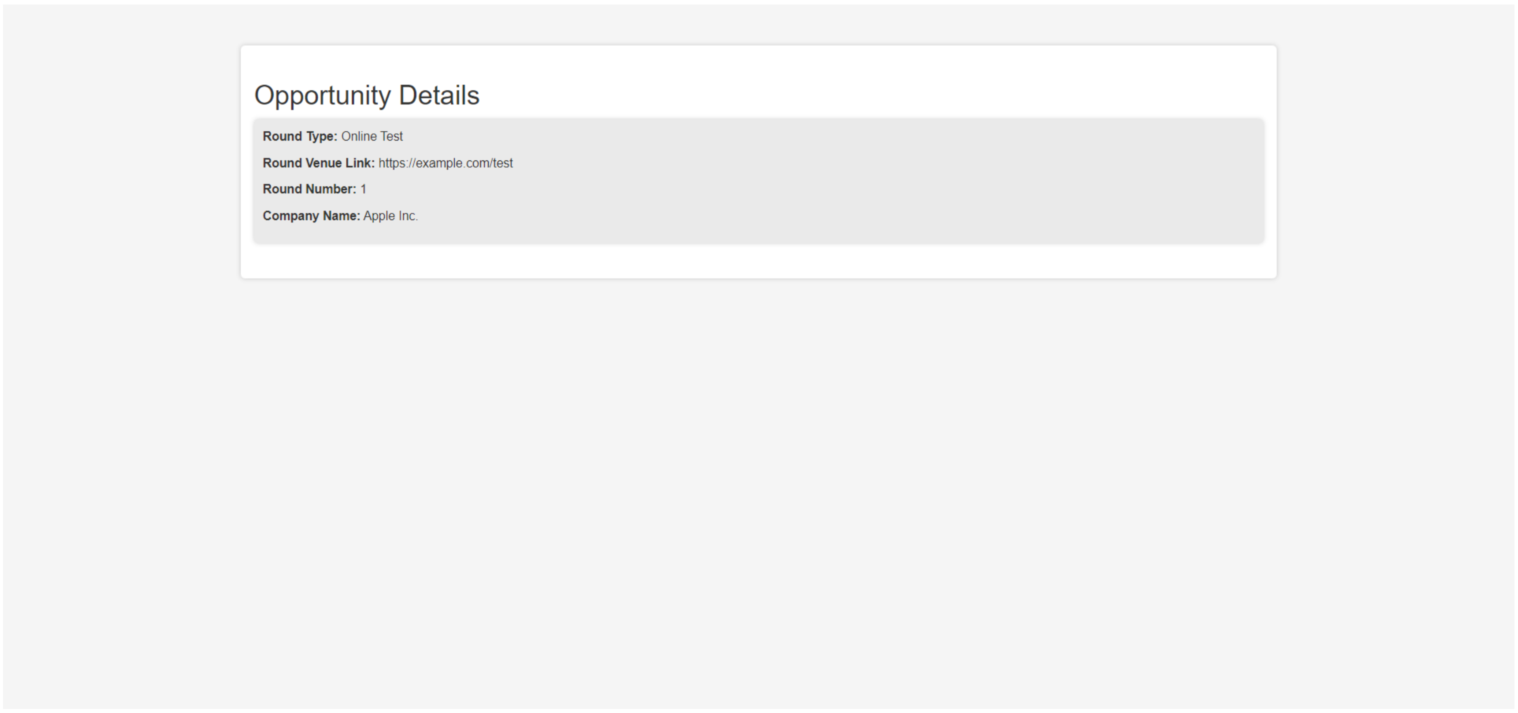
Data Science Intern
789 Elm St., ,
internship


Opportunity Details

internship
Software Engineering Intern
1
8.0
yes
btech
CSE
2023
500000
open
Wed, 01 Mar 2023 00:00:00 GMT
Fri, 31 Mar 2023 00:00:00 GMT
John_Doe_Resume.pdf

By clicking on any opportunity, the student can then view the detailed information of the opportunity and can select any one resume from the existing resumes and apply for the opportunity. The student only has the privilege to view and add resumes in this section.

- In the *Applied*, student can view all the opportunities for which they have applied to.



Opportunity Details

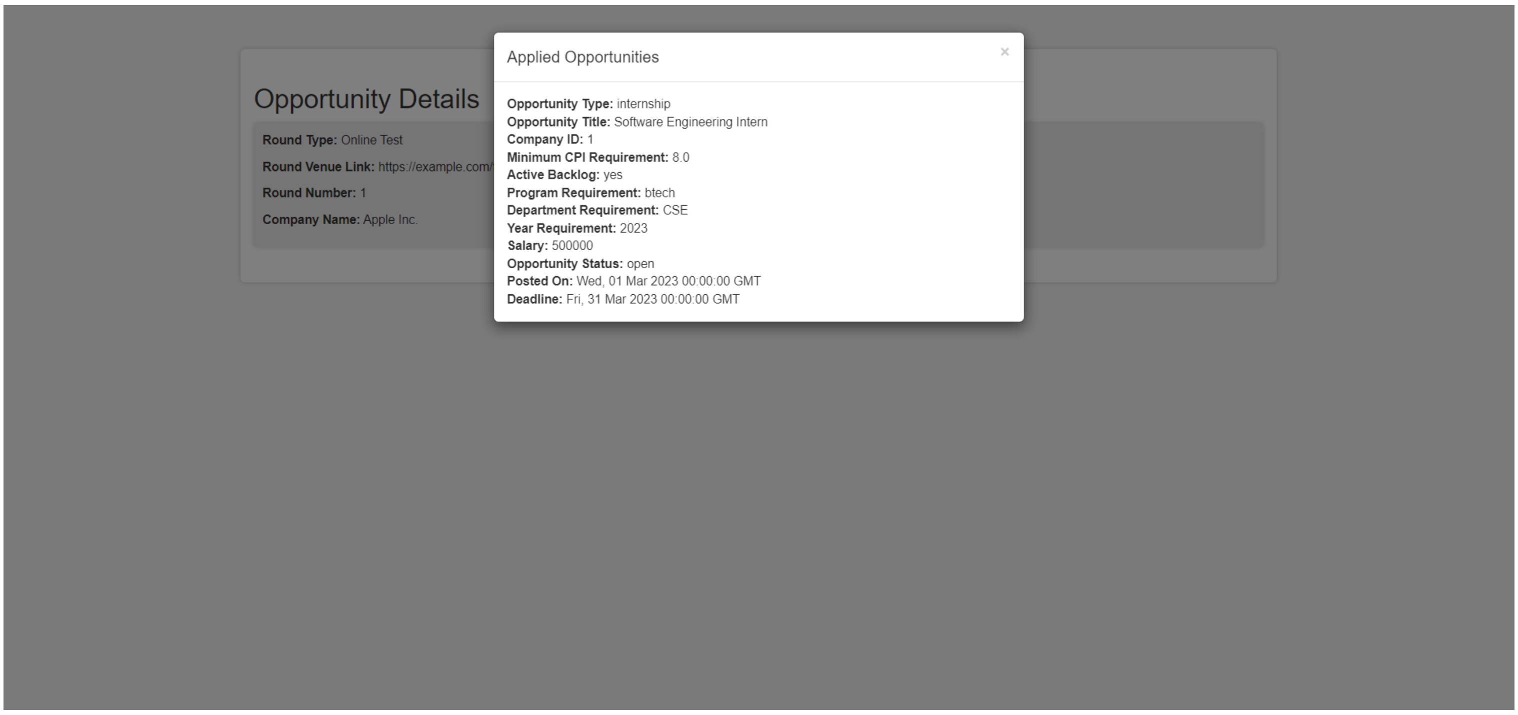
Round Type: Online Test

Round Venue Link: https://example.com/test

Round Number: 1

Company Name: Apple Inc.

- In the *Accepted* and *Rejected*, student can view all the opportunities for which they have been accepted or rejected respectively.

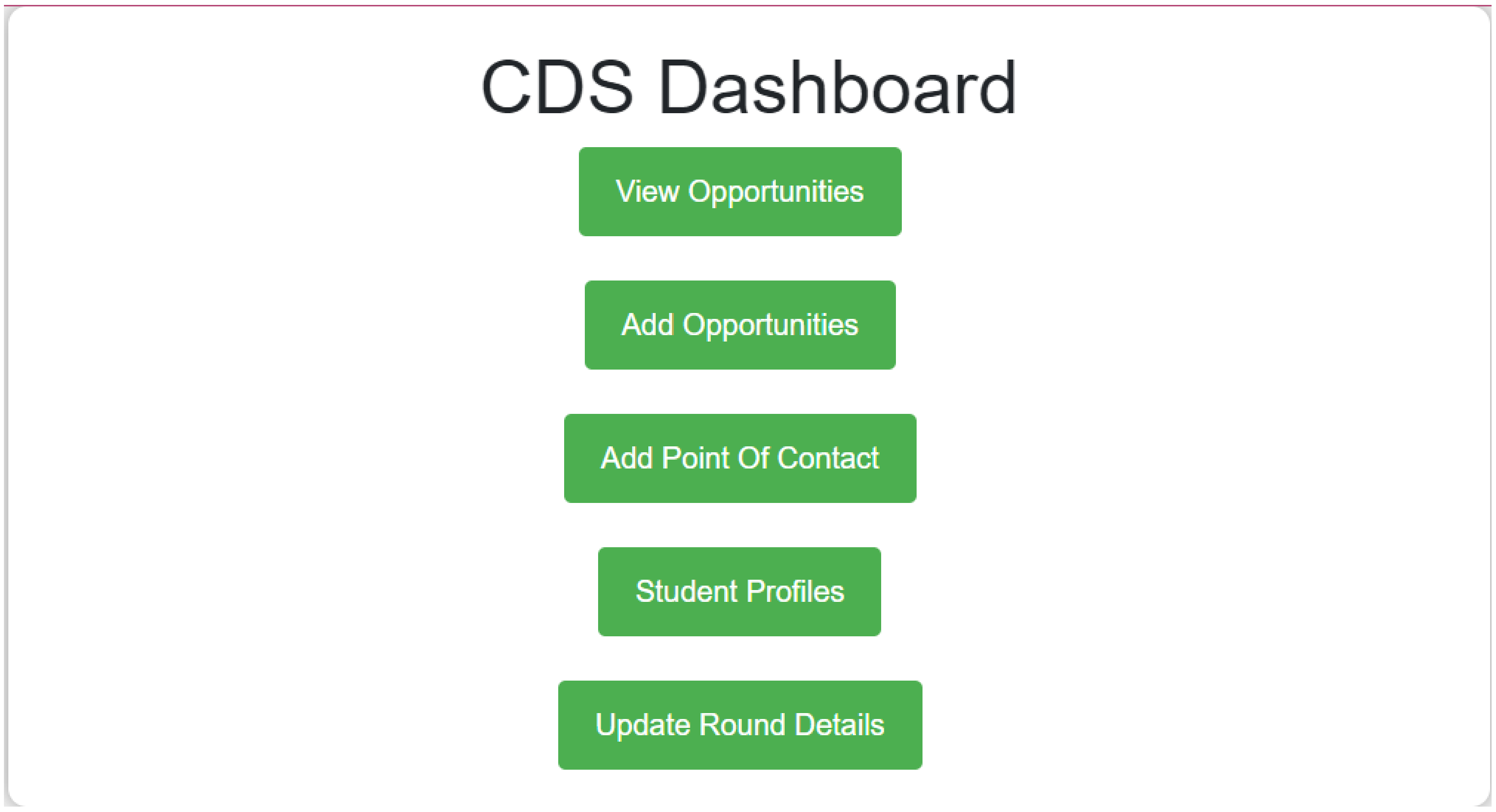


Applied Opportunities

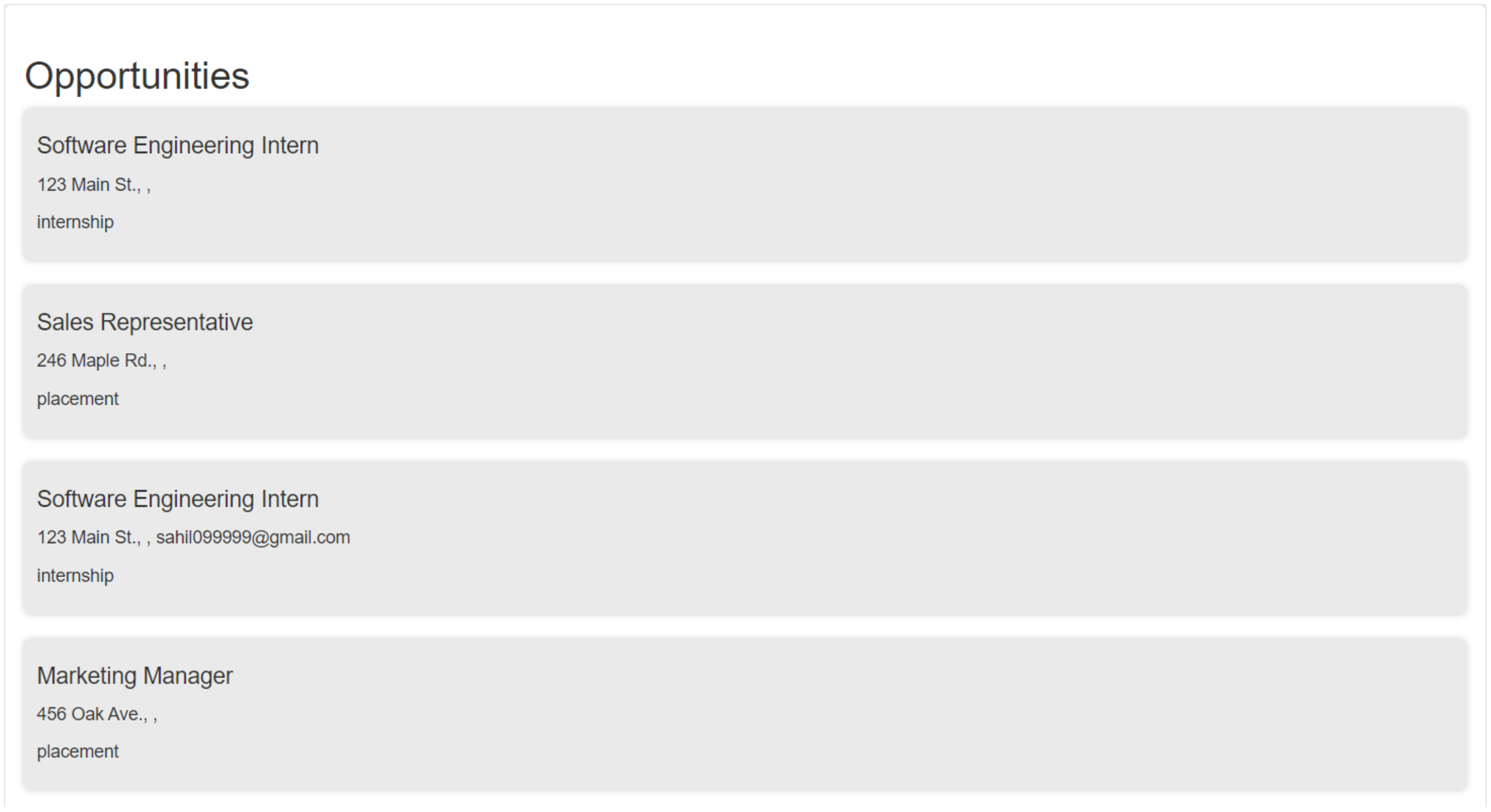
Opportunity Type: internship
Opportunity Title: Software Engineering Intern
Company ID: 1
Minimum CPI Requirement: 8.0
Active Backlog: yes
Program Requirement: btech
Department Requirement: CSE
Year Requirement: 2023
Salary: 500000
Opportunity Status: open
Posted On: Wed, 01 Mar 2023 00:00:00 GMT
Deadline: Fri, 31 Mar 2023 00:00:00 GMT

In both these pages, student only has the privilege to view opportunities and information.

CDS View:
CDS have access to view, add and delete opportunities, add point of contact, edit student profiles, and update the round details.



CDS employee can view as well as delete opportunities.

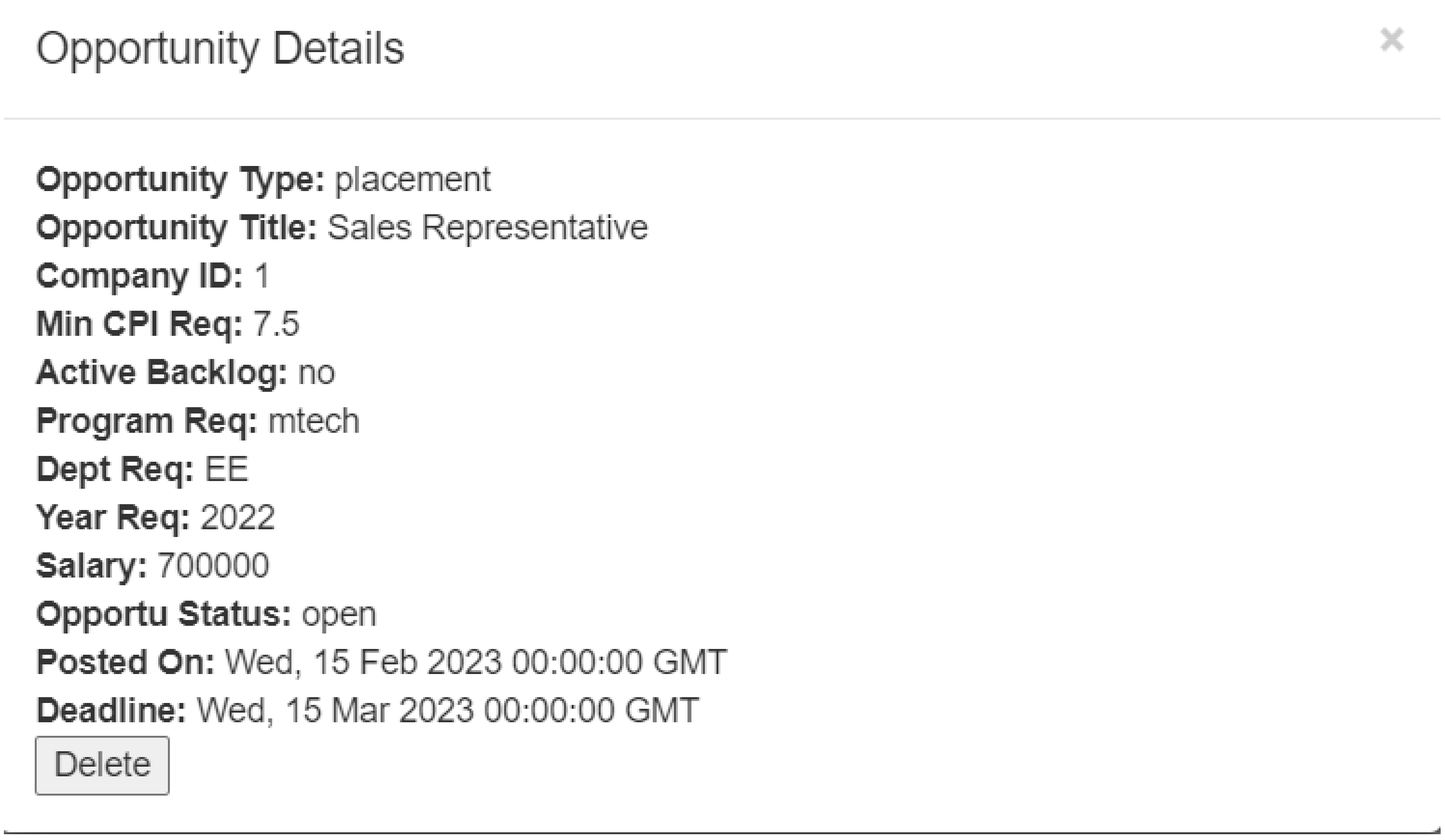

Opportunities

Software Engineering Intern
123 Main St., ,
internship

Sales Representative
246 Maple Rd., ,
placement

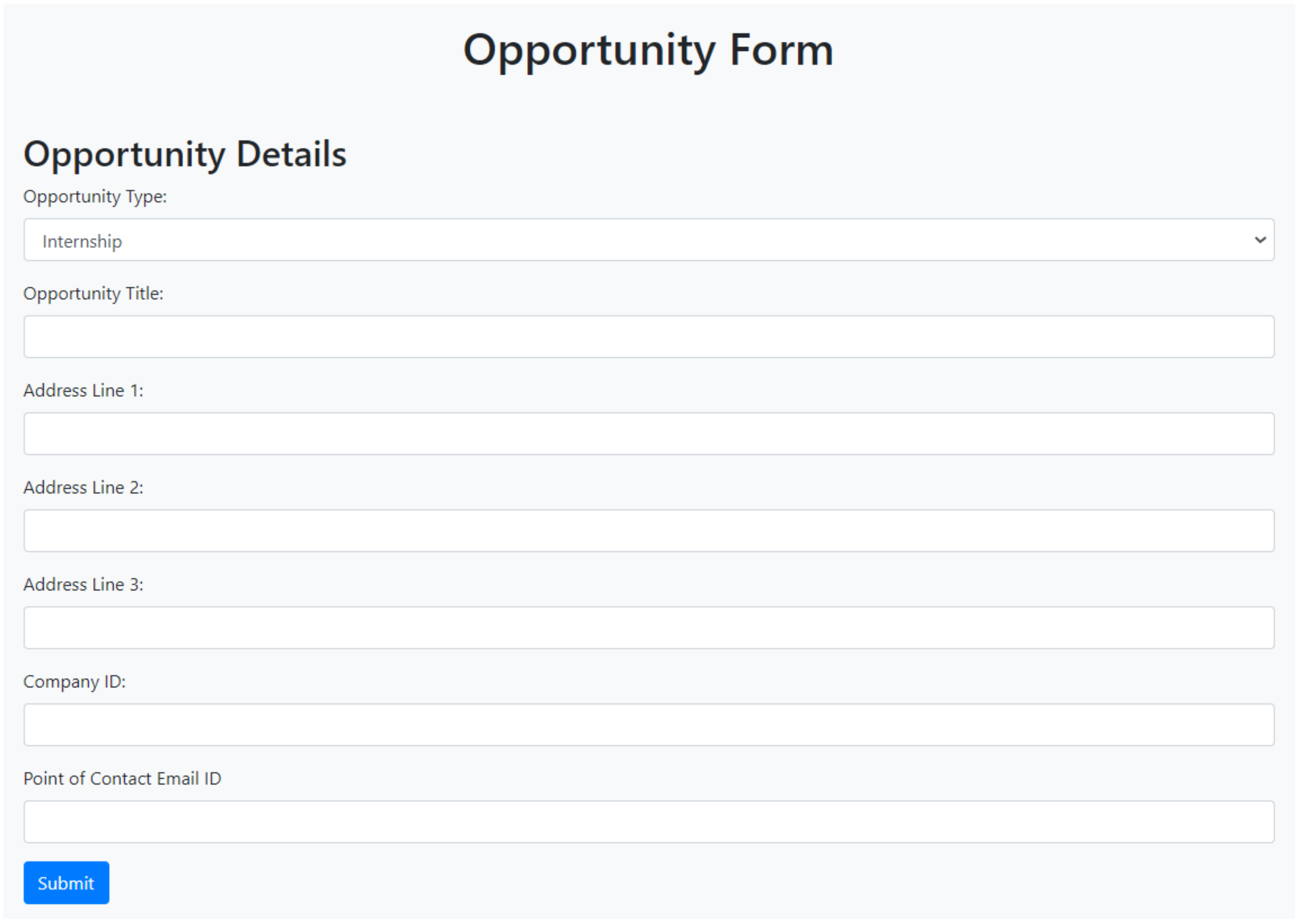
Software Engineering Intern
123 Main St., , sahil099999@gmail.com
internship

Marketing Manager
456 Oak Ave., ,
placement


Opportunity Details

Opportunity Type: placement
Opportunity Title: Sales Representative
Company ID: 1
Min CPI Req: 7.5
Active Backlog: no
Program Req: mtech
Dept Req: EE
Year Req: 2022
Salary: 700000
Opportu Status: open
Posted On: Wed, 15 Feb 2023 00:00:00 GMT
Deadline: Wed, 15 Mar 2023 00:00:00 GMT

CDS can add a opportunity by filling up the opportunity form and the requirements form.


Opportunity Form

Opportunity Details

Opportunity Type: Internship

Opportunity Title:

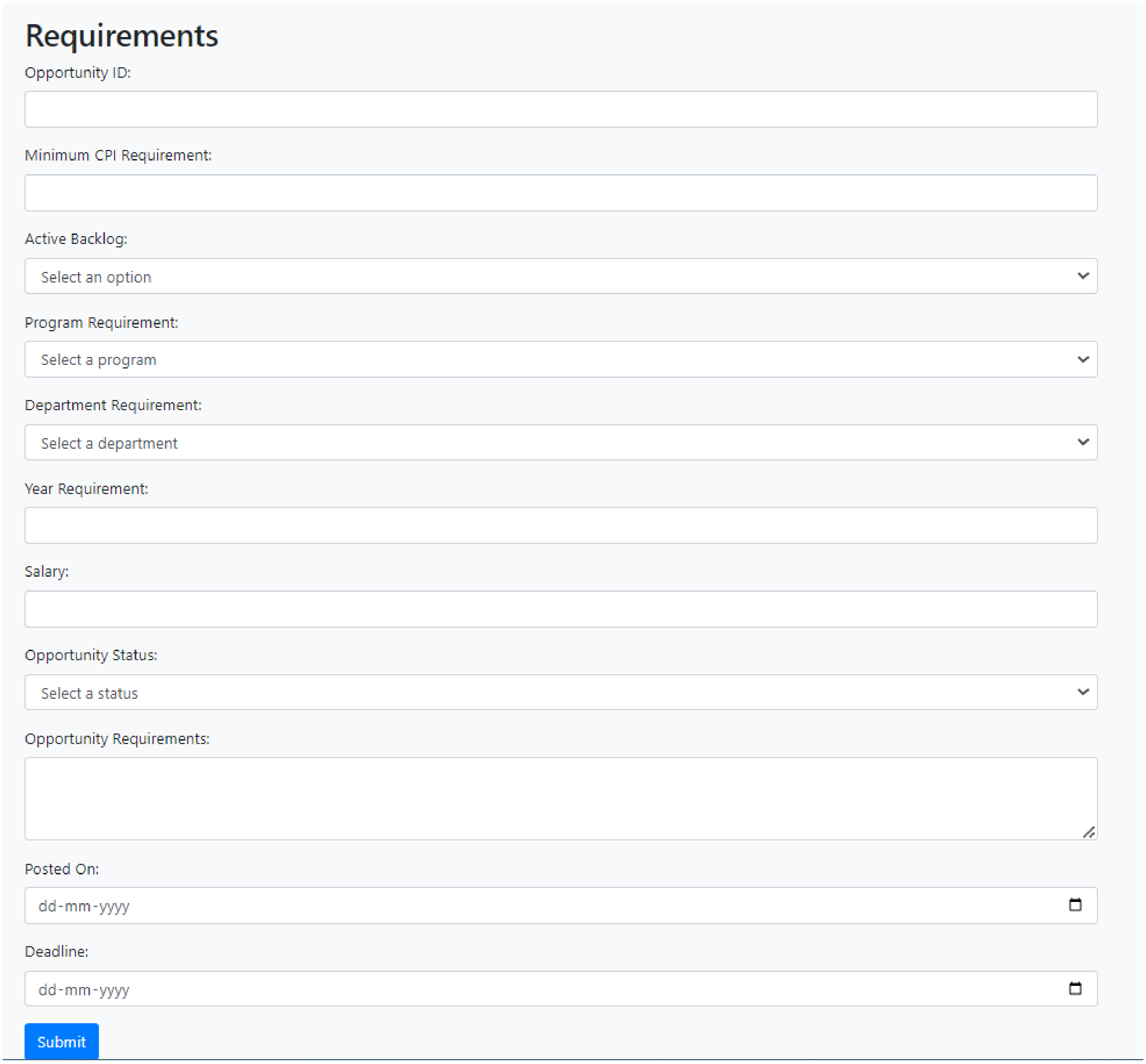
Address Line 1:

Address Line 2:

Address Line 3:

Company ID:

Point of Contact Email ID



Requirements

Opportunity ID:

Minimum CPI Requirement:

Active Backlog:

Select an option

Program Requirement:

Select a program

Department Requirement:

Select a department

Year Requirement:

Salary:

Opportunity Status:

Select a status

Opportunity Requirements:

Posted On:

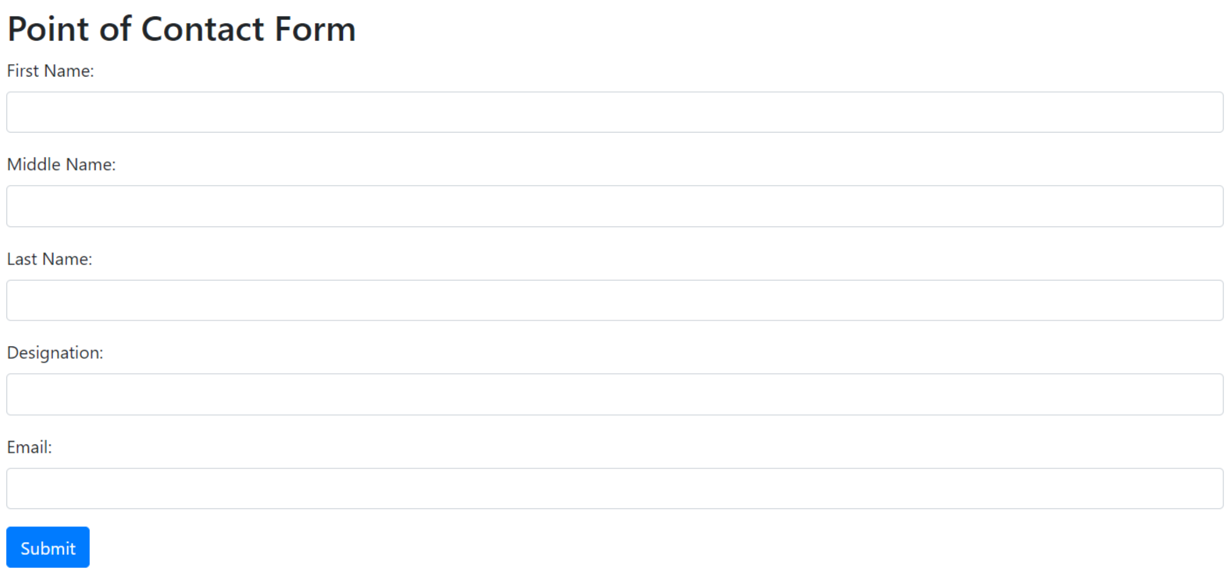
dd-mm-yyyy

Deadline:

dd-mm-yyyy

Submit

CDS can add points of contact and their details.



Point of Contact Form

First Name:

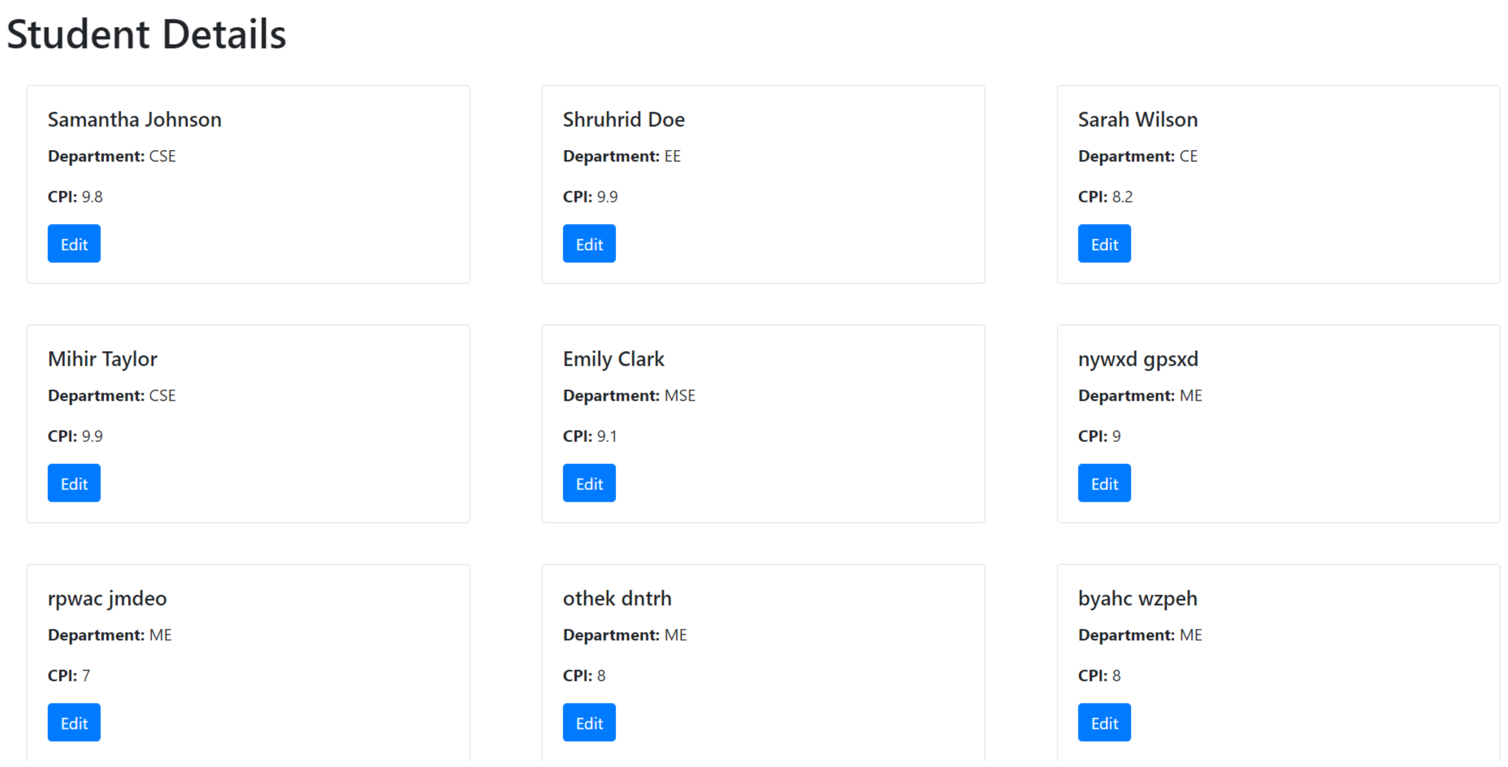
Middle Name:

Last Name:

Designation:

Email:

CDS have access to the profiles of each student and they can edit it as well.



Student Details

Samantha Johnson
Department: CSE
CPI: 9.8

Shruhrid Doe
Department: EE
CPI: 9.9

Sarah Wilson
Department: CE
CPI: 8.2

Mihir Taylor
Department: CSE
CPI: 9.9

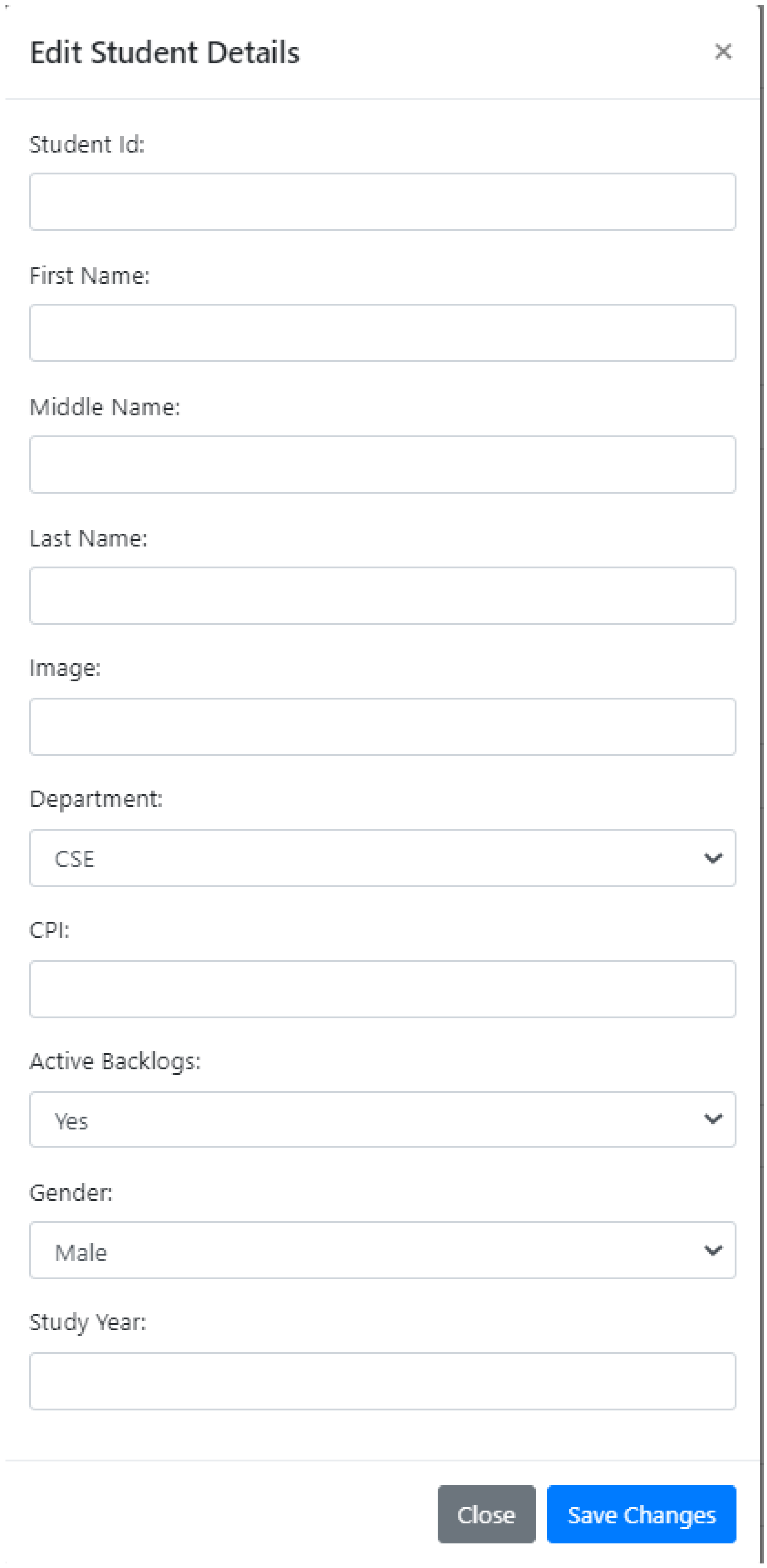
Emily Clark
Department: MSE
CPI: 9.1

nywxd gpsxd
Department: ME
CPI: 9

rpwac jmdeo
Department: ME
CPI: 7

othek dntrh
Department: ME
CPI: 8

byahc wzpeh
Department: ME
CPI: 8


Edit Student Details

Student Id:

First Name:

Middle Name:

Last Name:

Image:

Department: CSE

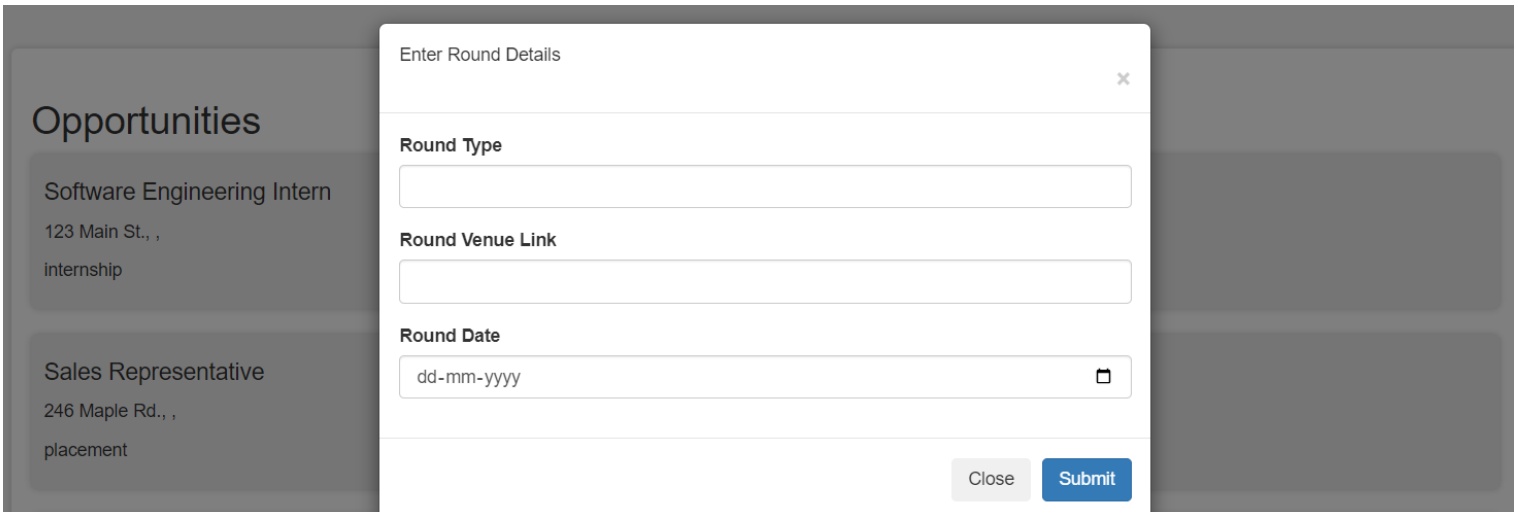
CPI:

Active Backlogs: Yes

Gender: Male

Study Year:

CDS can also update the round details for a particular opportunity.


Opportunities

Software Engineering Intern

123 Main St., ,

internship

Sales Representative

246 Maple Rd., ,

placement

Enter Round Details

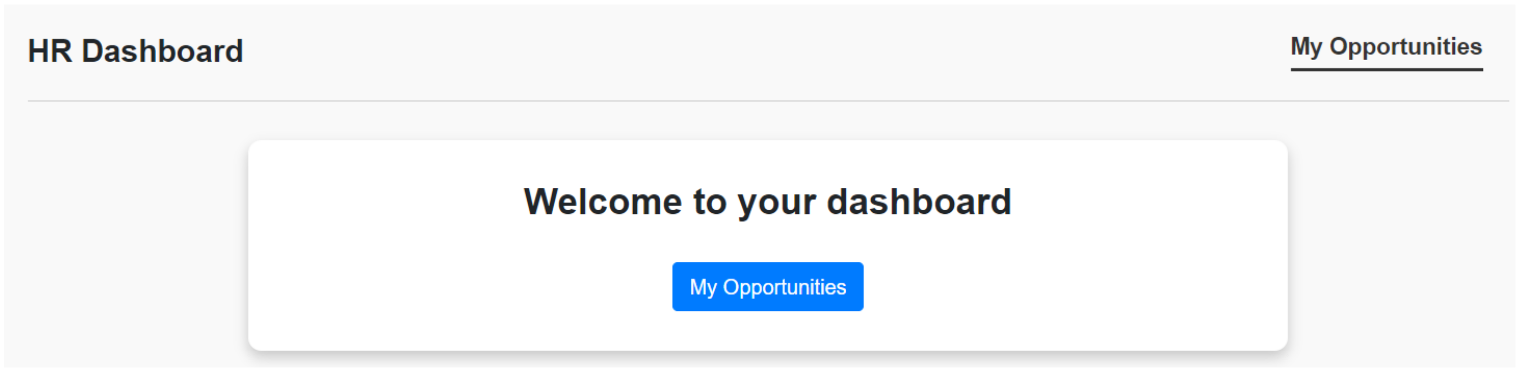
Round Type

Round Venue Link

Round Date

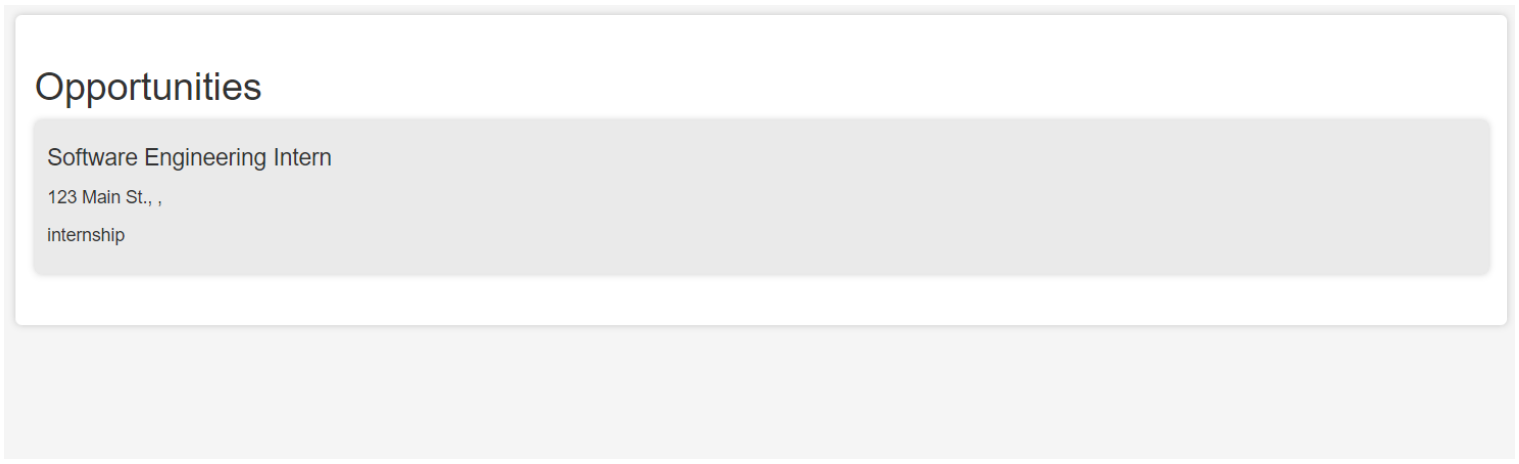
dd-mm-yyyy

POC View:


HR Dashboard

Welcome to your dashboard

By clicking on the My Opportunities button, we get the list of opportunities under that particular POC.

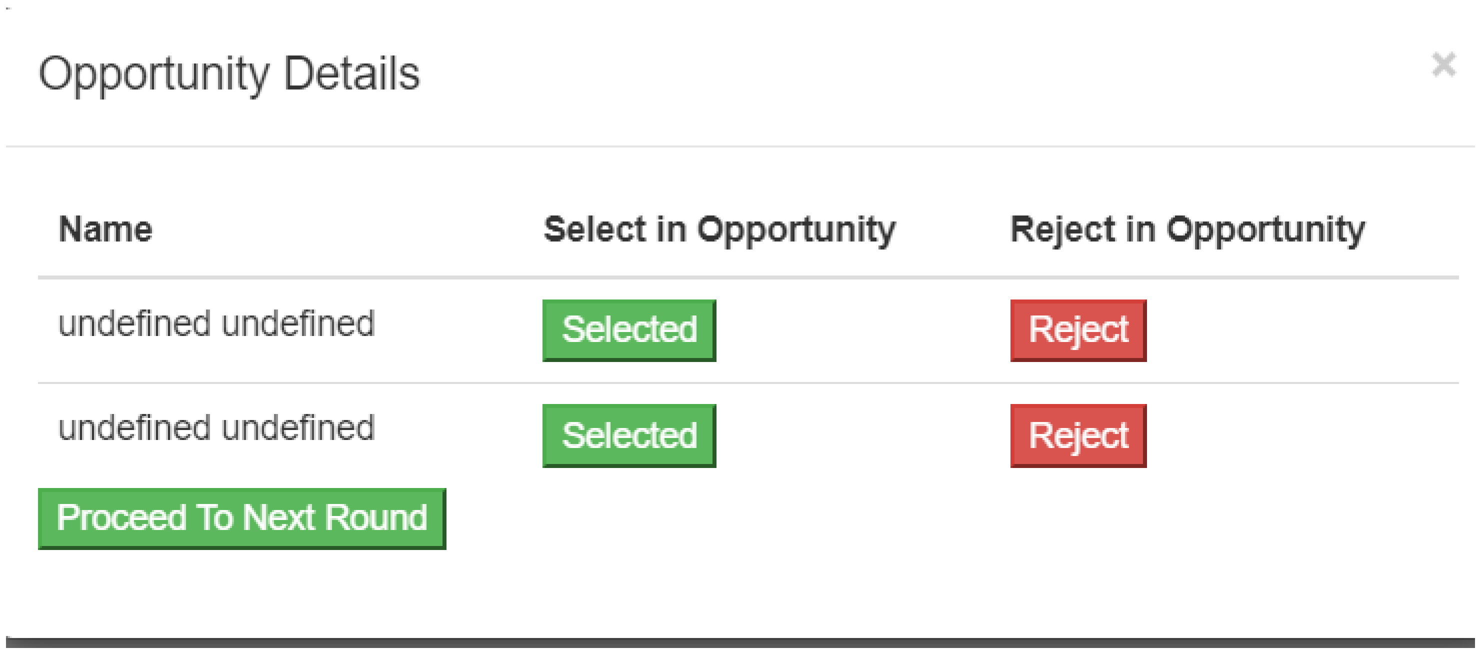

Opportunities

Software Engineering Intern

123 Main St., ,

internship

The POC can select / reject a student for a particular opportunity, and can also proceed him/her to the next round.



Opportunity Details

Name	Select in Opportunity	Reject in Opportunity
undefined undefined	Selected	Reject
undefined undefined	Selected	Reject

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.

Locks need to be implemented to store data into stable storage before any changes are made to the database on disk. A query starts in an active state. In case a query is executed but not committed, the query fails and gets aborted. This leads to the consistency property.

We have used locks in POST requests for:

1. cds/opportunity
2. cds/requirements
3. cds/opportunity_delete
4. student/apply/
5. poc/opportunity
6. cds/poc_add
7. student/resume
8. student/image
9. poc/opportunity/student
10. cds/student
11. cds/update_round_details

We have added locks in all POST requests which have INSERT, UPDATE and DELETE.
The FOR UPDATE clause is used to lock the selected rows in a table for exclusive access by the current transaction.

We also used the try-except block to handle any errors that may occur while executing SQL queries and rollback the transaction in case of an exception.

If any exception is raised during the execution of SQL queries, the except block will be executed, and the transaction will be rolled back to its previous state.

For the case of insertion, there might be a case when multiple users add the same set of rows. Therefore, we check if the row exists, and if it does, we return a 409 Conflict response. If it doesn't exist, we insert a new row and return a 200 OK response. Finally, we commit the transaction, close the cursor, and handle any exceptions that may occur while executing SQL queries.

For the case of update image, we don't need to use try-except or rollback because the update statement will automatically release the lock when the transaction is committed.

Note that it's important to use placeholders in the SQL statement to prevent SQL injection attacks.

Table level lock vs row level lock:
The appropriateness of using a row-level lock versus a table-level lock depends on the specifics of the database system and the application requirements. Locking the whole table might be an overkill for just adding a single row. In general, it's best to use the smallest amount of locking necessary to ensure data consistency and integrity. A row-level lock will only prevent other transactions from modifying the same

row that is being inserted or updated, instead of the entire table. This can reduce contention and improve performance.

Table level locks can be used in cases where the entire table needs to be locked, for example, during a large batch operation that affects the entire table. This can prevent other transactions from modifying the table while the operation is being performed, which can be useful in situations where data consistency is critical.

Table level locks can also be used when there is a need to perform maintenance operations on the table, such as rebuilding indexes or updating statistics. In such cases, a table level lock can prevent other transactions from accessing the table and interfering with the maintenance operation.

Adding row level lock:
A row-level lock can be applied by adding the "FOR UPDATE" clause at the end of the SELECT statement.

Adding table level lock:
To use table level lock, LOCK TABLES and UNLOCK TABLES statements are used.
Syntax:
LOCK TABLES table_name [AS alias] lock_type [, table_name [AS alias] lock_type] ...
where, alias is an optional table alias and lock_type is the type of lock to apply, which can be READ, WRITE, or LOW_PRIORITY_WRITE

E.g.,
LOCK TABLES my_table WRITE;

To release the lock:

UNLOCK TABLES;

Note: All the changes in the codebase for this question can be seen in app.py file upload in github.
Gihub Link: https://github.com/sanskarfc/placement_system_2

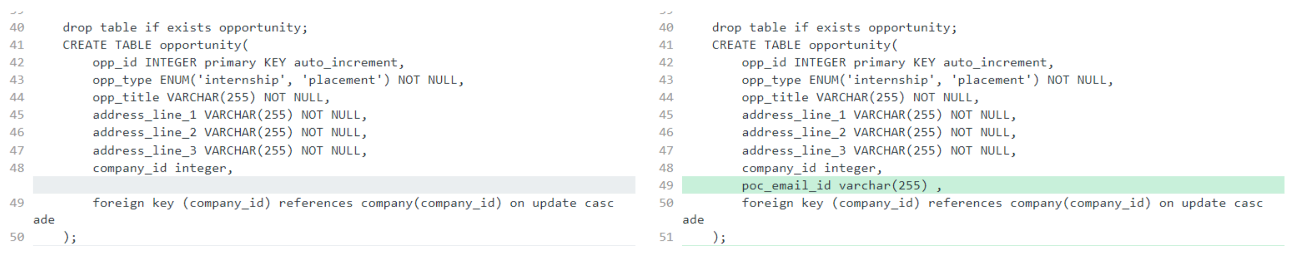
2. Implement the changes in the database as per the feedback received from stakeholders.

We had an opportunity table which contained all the opportunities, and another point_of_contact table which contained the details of all the points of contacts. But the problem here is that we had to map each opportunity with its corresponding point of contact. The method which we used initially was picking all the opportunity id's and putting them manually in the point of contact table corresponding to each point of contact.

We found this a very inefficient way, so we thought of changing it, because if let say the number of students increase in the college, and if the number of opportunities provided by the companies increases, then it will become too laborious to assign each opportunity its point of contact, which also increases chances of errors.

Hence, we didi the following major change in our database which could improve the overall performance of our placement management system.

We add the point of contact email id attribute in the opportunity table, so that we could directly add the email id of the point of contact, by this way we do not need to pick opportunity id's and map to corresponding POC's. This makes it very easy and convenient for the students to contact the respective point of contact for a particular opportunity.



```
drop table if exists opportunity;
CREATE TABLE opportunity(
    opp_id INTEGER primary KEY auto_increment,
    opp_type ENUM('internship', 'placement') NOT NULL,
    opp_title VARCHAR(255) NOT NULL,
    address_line_1 VARCHAR(255) NOT NULL,
    address_line_2 VARCHAR(255) NOT NULL,
    address_line_3 VARCHAR(255) NOT NULL,
    company_id integer,

    foreign key (company_id) references company(company_id) on update casc
ade
);
```

```
drop table if exists opportunity;
CREATE TABLE opportunity(
    opp_id INTEGER primary KEY auto_increment,
    opp_type ENUM('internship', 'placement') NOT NULL,
    opp_title VARCHAR(255) NOT NULL,
    address_line_1 VARCHAR(255) NOT NULL,
    address_line_2 VARCHAR(255) NOT NULL,
    address_line_3 VARCHAR(255) NOT NULL,
    company_id integer,
    poc_email_id varchar(255) ,
    foreign key (company_id) references company(company_id) on update casc
ade
);
```

The change is quite evident in the above image, the green highlighted thing is the change - added poc email id in the opportunity table.

This was the major change we did in our database as per the feedback received from the stakeholders.

To be Noted: All the other changes in the feedback were related to front end and backend, hence no other changes were made in the database.

3.3 Responsibility of G1 & G2:

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

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.

Vulnerability in the code(Python Backend):

Line 548: in

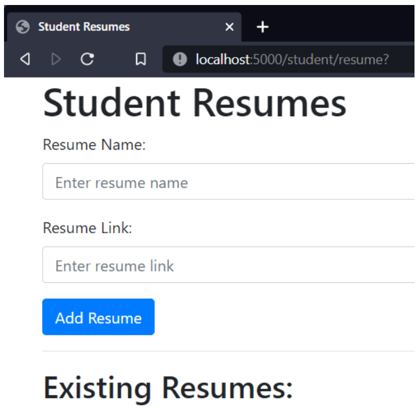
```
@app.route('/api/student/resume', methods=['POST'])
def upload_resume():
```

```
cur.execute("SELECT resume_id FROM resume WHERE resume_file = '"+resume_file+"' AND resume_file_name = '"+resume_file_name+"' FOR UPDATE")
```

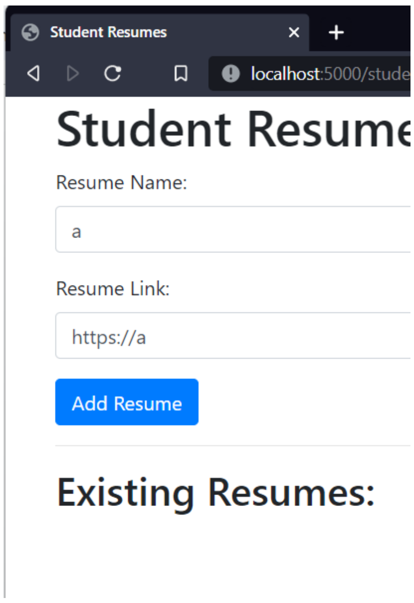
The given code shows that the resume data is being added to the database using SQL queries. An attacker can exploit this by injecting SQL statements into the resume_file_name or resume_file fields.

Steps to recreate the attack:

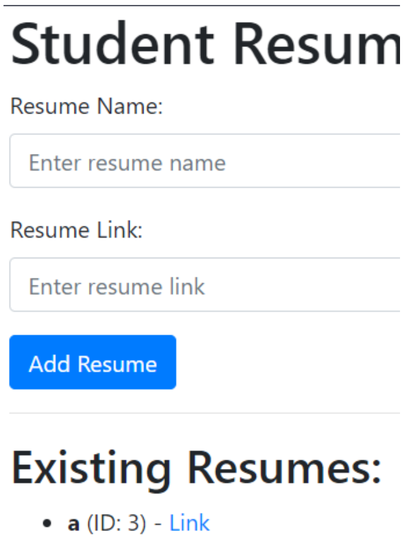
1) Assume you have student privilege/access so that you can view the student dashboard and, similarly, the resume page:



Now here, first, there are no entries, but we'll add dummy values here:



After pressing submit, the database will have this entry in the resume table. And we will refresh the page to view whether the entry has been added or not.



- **a** (ID: 3) - Link

As you can see that the entry has been added. Now we begin the SQL injection part by entering the following input into the resume name field:

'; SET FOREIGN_KEY_CHECKS = 0; SET UNIQUE_CHECKS = 0; drop table if exists company; select * from resume where resume_file_name = 'a

And corresponding in the resume link field:

http://a'; select * from resume where resume_file_name ='a

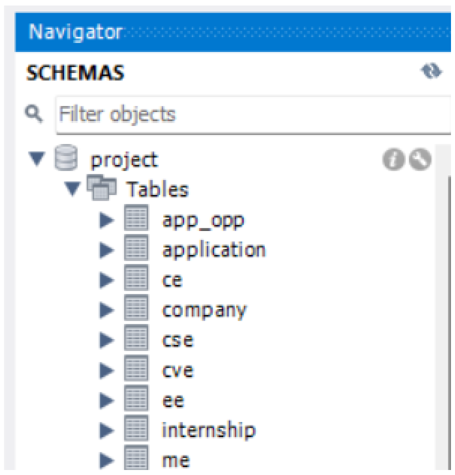
This input contains a SQL injection attack that drops the company table and selects all records from the resume table where the resume_file_name is equal to a.

The injection starts with the ***;*** character, which indicates the end of the original SQL statement. The ***SET*** statements are then used to disable foreign key checks and unique key checks. This makes it easier for the attacker to perform their attack since they do not have to worry about these constraints.

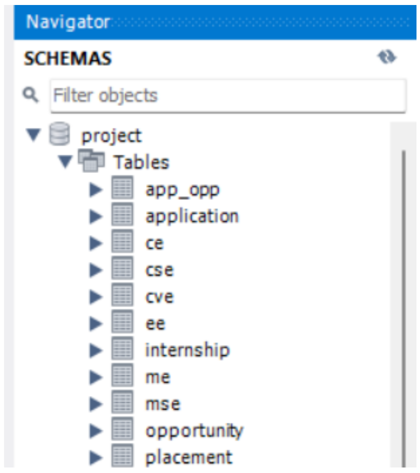
The ***DROP TABLE IF EXISTS*** statement then deletes the entire company table. This could cause major issues for the application and the users that depend on it.

Finally, the attacker uses the ***SELECT*** statement which is a dummy so everything stays working in backend.

Before executing the injection:



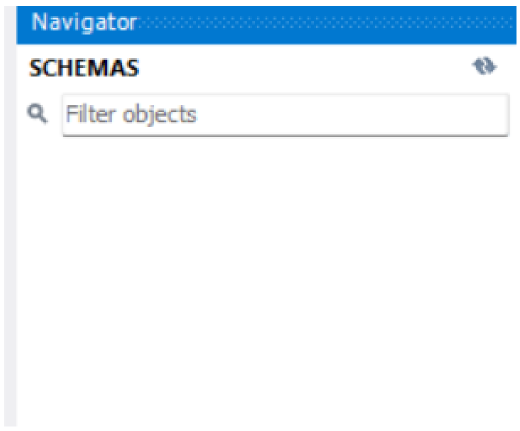
After:



Thus instead of just dropping the one table we can drop multiple tables as well by just changing the table name from company to something else. Similarly we can also add any such SQL command. Hell, we can even drop the entire database by executing the following:

'; drop database if exists project; --

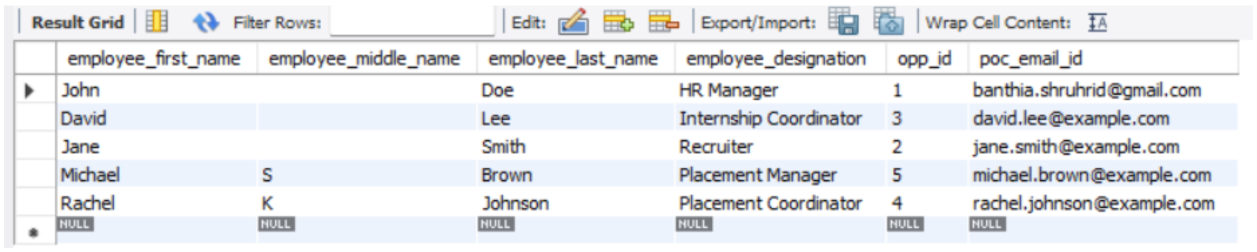
After:



This was one way of deleting things. What if someone wanted to gain poc/employee access or something? So here we demonstrate ***privilege escalation***:

So the code is such that if your access is checked by seeing if your email is available in the poc table of the database:

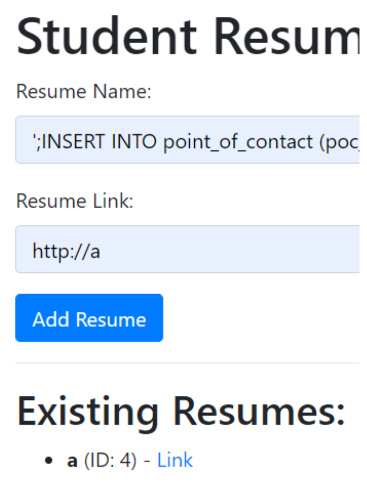
Before:



	employee_first_name	employee_middle_name	employee_last_name	employee_designation	opp_id	poc_email_id
▶	John		Doe	HR Manager	1	banthia.shruhrid@gmail.com
	David		Lee	Internship Coordinator	3	david.lee@example.com
	Jane		Smith	Recruiter	2	jane.smith@example.com
	Michael	S	Brown	Placement Manager	5	michael.brown@example.com
	Rachel	K	Johnson	Placement Coordinator	4	rachel.johnson@example.com
*	NULL	NULL	NULL	NULL	NULL	NULL

Here, we can see that only a few entries of emails are there. Next, we enter the following input into the resume_file_name field:

*';INSERT INTO point_of_contact (poc_email_id) VALUES ('dhyeythummar7@gmail.com'); commit; select * from resume where resume_file_name = 'a*



Student Resum

Resume Name:

';INSERT INTO point_of_contact (poc

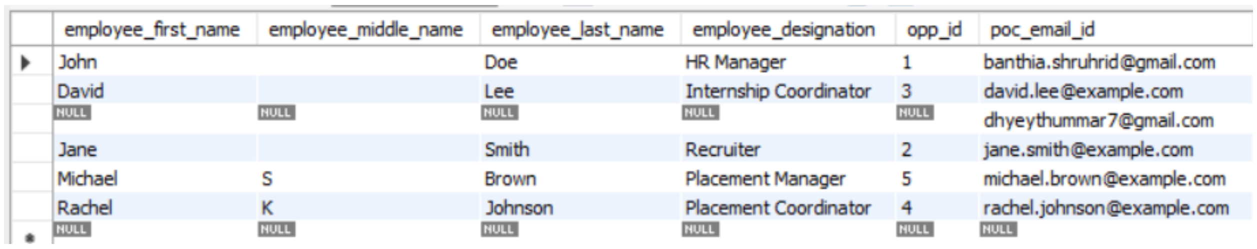
Resume Link:

http://a

Existing Resumes:

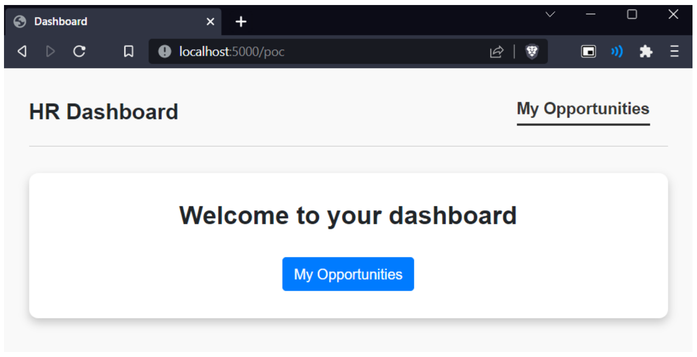
- **a** (ID: 4) - Link

This input contains a SQL injection attack that inserts a record into the point_of_contact table with the email dhyeythummar7@gmail.com. Thus we can use that email or any other dummy email to gain access to point of contact view of the application.



	employee_first_name	employee_middle_name	employee_last_name	employee_designation	opp_id	poc_email_id
▶	John		Doe	HR Manager	1	banthia.shruhrid@gmail.com
	David		Lee	Internship Coordinator	3	david.lee@example.com
	NULL	NULL	NULL	NULL	NULL	dhyeythummar7@gmail.com
	Jane		Smith	Recruiter	2	jane.smith@example.com
	Michael	S	Brown	Placement Manager	5	michael.brown@example.com
	Rachel	K	Johnson	Placement Coordinator	4	rachel.johnson@example.com
*	NULL	NULL	NULL	NULL	NULL	NULL

As you can see one entry of email has been added. Now using that we gain access to the poc view by logging out and then viewing the page http://localhost:5000/poc and signing in using the email just used above:



HR Dashboard

Welcome to your dashboard

Similarly, we can also add entries to any such table, thus being able to modify entries in the database by executing the corresponding SQL queries.

Defense against SQL injection attacks:

To prevent SQL injection attacks, it is important to use prepared statements or parameterized queries instead of dynamically generating SQL queries. This ensures that user input is treated as data rather than as code. The given code's SQL query in line 548 should be modified to use a prepared statement as follows:

Line 548: in

```
@app.route('/api/student/resume', methods=['POST'])
def upload_resume():
```

```
cur.execute("INSERT      INTO      resume(resume_file,      resume_file_name)      VALUES(%s,
%s);",(resume_file, resume_file_name))
```

Thus, here we have changed from concatenating strings to using **%s**.

After doing this we can see that our above SQL injection doesn't work, and the database is protected. Additionally, input validation and sanitization can also help prevent SQL injection attacks. In the given code, the input fields can be sanitized using a library like bleach to remove any characters that could be used in SQL injection attacks. For example, the following code can be added to sanitize the resume_file_name and resume_file fields:

```
import bleach
resume_file_name = bleach.clean(resume['resume_file_name'])
resume_file = bleach.clean(resume['resume_file'])
```

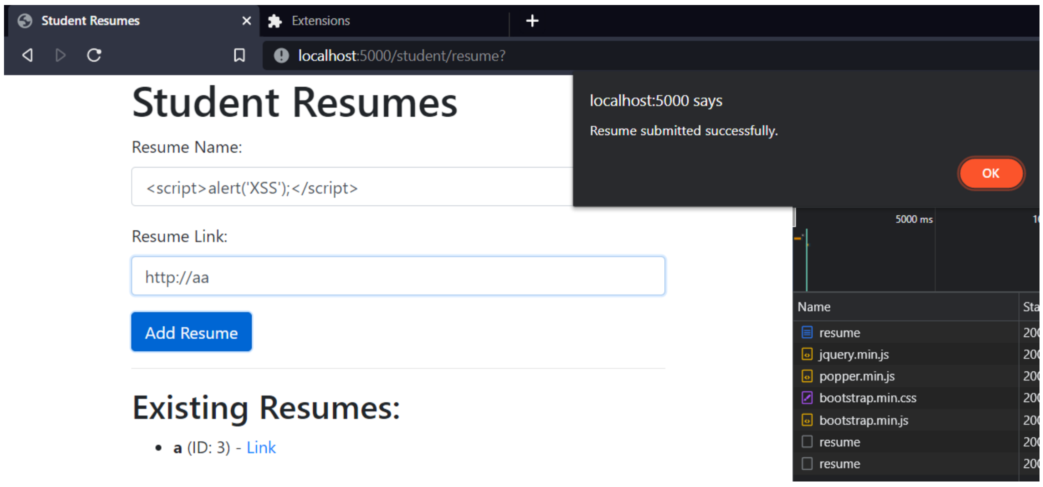
Cross-Site Scripting (XSS):

Cross-site scripting (XSS) is another common type of web attack in which an attacker injects malicious scripts into a web page viewed by other users. This can allow the attacker to steal sensitive information, such as session cookies, or perform actions on the user's behalf.

In the given code, the resume_file_name and resume_file fields are displayed on the HTML page without any sanitization. An attacker can exploit this by injecting a script into these fields that will be executed by other users who view the page.

For example, an attacker can enter the following input into the resume_file_name field:

<script>alert('XSS');</script>

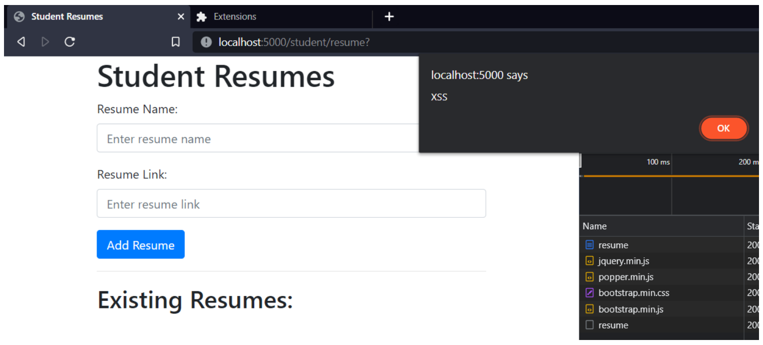


localhost:5000 says

Resume submitted successfully.

This input contains a script that displays an alert box with the message XSS when the page is viewed or reloaded.

After reloading the page:


Student Resumes

Resume Name:

Enter resume name

Resume Link:

Enter resume link

Add Resume

Existing Resumes:

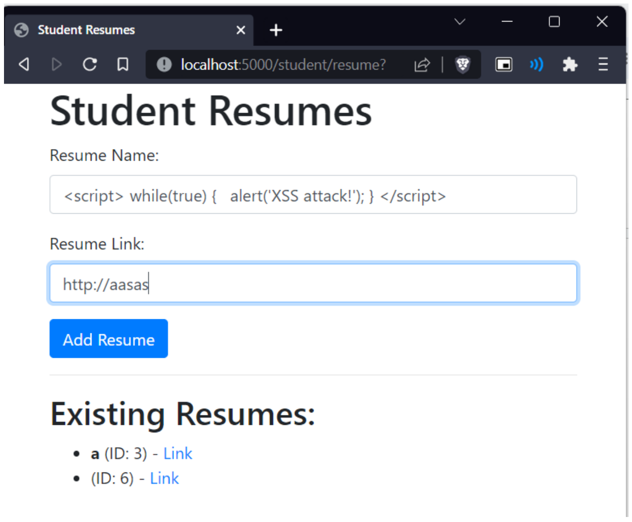
localhost:5000 says

xss

OK

Now we can do better than this by inserting an infinite loop so that the dialog box will never be closed by having the input:

<script> while(true) {   alert('XSS attack!'); } </script>


Student Resumes

Resume Name:

<script> while(true) {   alert('XSS attack!'); } </script>

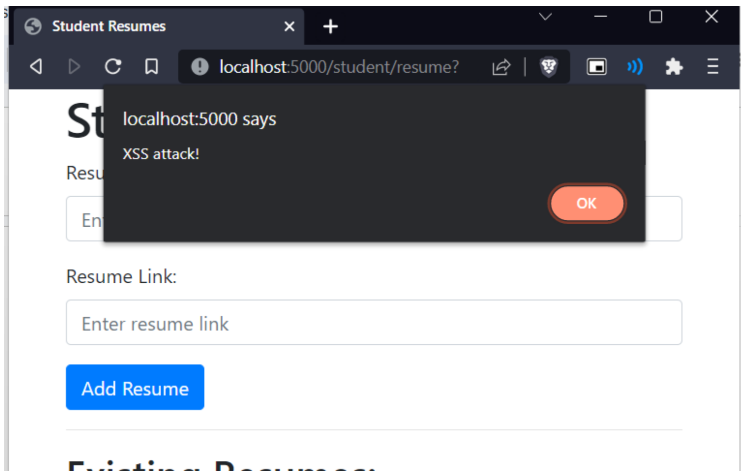
Resume Link:

http://aasas

Add Resume

Existing Resumes:

- **a** (ID: 3) - Link
- (ID: 6) - Link


localhost:5000 says

XSS attack!

OK

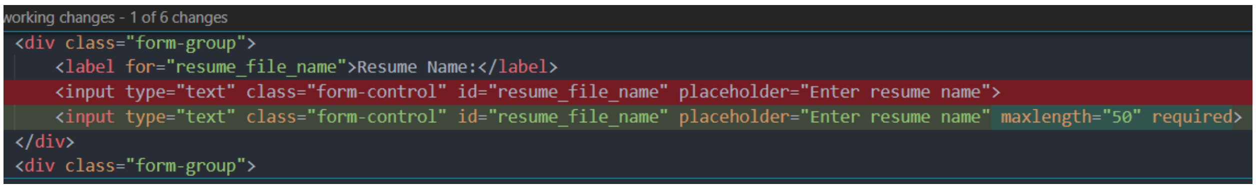
Since this a forever loop, you can never exit out of the loop. This was a demonstration of XSS.

Defense against XSS attack:

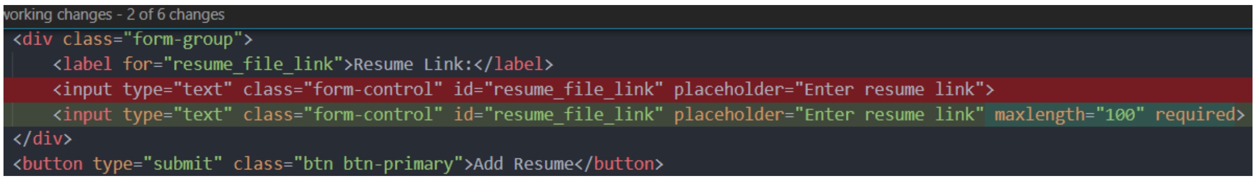
To prevent XSS attacks, we need to validate and sanitize all user input on both the client and server sides. In the code provided, we can do the following to implement input validation:

Client-side validation:

- Add the **type="text"** and **maxlength** attributes to the input elements to restrict the input length and ensure that only text is entered. (Changes in the resume_page.html)



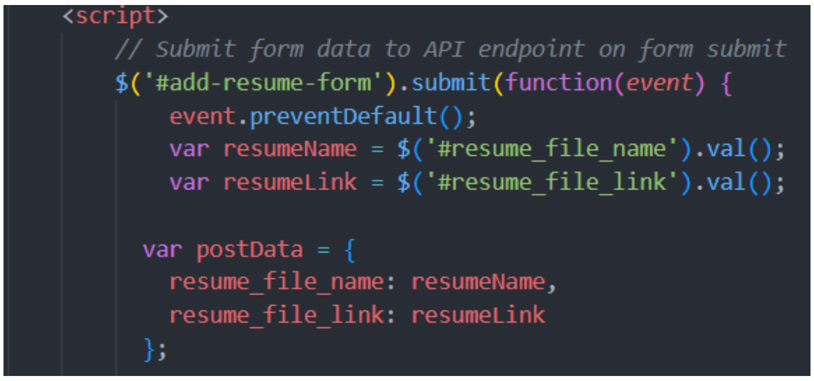
```
working changes - 1 of 6 changes
<div class="form-group">
    <label for="resume_file_name">Resume Name:</label>
    <input type="text" class="form-control" id="resume_file_name" placeholder="Enter resume name">
    <input type="text" class="form-control" id="resume_file_name" placeholder="Enter resume name" maxlength="50" required>
</div>
<div class="form-group">
```



```
working changes - 2 of 6 changes
<div class="form-group">
    <label for="resume_file_link">Resume Link:</label>
    <input type="text" class="form-control" id="resume_file_link" placeholder="Enter resume link">
    <input type="text" class="form-control" id="resume_file_link" placeholder="Enter resume link" maxlength="100" required>
</div>
<button type="submit" class="btn btn-primary">Add Resume</button>
```

- 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. (script tag in resume_page.html)

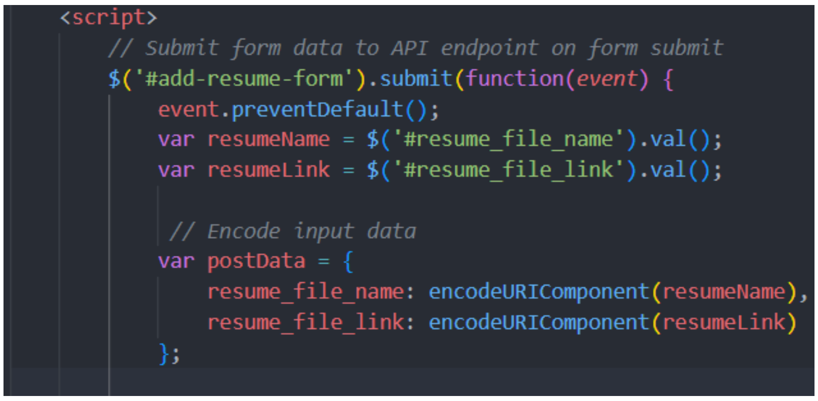
Before:



```
<script>
    // Submit form data to API endpoint on form submit
    $('#add-resume-form').submit(function(event) {
        event.preventDefault();
        var resumeName = $('#resume_file_name').val();
        var resumeLink = $('#resume_file_link').val();

      var postData = {
        resume_file_name: resumeName,
        resume_file_link: resumeLink
      };
```

After:



```
<script>
    // Submit form data to API endpoint on form submit
    $('#add-resume-form').submit(function(event) {
        event.preventDefault();
        var resumeName = $('#resume_file_name').val();
        var resumeLink = $('#resume_file_link').val();

        // Encode input data
        var postData = {
            resume_file_name: encodeURIComponent(resumeName),
            resume_file_link: encodeURIComponent(resumeLink)
        };
```

Other Attack:

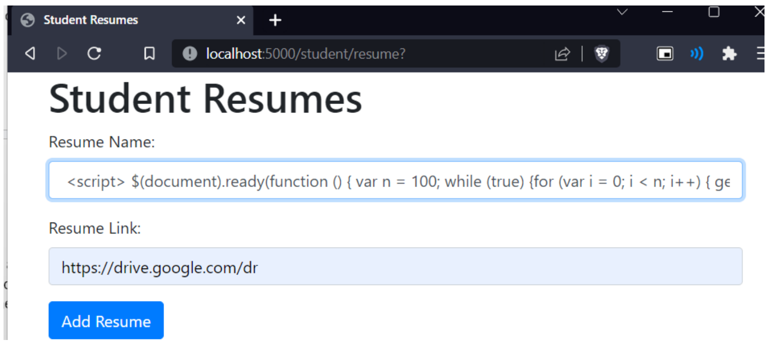
The above XSS attack is just the client side and thus not affecting the server side running the application. We can also some other things such as entering this in the resume_file_name field:

```
<script>
    $(document).ready(function () {
        var n = 100;
        while (true) {
            for (var i = 0; i < n; i++) {
                getResumes();
            }
            }
        });
</script>
```

The given code is a script that uses **jQuery** to execute a function repeatedly on document load. The function, **getResumes()**, is called multiple times within an infinite while loop. This means that the **getResumes()** function will be called continuously and repeatedly, leading to a **denial-of-service (DoS)** attack on the server hosting the API endpoint.

The attack essentially sends a large number of requests to the server, overwhelming its resources and causing it to become unresponsive or crash. This type of attack is known as a **"busy loop"** attack, where the attacker repeatedly executes a task that consumes server resources, leading to a **DoS**.

Therefore, the given code is a malicious script that aims to perform a DoS attack on the server by calling the **getResumes()** function multiple times in a loop.



Student Resumes

localhost:5000/student/resume?

Student Resumes

Resume Name:

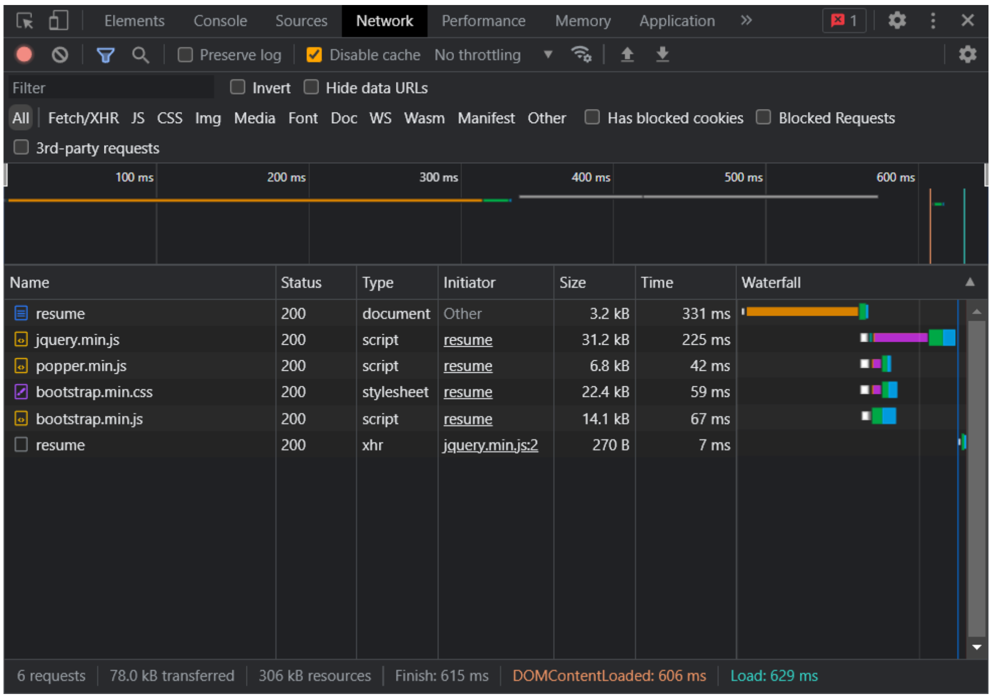
<script> $(document).ready(function () { var n = 100; while (true) {for (var i = 0; i < n; i++) { ge

Resume Link:

https://drive.google.com/dr

Add Resume

Before executing this:



Elements Console Sources Network Performance Memory Application

Preserve log · [✓ Disable cache] · No throttling

Filter · Invert · Hide data URLs

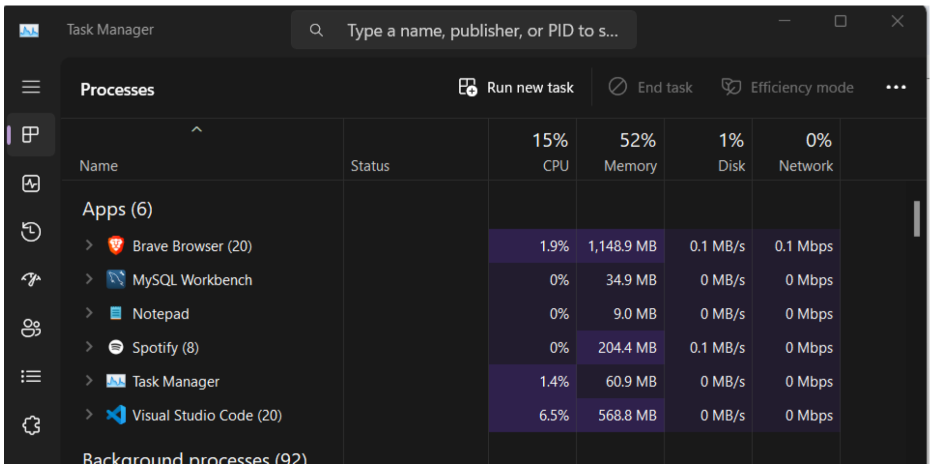
All | Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other · Has blocked cookies · Blocked Requests

3rd-party requests

100 ms 200 ms 300 ms 400 ms 500 ms 600 ms

Name	Status	Type	Initiator	Size	Time	Waterfall
resume	200	document	Other	3.2 kB	331 ms	
jquery.min.js	200	script	resume	31.2 kB	225 ms	
popper.min.js	200	script	resume	6.8 kB	42 ms	
bootstrap.min.css	200	stylesheet	resume	22.4 kB	59 ms	
bootstrap.min.js	200	script	resume	14.1 kB	67 ms	
resume	200	xhr	jquery.min.js:2	270 B	7 ms	

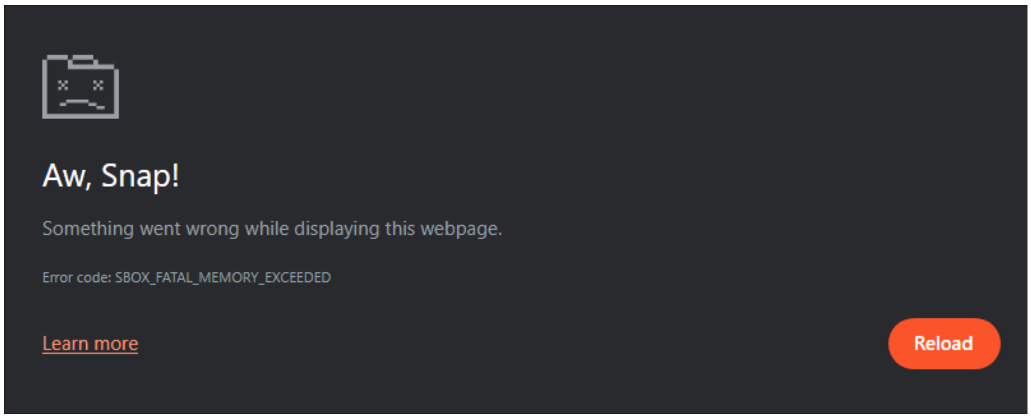
6 requests | 78.0 kB transferred | 306 kB resources | Finish: 615 ms | DOMContentLoaded: 606 ms | Load: 629 ms



Name	Status	15% CPU	52% Memory	1% Disk	0% Network
Apps (6)					
Brave Browser (20)		1.9%	1,148.9 MB	0.1 MB/s	0.1 Mbps
MySQL Workbench		0%	34.9 MB	0 MB/s	0 Mbps
Notepad		0%	9.0 MB	0 MB/s	0 Mbps
Spotify (8)		0%	204.4 MB	0.1 MB/s	0 Mbps
Task Manager		1.4%	60.9 MB	0 MB/s	0 Mbps
Visual Studio Code (20)		6.5%	568.8 MB	0 MB/s	0 Mbps
Background processes (92)					

After executing:

Browser crashes:



Aw, Snap!

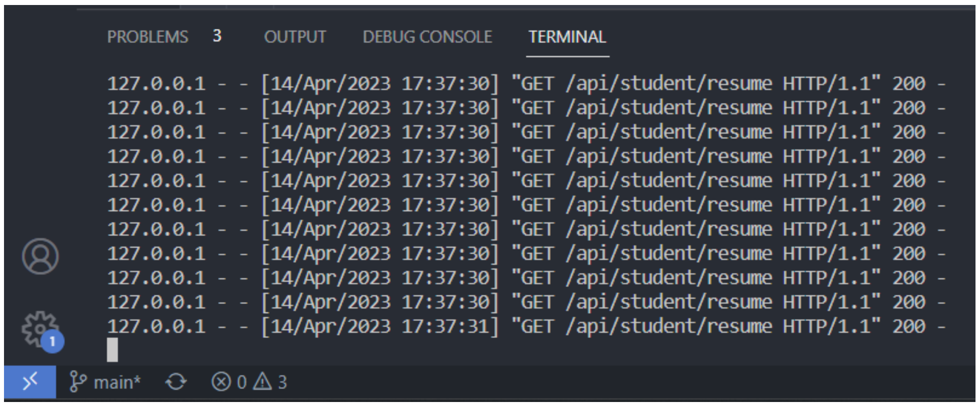
Something went wrong while displaying this webpage.

Error code: SBOX_FATAL_MEMORY_EXCEEDED

Learn more

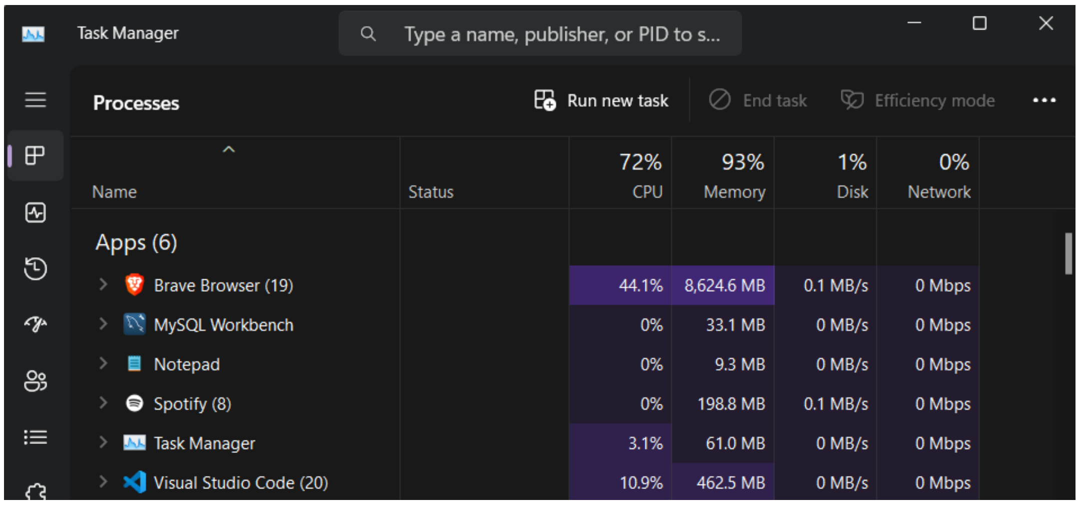
Reload

Vscode stops working:



```
127.0.0.1 - - [14/Apr/2023 17:37:30] "GET /api/student/resume HTTP/1.1" 200 -
127.0.0.1 - - [14/Apr/2023 17:37:30] "GET /api/student/resume HTTP/1.1" 200 -
127.0.0.1 - - [14/Apr/2023 17:37:30] "GET /api/student/resume HTTP/1.1" 200 -
127.0.0.1 - - [14/Apr/2023 17:37:30] "GET /api/student/resume HTTP/1.1" 200 -
127.0.0.1 - - [14/Apr/2023 17:37:30] "GET /api/student/resume HTTP/1.1" 200 -
127.0.0.1 - - [14/Apr/2023 17:37:30] "GET /api/student/resume HTTP/1.1" 200 -
127.0.0.1 - - [14/Apr/2023 17:37:30] "GET /api/student/resume HTTP/1.1" 200 -
127.0.0.1 - - [14/Apr/2023 17:37:30] "GET /api/student/resume HTTP/1.1" 200 -
127.0.0.1 - - [14/Apr/2023 17:37:30] "GET /api/student/resume HTTP/1.1" 200 -
127.0.0.1 - - [14/Apr/2023 17:37:30] "GET /api/student/resume HTTP/1.1" 200 -
127.0.0.1 - - [14/Apr/2023 17:37:31] "GET /api/student/resume HTTP/1.1" 200 -
```

Utilisation is high:



Name	Status	72% CPU	93% Memory	1% Disk	0% Network
Apps (6)					
Brave Browser (19)		44.1%	8,624.6 MB	0.1 MB/s	0 Mbps
MySQL Workbench		0%	33.1 MB	0 MB/s	0 Mbps
Notepad		0%	9.3 MB	0 MB/s	0 Mbps
Spotify (8)		0%	198.8 MB	0.1 MB/s	0 Mbps
Task Manager		3.1%	61.0 MB	0 MB/s	0 Mbps
Visual Studio Code (20)		10.9%	462.5 MB	0 MB/s	0 Mbps

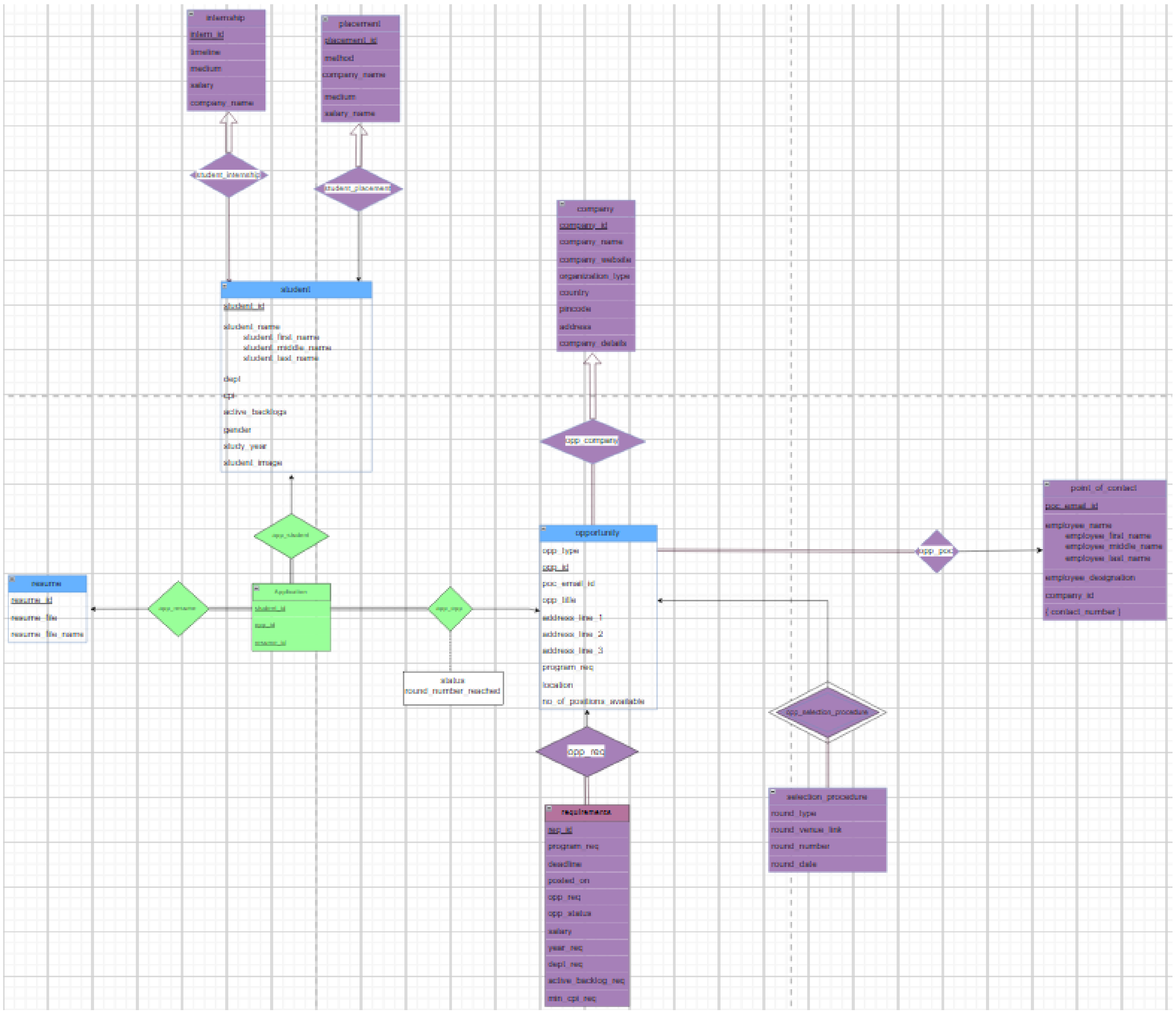
Thus, other users won't be able to able to access the application as this server gets too busy.

Defense against this:

The above DoS could be avoided by the above XSS prevention that is input sanitisation.

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.

This is the updated ER diagram after the feedback on assignment 1.



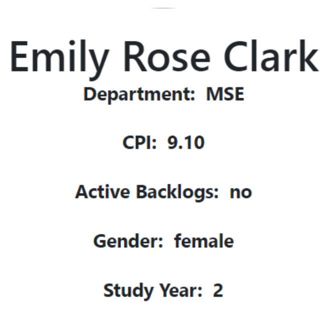
The ER Diagram can be accessed from the following link:
https://drive.google.com/file/d/1EdMChBSphpZ9cEx50h36_B1W9RiAIyUy/view?usp=sharing

Relational schema:
[Realized by both G1 and G2]
● Student (<u>student_id</u>, student_first_name, student_middle_name, student_last_name, dept, CPI, active_backlogs, gender, study_year)
● opportunity (<u>opp_id</u>, poc_email_id, opp_type, opp_title, address, program_req, location, no_of_positions_available)
● requirements (<u>req_id</u>, program_req, deadline, posted_on, opp_req, opp_status, salary, year_req, dept_req, active_backlog_req, min_cpi_req)

- Point_of_contact (employee_first_name, employee_middle_name, employee_last_name, employee_designation, company_id, <u>poc_email_id</u>, {contact_number})
- Company (company_id, company_name, company_website, organization_type, industry_type, country, pincode, address, company_details)
- Placement (<u>placement_id</u>, method (ppo or interview), medium (internal or external), salary, company_name)
- Internship (<u>internship_id</u>, timeline (duration - year), medium (internal or external), salary, company_name)
- Resume (<u>resume_id</u>, resume_file, resume_file_name)
- Selection Procedure (<u>round_type (test or interview), round_venue/link, round_number, round_date</u>)
- Application (<u>student_id</u>, <u>opp_id</u>, <u>resume_id</u>)

Below are the screenshots from the UI showing that tables from the ER diagram are present on our system (website).

Student table



Emily Rose Clark

Department: MSE

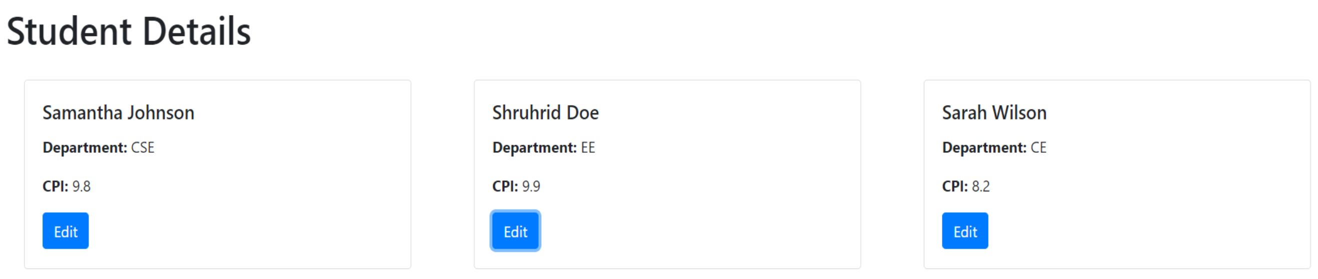
CPI: 9.10

Active Backlogs: no

Gender: female

Study Year: 2

Above shows the details of a particular student, as per the attributes in the student table in the database.



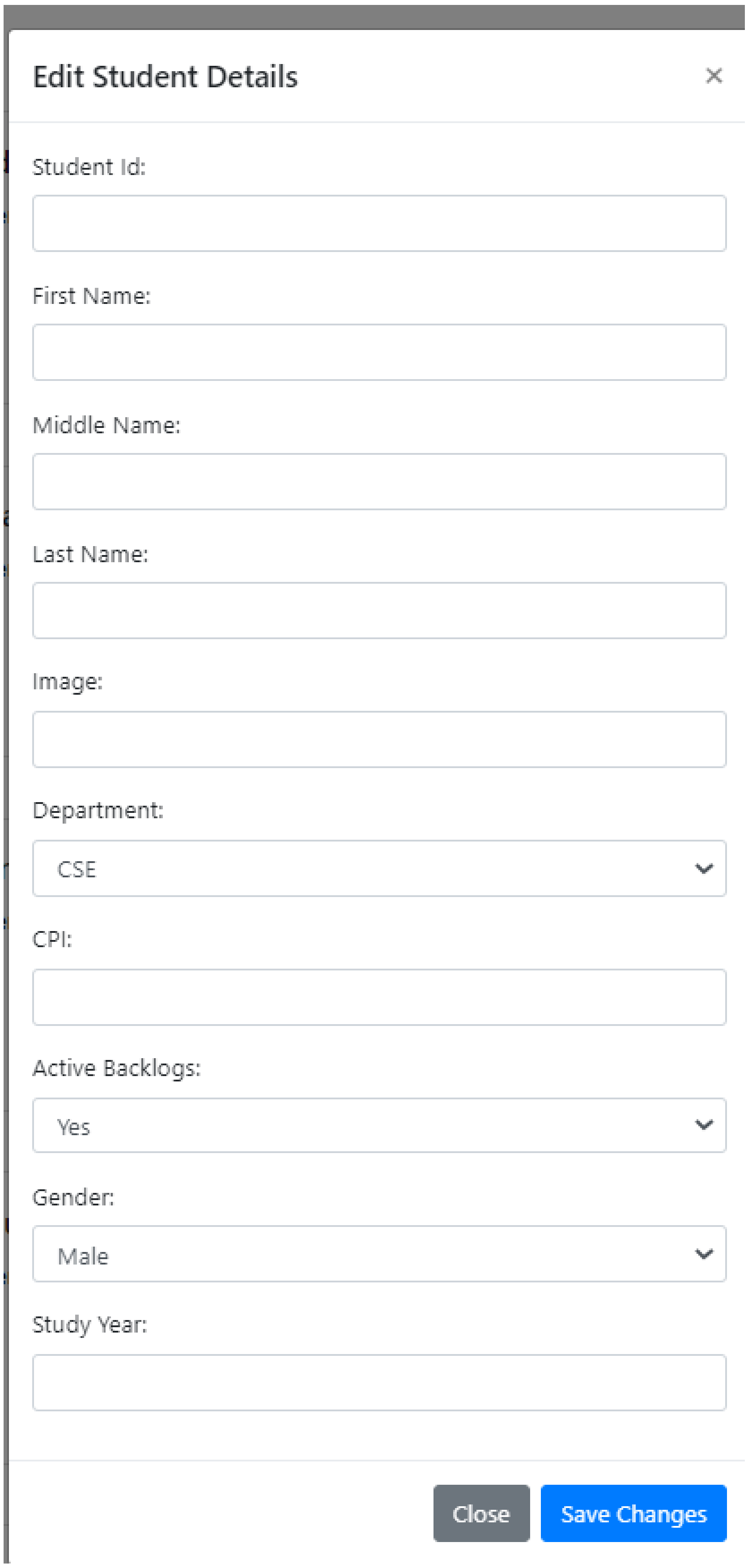
Student Details

Samantha Johnson
Department: CSE
CPI: 9.8

Shruhrid Doe
Department: EE
CPI: 9.9

Sarah Wilson
Department: CE
CPI: 8.2

These are the student details visible to the CDS team, and they can also be edited by clicking on the edit button.

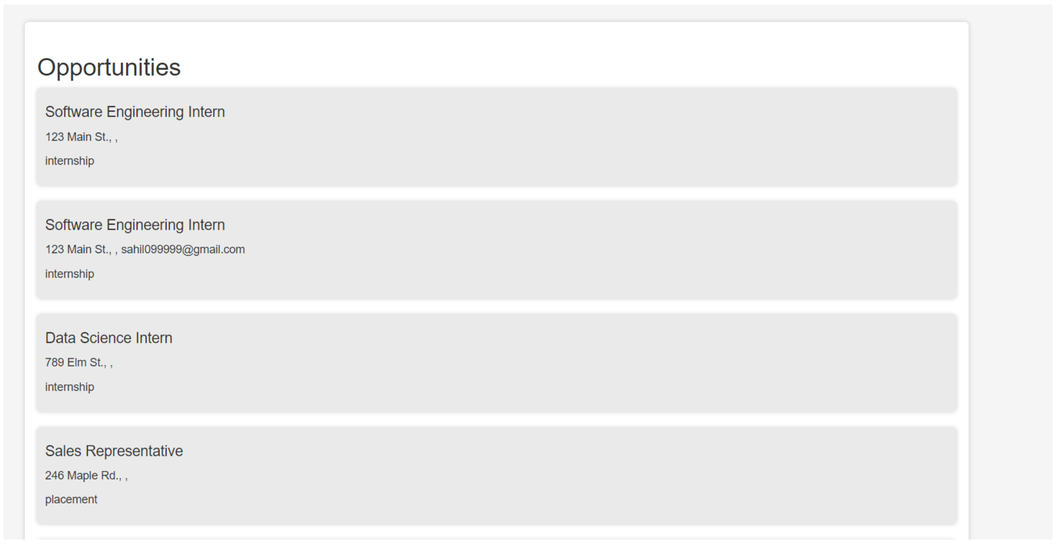


Edit Student Details

Student Id:	
First Name:	
Middle Name:	
Last Name:	
Image:	
Department:	CSE
CPI:	
Active Backlogs:	Yes
Gender:	Male
Study Year:	

The above form contains all the fields as mentioned in the student table in the database and also follows all the constraints.

Opportunity table


Opportunities

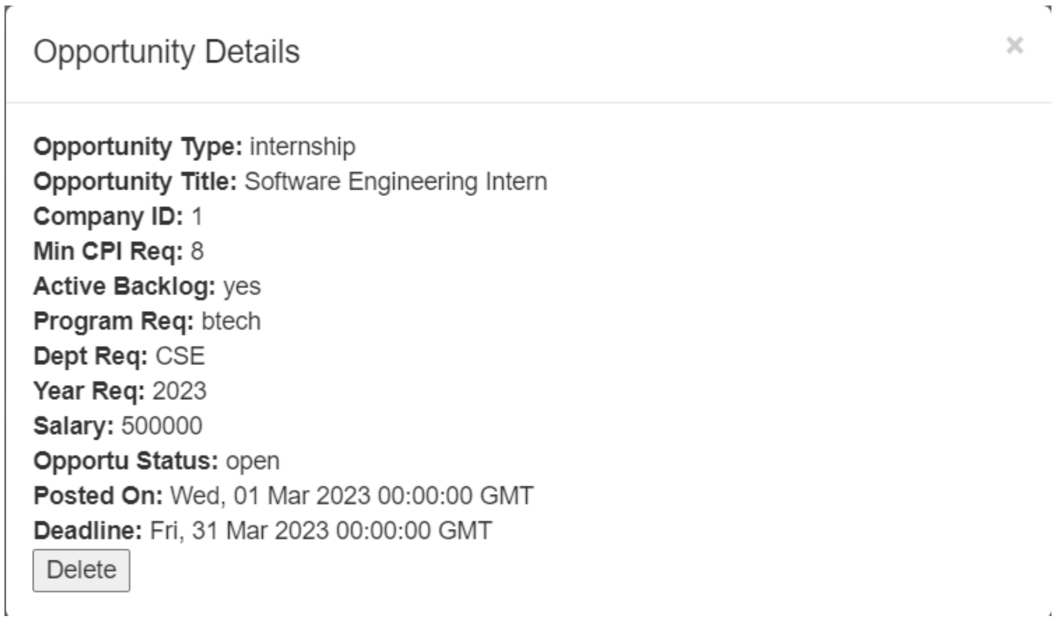
Software Engineering Intern
123 Main St., ,
internship

Software Engineering Intern
123 Main St., , sahil099999@gmail.com
internship

Data Science Intern
789 Elm St., ,
internship

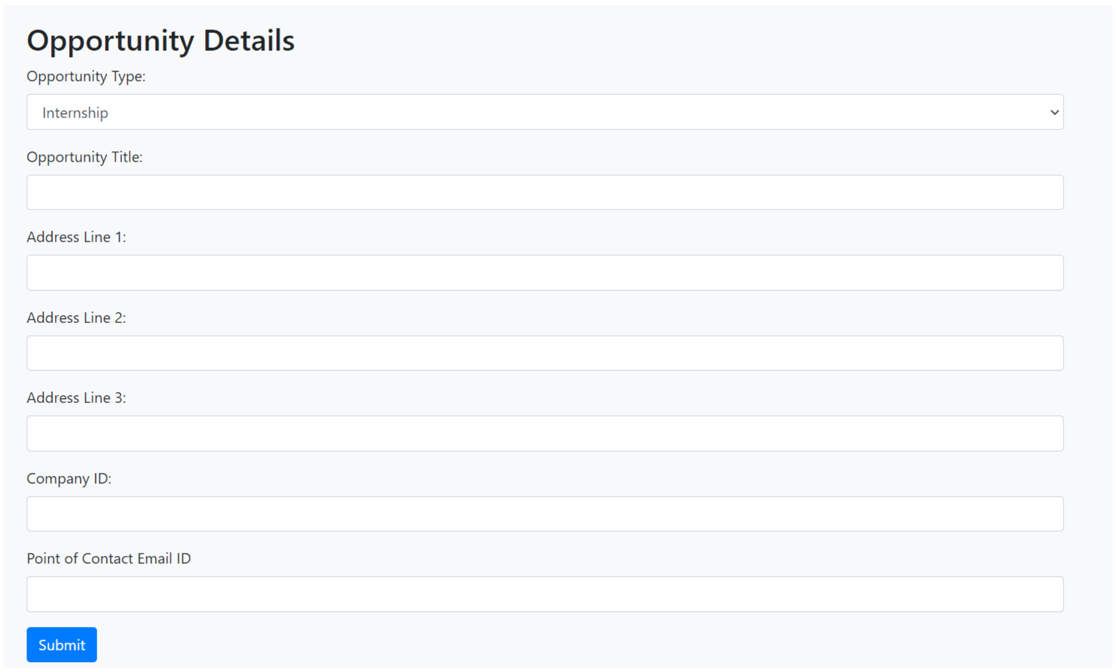
Sales Representative
246 Maple Rd., ,
placement

List of opportunities seen by student and CDS


Opportunity Details

Opportunity Type: internship
Opportunity Title: Software Engineering Intern
Company ID: 1
Min CPI Req: 8
Active Backlog: yes
Program Req: btech
Dept Req: CSE
Year Req: 2023
Salary: 500000
Opportu Status: open
Posted On: Wed, 01 Mar 2023 00:00:00 GMT
Deadline: Fri, 31 Mar 2023 00:00:00 GMT

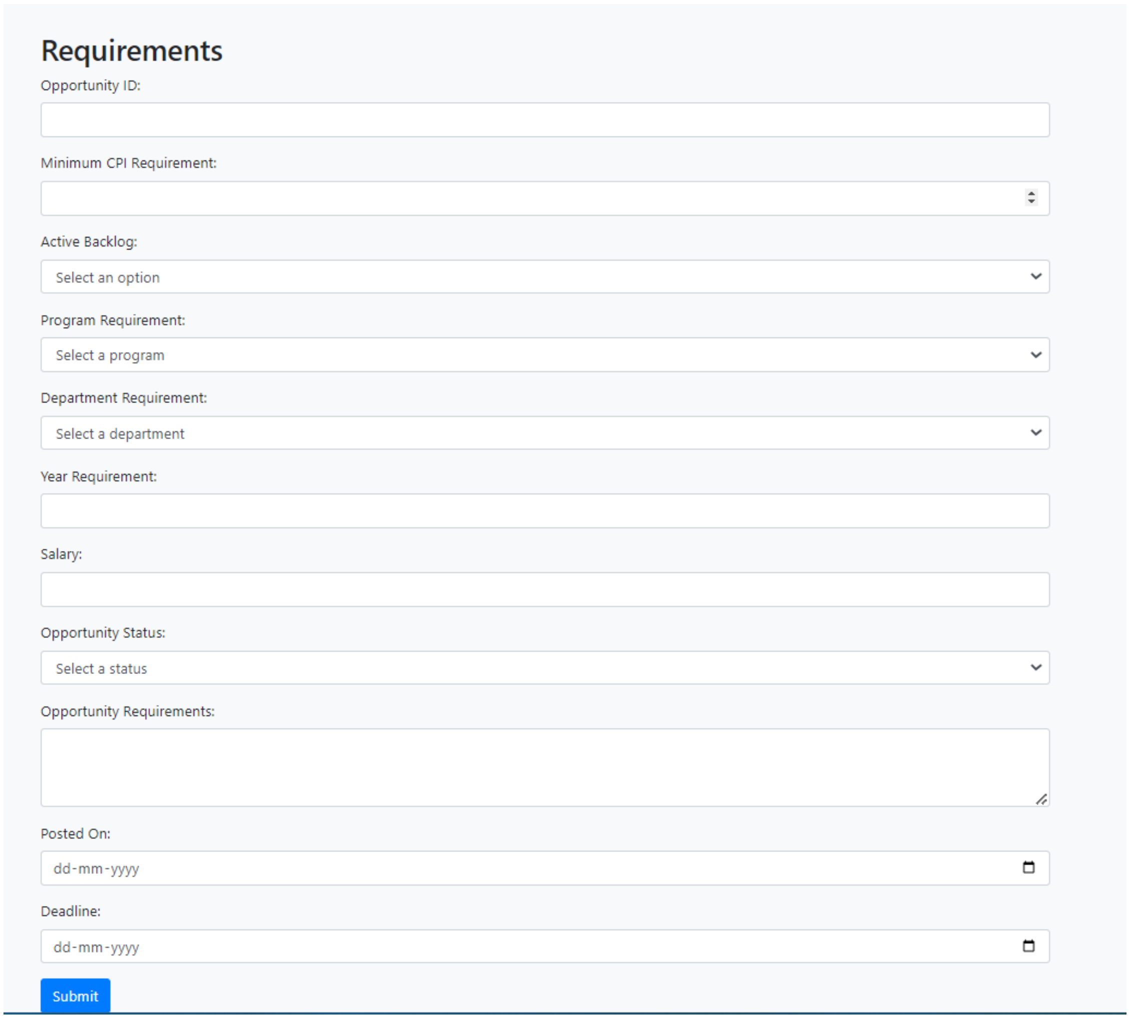
Upon clicking on the opportunity, we get to see the details of it as per the fields mentioned in the database.


Opportunity Details

Opportunity Type:	Internship
Opportunity Title:	
Address Line 1:	
Address Line 2:	
Address Line 3:	
Company ID:	
Point of Contact Email ID	

This above thing is from the CDS View, they can add an opportunity by filling up the details. All the constraints are specified here as per the opportunity table in the database.

Requirements table



Requirements

Opportunity ID:

Minimum CPI Requirement:

Active Backlog: Select an option

Program Requirement: Select a program

Department Requirement: Select a department

Year Requirement:

Salary:

Opportunity Status: Select a status

Opportunity Requirements:

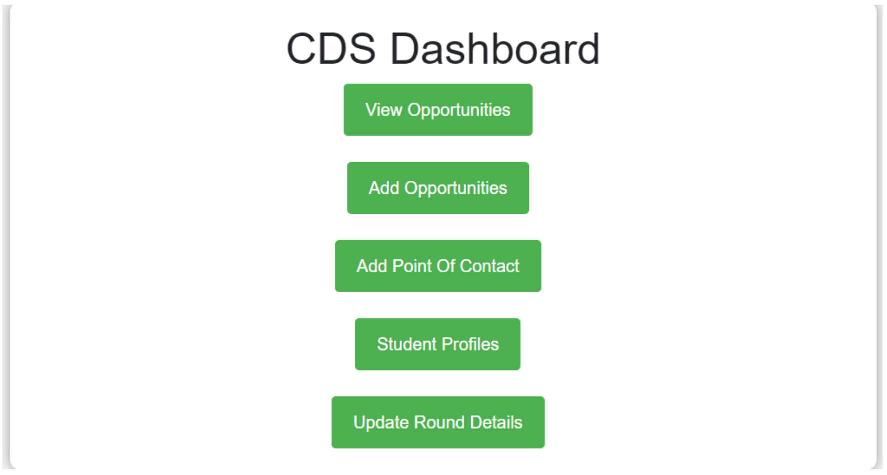
Posted On: dd-mm-yyyy

Deadline: dd-mm-yyyy

Submit

The above form will be filled by the CDS to add requirements for an opportunity in addition to the opportunity form.

Point_of_conatct table



CDS Dashboard

View Opportunities

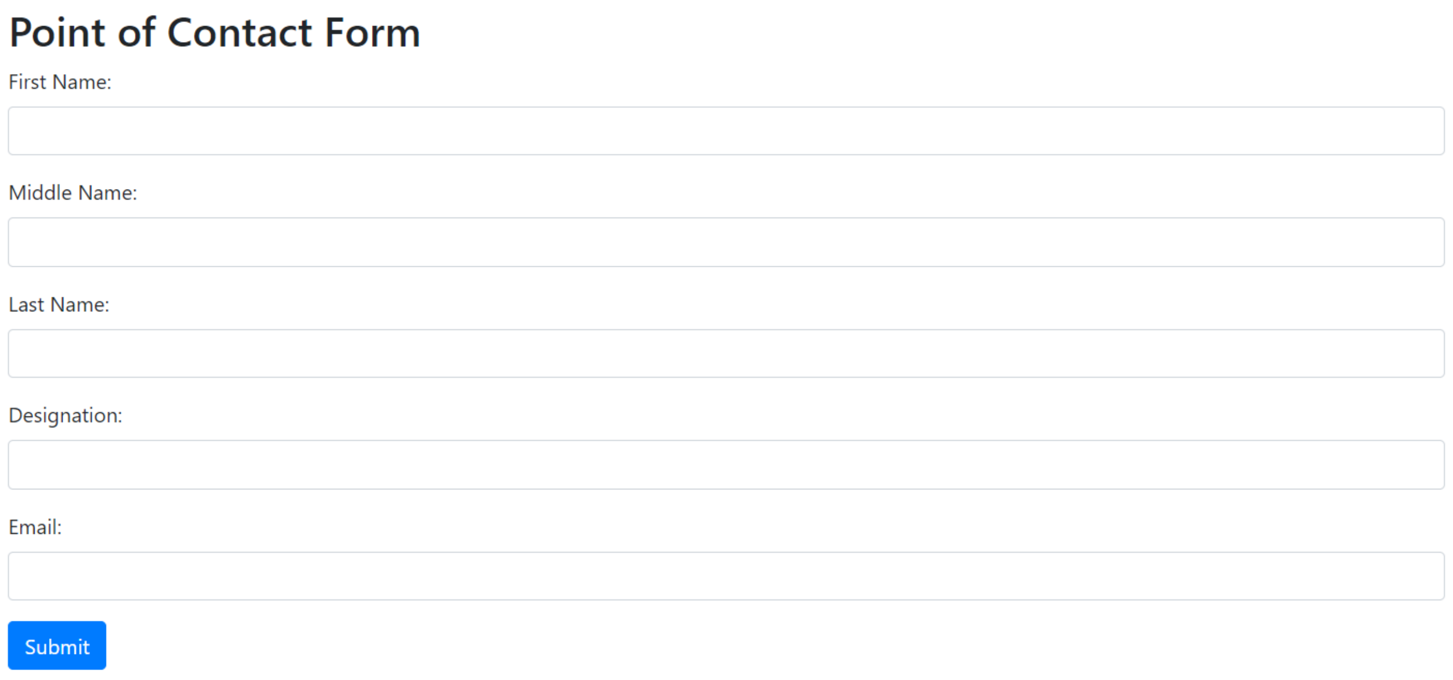
Add Opportunities

Add Point Of Contact

Student Profiles

Update Round Details

CDS has the access to above 5 options, out of which adding point of contact details is also one option. Upon clicking on the add point of contact option, the following page opens up, which satisfies the constraints specified in the point of contact table.



Point of Contact Form

First Name:

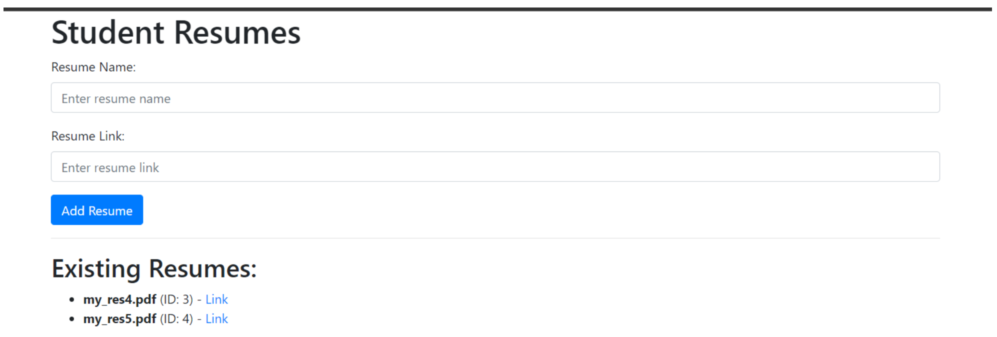
Middle Name:

Last Name:

Designation:

Email:

Resume table



Student Resumes

Resume Name:

Enter resume name

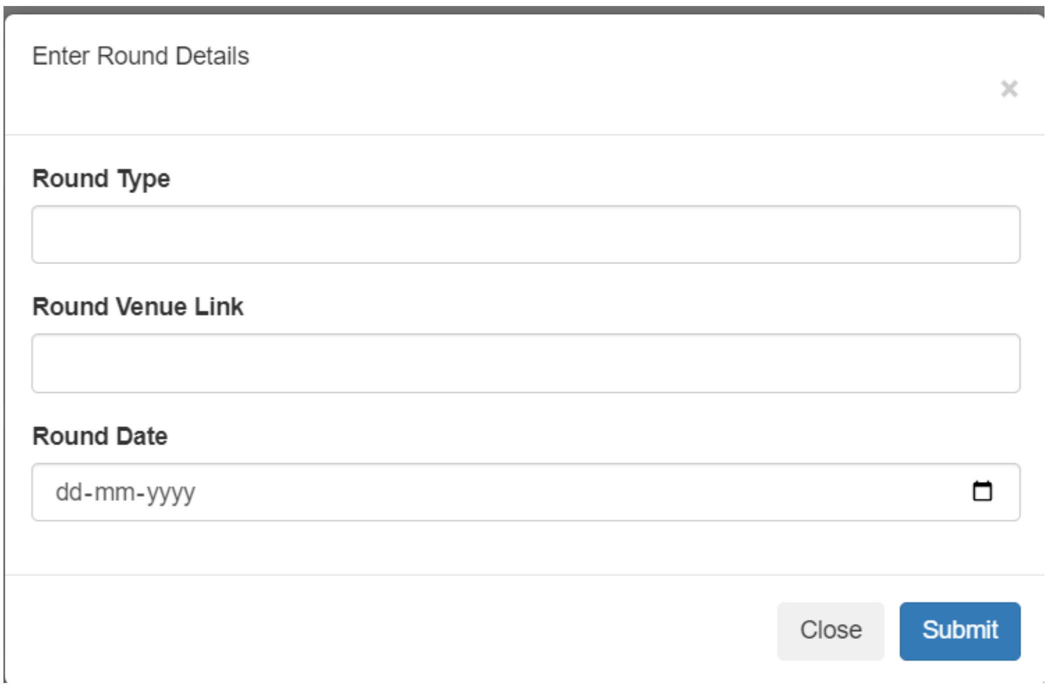
Resume Link:

Enter resume link

Existing Resumes:

- **my_res4.pdf** (ID: 3) - Link
- **my_res5.pdf** (ID: 4) - Link

Selection Procedure table



Enter Round Details

Round Type

Round Venue Link

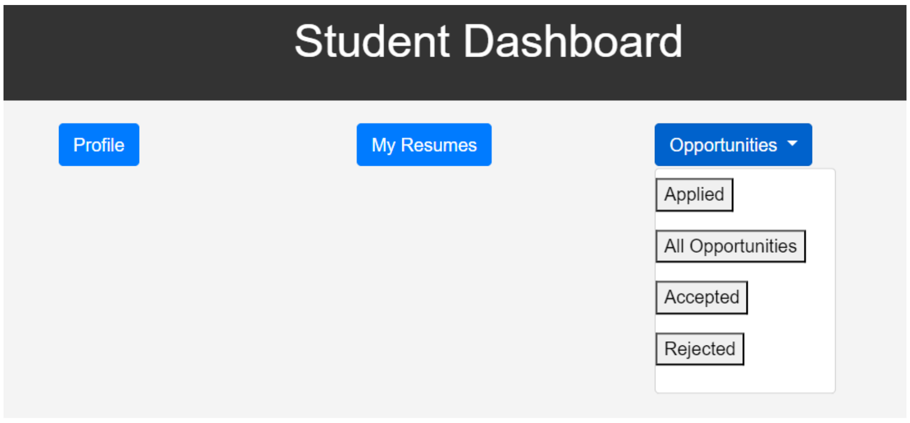
Round Date

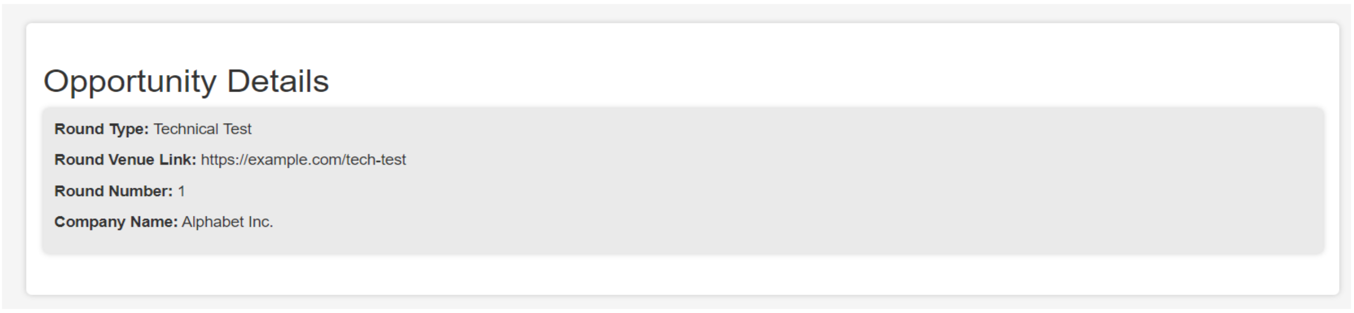
dd-mm-yyyy

Above is the form to update the status of each opportunity, that is which round is going on, etc. The above details have been mentioned in the selection procedure table in the database, and as you can see, all the fields are visible in the UI, and obviously, all the constraints are satisfied.

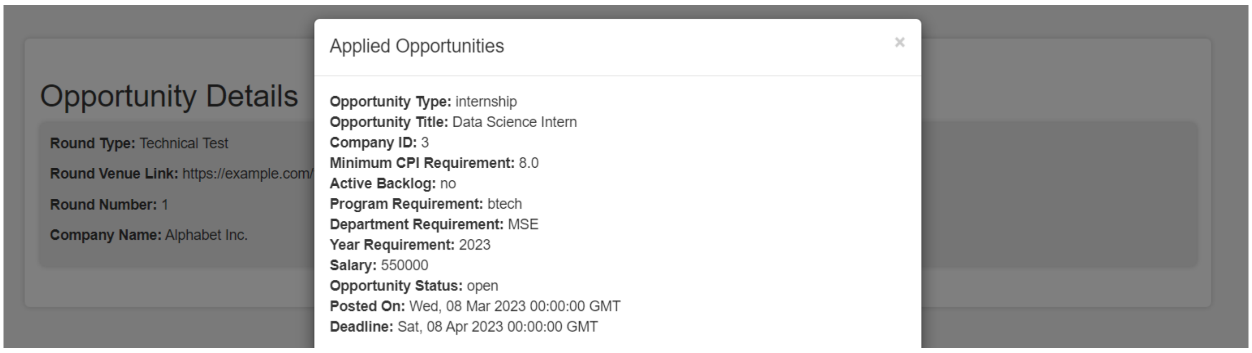
Application table

Selecting the applied option opens up the list of opportunities to which the student has applied.




Opportunity Details

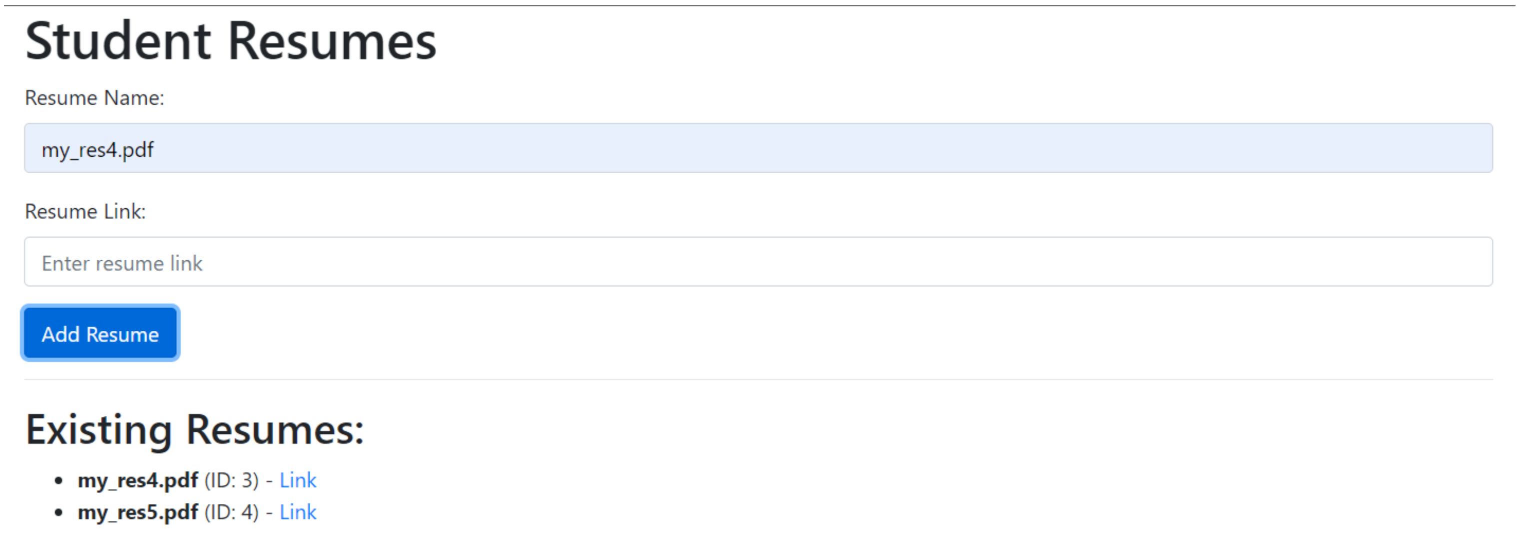
Round Type: Technical Test
Round Venue Link: https://example.com/tech-test
Round Number: 1
Company Name: Alphabet Inc.


Applied Opportunities

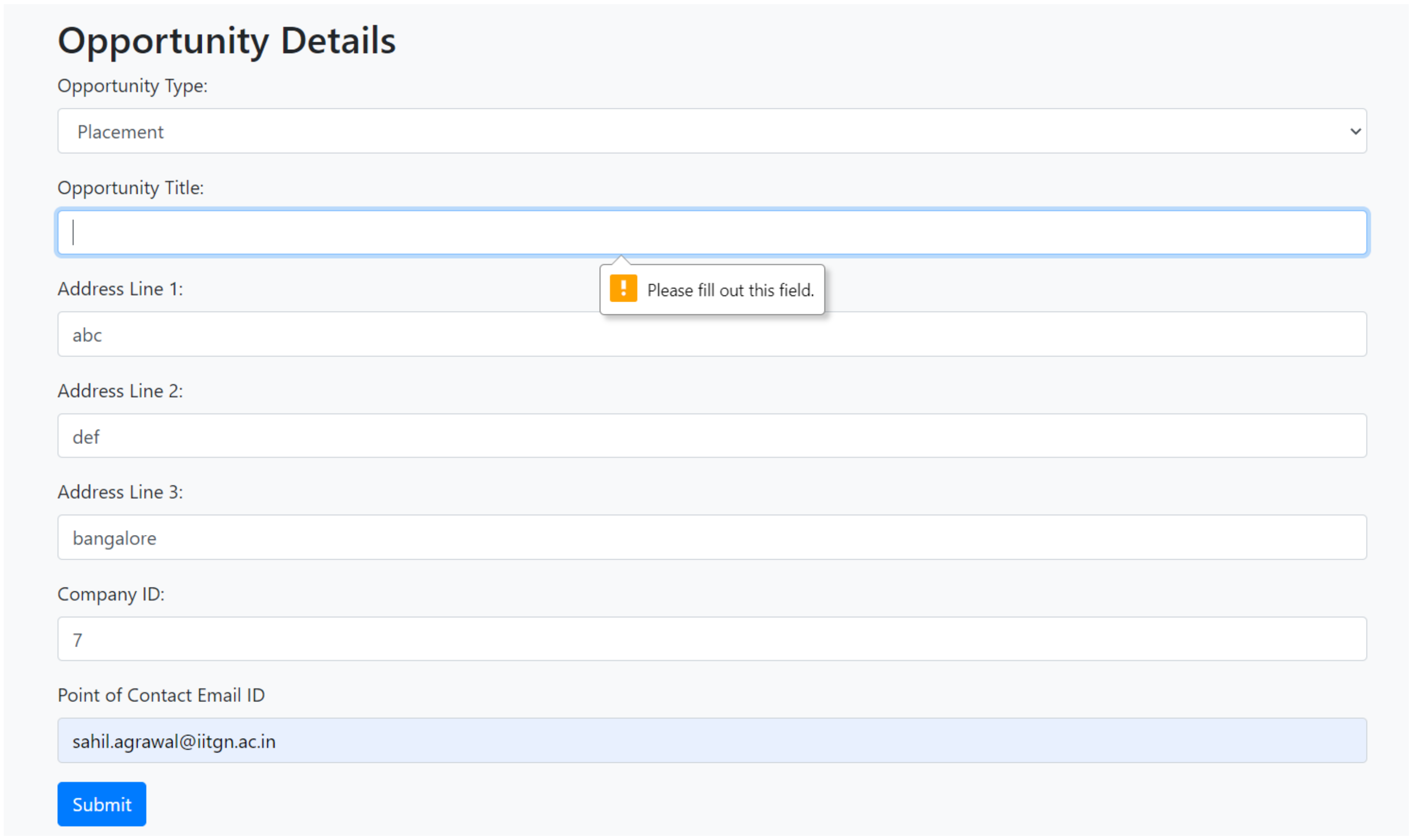
Opportunity Type: internship
Opportunity Title: Data Science Intern
Company ID: 3
Minimum CPI Requirement: 8.0
Active Backlog: no
Program Requirement: btech
Department Requirement: MSE
Year Requirement: 2023
Salary: 550000
Opportunity Status: open
Posted On: Wed, 08 Mar 2023 00:00:00 GMT
Deadline: Sat, 08 Apr 2023 00:00:00 GMT

The above shows the details of the opportunity to which the student has applied.

Constraints are followed!


Student Resumes
Resume Name:
my_res4.pdf
Resume Link:
Enter resume link
Add Resume
Existing Resumes:
- **my_res4.pdf** (ID: 3) - Link
- **my_res5.pdf** (ID: 4) - Link

As per the constraint, the resume link cannot be NULL, therefore when we try to add a resume without putting any resume link, then the form is not getting submitted.


Opportunity Details
Opportunity Type:
Placement
Opportunity Title:

Please fill out this field.

Address Line 1:
abc
Address Line 2:
def
Address Line 3:
bangalore
Company ID:
7
Point of Contact Email ID
sahil.agrawal@iitgn.ac.in
Submit

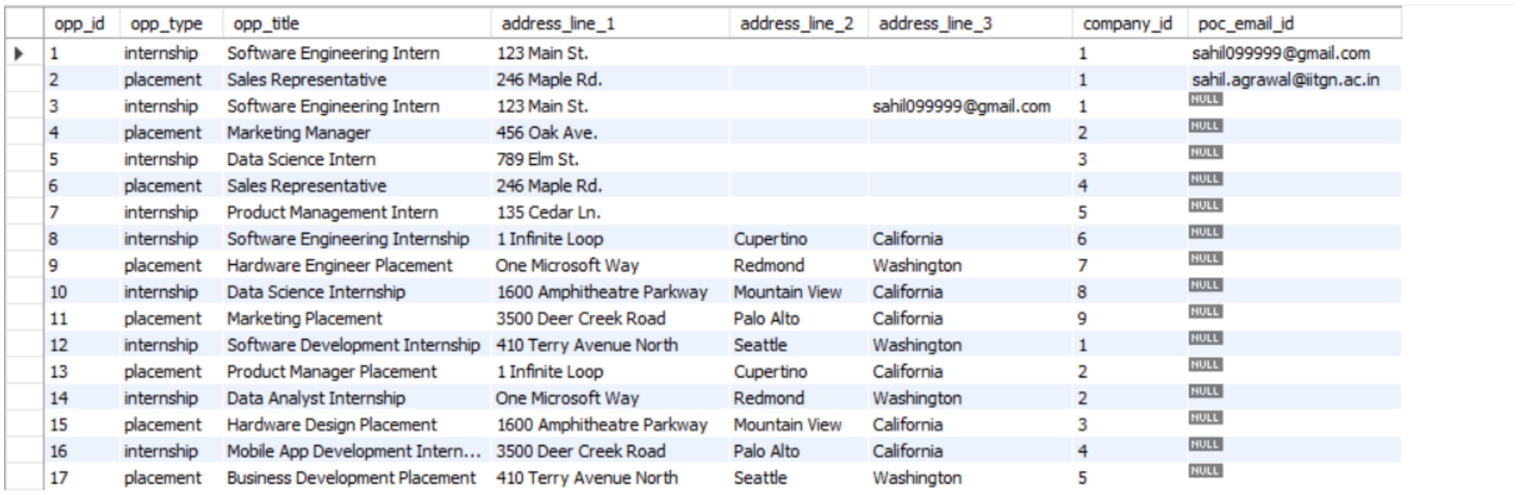
As per the constraint, the opportunity title cannot be NULL, therefore when we try to add a new opportunity and submit the above form, it doesn't get submitted and a message shows up asking us to fill the field. Same is the case of other attributes such as Address Line 1, 2 and 3. Similarly, the constraint on the opportunity type that it can only contain any one of the 2 values internship or placement, the UI gives us a dropdown with only 2 options placement and internship, so it becomes mandatory for us to select from these two, hence following the constraint specified in the database.

Opportunity ID is the primary key of the opportunity table and a foreign key of the requirement table. The constraint is as follows:

foreign key (opp_id) references opportunity(opp_id) on update cascade on delete cascade

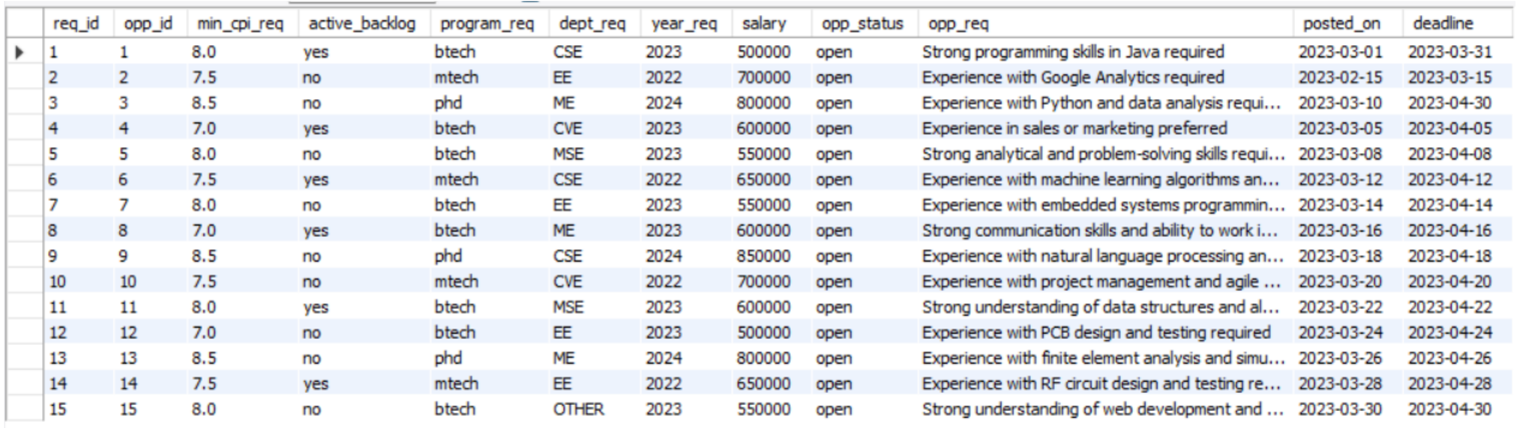
So, if I delete a particular opportunity ID from the opportunity table, then the requirements corresponding to that opportunity ID should get deleted.

This is the opportunity table initially.



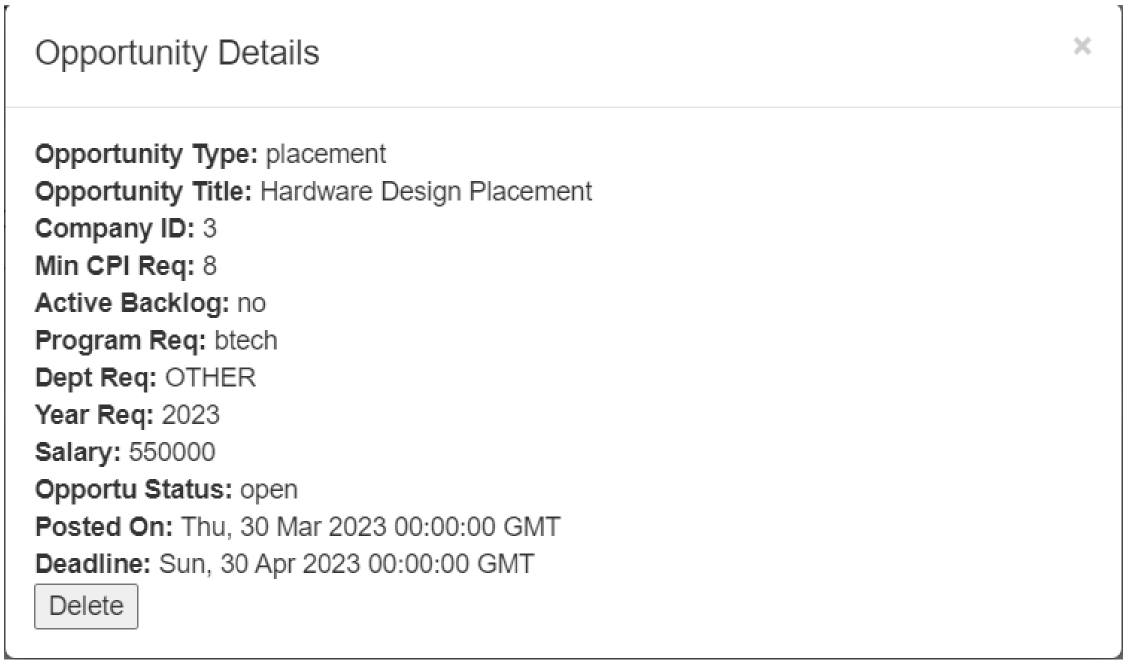
opp_id	opp_type	opp_title	address_line_1	address_line_2	address_line_3	company_id	poc_email_id
1	internship	Software Engineering Intern	123 Main St.			1	sahil099999@gmail.com
2	placement	Sales Representative	246 Maple Rd.			1	sahil.agrawal@iitgn.ac.in
3	internship	Software Engineering Intern	123 Main St.		sahil099999@gmail.com	1	NULL
4	placement	Marketing Manager	456 Oak Ave.			2	NULL
5	internship	Data Science Intern	789 Elm St.			3	NULL
6	placement	Sales Representative	246 Maple Rd.			4	NULL
7	internship	Product Management Intern	135 Cedar Ln.			5	NULL
8	internship	Software Engineering Internship	1 Infinite Loop	Cupertino	California	6	NULL
9	placement	Hardware Engineer Placement	One Microsoft Way	Redmond	Washington	7	NULL
10	internship	Data Science Internship	1600 Amphitheatre Parkway	Mountain View	California	8	NULL
11	placement	Marketing Placement	3500 Deer Creek Road	Palo Alto	California	9	NULL
12	internship	Software Development Internship	410 Terry Avenue North	Seattle	Washington	1	NULL
13	placement	Product Manager Placement	1 Infinite Loop	Cupertino	California	2	NULL
14	internship	Data Analyst Internship	One Microsoft Way	Redmond	Washington	2	NULL
15	placement	Hardware Design Placement	1600 Amphitheatre Parkway	Mountain View	California	3	NULL
16	internship	Mobile App Development Intern...	3500 Deer Creek Road	Palo Alto	California	4	NULL
17	placement	Business Development Placement	410 Terry Avenue North	Seattle	Washington	5	NULL

This is the requirements table initially.



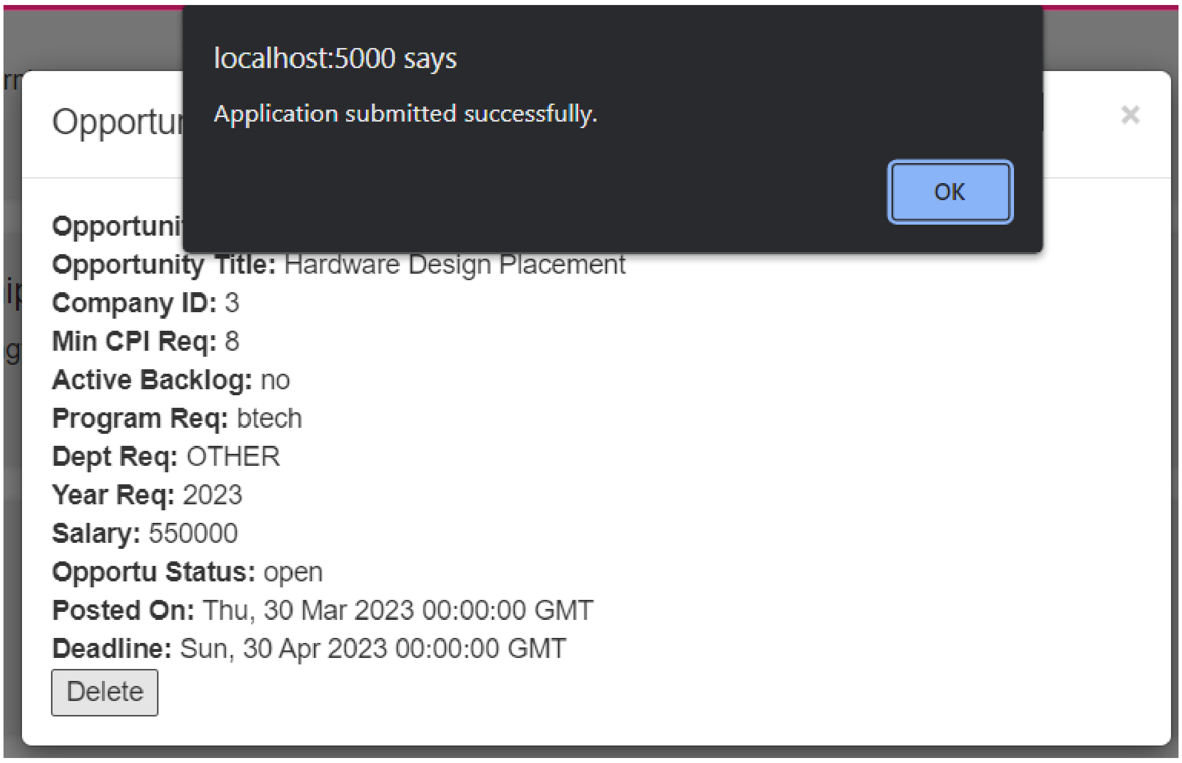
req_id	opp_id	min_cpi_req	active_backlog	program_req	dept_req	year_req	salary	opp_status	opp_req	posted_on	deadline
1	1	8.0	yes	btech	CSE	2023	500000	open	Strong programming skills in Java required	2023-03-01	2023-03-31
2	2	7.5	no	mtech	EE	2022	700000	open	Experience with Google Analytics required	2023-02-15	2023-03-15
3	3	8.5	no	phd	ME	2024	800000	open	Experience with Python and data analysis requi...	2023-03-10	2023-04-30
4	4	7.0	yes	btech	CVE	2023	600000	open	Experience in sales or marketing preferred	2023-03-05	2023-04-05
5	5	8.0	no	btech	MSE	2023	550000	open	Strong analytical and problem-solving skills requi...	2023-03-08	2023-04-08
6	6	7.5	yes	mtech	CSE	2022	650000	open	Experience with machine learning algorithms an...	2023-03-12	2023-04-12
7	7	8.0	no	btech	EE	2023	550000	open	Experience with embedded systems programmin...	2023-03-14	2023-04-14
8	8	7.0	yes	btech	ME	2023	600000	open	Strong communication skills and ability to work i...	2023-03-16	2023-04-16
9	9	8.5	no	phd	CSE	2024	850000	open	Experience with natural language processing an...	2023-03-18	2023-04-18
10	10	7.5	no	mtech	CVE	2022	700000	open	Experience with project management and agile ...	2023-03-20	2023-04-20
11	11	8.0	yes	btech	MSE	2023	600000	open	Strong understanding of data structures and al...	2023-03-22	2023-04-22
12	12	7.0	no	btech	EE	2023	500000	open	Experience with PCB design and testing required	2023-03-24	2023-04-24
13	13	8.5	no	phd	ME	2024	800000	open	Experience with finite element analysis and simu...	2023-03-26	2023-04-26
14	14	7.5	yes	mtech	EE	2022	650000	open	Experience with RF circuit design and testing re...	2023-03-28	2023-04-28
15	15	8.0	no	btech	OTHER	2023	550000	open	Strong understanding of web development and ...	2023-03-30	2023-04-30

I wish to delete this particular opportunity with 15 being its opportunity ID.



Opportunity Details

Opportunity Type: placement
Opportunity Title: Hardware Design Placement
Company ID: 3
Min CPI Req: 8
Active Backlog: no
Program Req: btech
Dept Req: OTHER
Year Req: 2023
Salary: 550000
Opportu Status: open
Posted On: Thu, 30 Mar 2023 00:00:00 GMT
Deadline: Sun, 30 Apr 2023 00:00:00 GMT



localhost:5000 says

Application submitted successfully.

OK

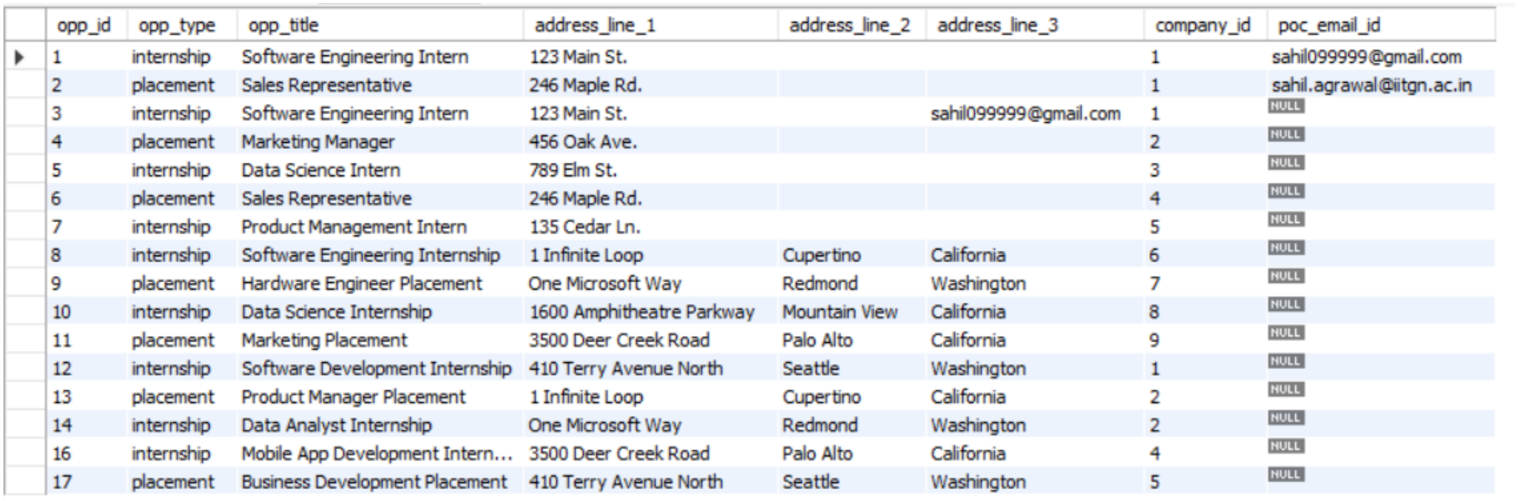
Opportu

Opportuni

Opportunity Title: Hardware Design Placement
Company ID: 3
Min CPI Req: 8
Active Backlog: no
Program Req: btech
Dept Req: OTHER
Year Req: 2023
Salary: 550000
Opportu Status: open
Posted On: Thu, 30 Mar 2023 00:00:00 GMT
Deadline: Sun, 30 Apr 2023 00:00:00 GMT

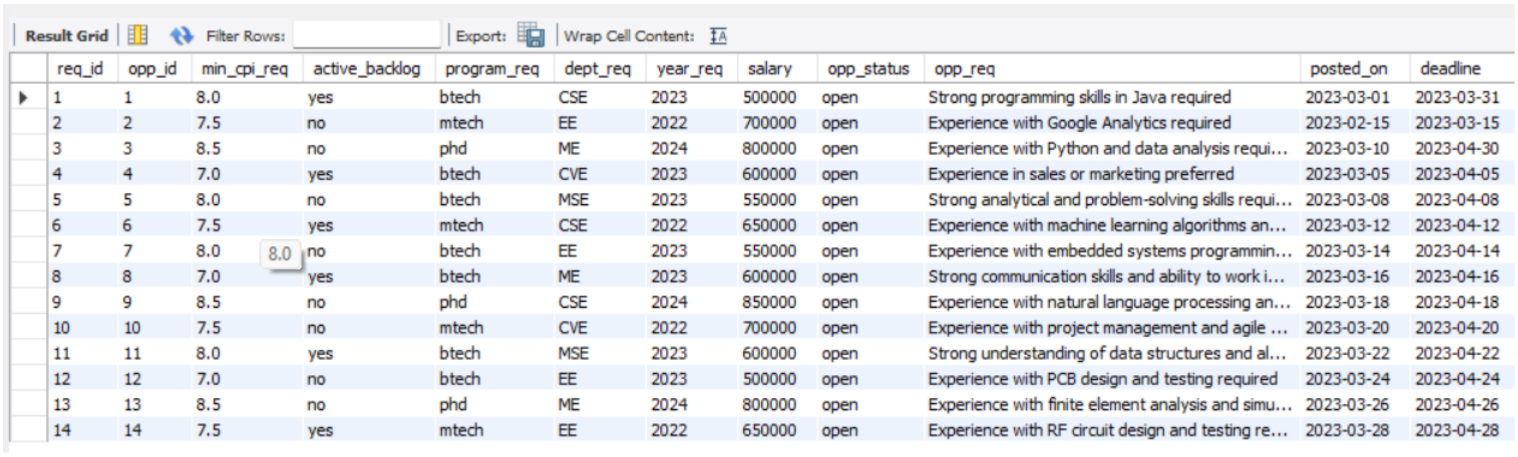
Delete

That particular opportunity gets deleted from the opportunity table, you can clearly see that below, there is no entry whose opp_id = 15.



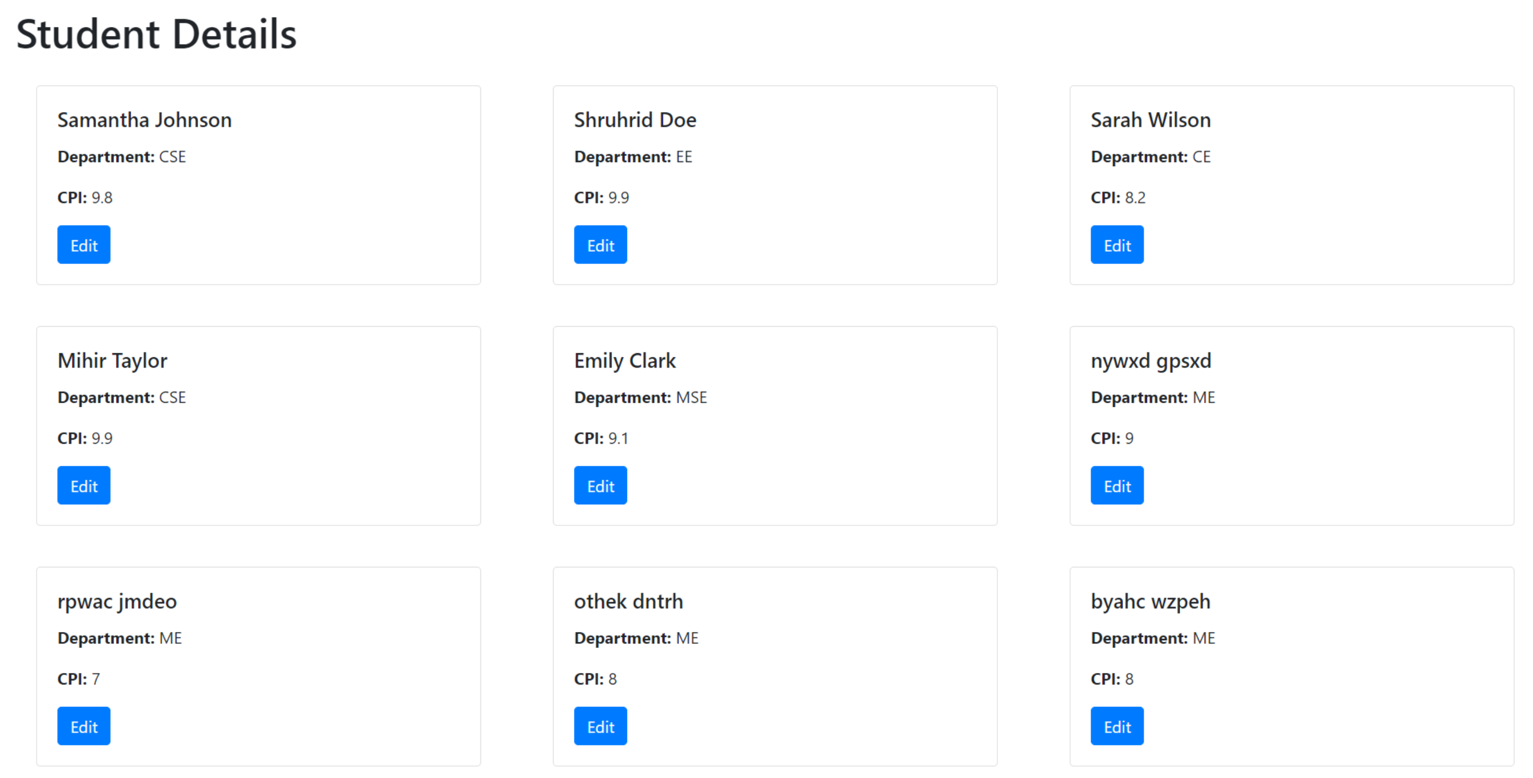
opp_id	opp_type	opp_title	address_line_1	address_line_2	address_line_3	company_id	poc_email_id
1	internship	Software Engineering Intern	123 Main St.			1	sahil099999@gmail.com
2	placement	Sales Representative	246 Maple Rd.			1	sahil.agrawal@iitgn.ac.in
3	internship	Software Engineering Intern	123 Main St.		sahil099999@gmail.com	1	NULL
4	placement	Marketing Manager	456 Oak Ave.			2	NULL
5	internship	Data Science Intern	789 Elm St.			3	NULL
6	placement	Sales Representative	246 Maple Rd.			4	NULL
7	internship	Product Management Intern	135 Cedar Ln.			5	NULL
8	internship	Software Engineering Internship	1 Infinite Loop	Cupertino	California	6	NULL
9	placement	Hardware Engineer Placement	One Microsoft Way	Redmond	Washington	7	NULL
10	internship	Data Science Internship	1600 Amphitheatre Parkway	Mountain View	California	8	NULL
11	placement	Marketing Placement	3500 Deer Creek Road	Palo Alto	California	9	NULL
12	internship	Software Development Internship	410 Terry Avenue North	Seattle	Washington	1	NULL
13	placement	Product Manager Placement	1 Infinite Loop	Cupertino	California	2	NULL
14	internship	Data Analyst Internship	One Microsoft Way	Redmond	Washington	2	NULL
16	internship	Mobile App Development Intern...	3500 Deer Creek Road	Palo Alto	California	4	NULL
17	placement	Business Development Placement	410 Terry Avenue North	Seattle	Washington	5	NULL

The requirements corresponding to opp_id = 15 also get deleted. Hence the constraints are satisfied.



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

req_id	opp_id	min_cpi_req	active_backlog	program_req	dept_req	year_req	salary	opp_status	opp_req	posted_on	deadline
1	1	8.0	yes	btech	CSE	2023	500000	open	Strong programming skills in Java required	2023-03-01	2023-03-31
2	2	7.5	no	mtech	EE	2022	700000	open	Experience with Google Analytics required	2023-02-15	2023-03-15
3	3	8.5	no	phd	ME	2024	800000	open	Experience with Python and data analysis requi...	2023-03-10	2023-04-30
4	4	7.0	yes	btech	CVE	2023	600000	open	Experience in sales or marketing preferred	2023-03-05	2023-04-05
5	5	8.0	no	btech	MSE	2023	550000	open	Strong analytical and problem-solving skills requi...	2023-03-08	2023-04-08
6	6	7.5	yes	mtech	CSE	2022	650000	open	Experience with machine learning algorithms an...	2023-03-12	2023-04-12
7	7	8.0	no	btech	EE	2023	550000	open	Experience with embedded systems programmin...	2023-03-14	2023-04-14
8	8	7.0	yes	btech	ME	2023	600000	open	Strong communication skills and ability to work i...	2023-03-16	2023-04-16
9	9	8.5	no	phd	CSE	2024	850000	open	Experience with natural language processing an...	2023-03-18	2023-04-18
10	10	7.5	no	mtech	CVE	2022	700000	open	Experience with project management and agile ...	2023-03-20	2023-04-20
11	11	8.0	yes	btech	MSE	2023	600000	open	Strong understanding of data structures and al...	2023-03-22	2023-04-22
12	12	7.0	no	btech	EE	2023	500000	open	Experience with PCB design and testing required	2023-03-24	2023-04-24
13	13	8.5	no	phd	ME	2024	800000	open	Experience with finite element analysis and simu...	2023-03-26	2023-04-26
14	14	7.5	yes	mtech	EE	2022	650000	open	Experience with RF circuit design and testing re...	2023-03-28	2023-04-28


Student Details

Samantha Johnson
Department: CSE
CPI: 9.8

Shruhrid Doe
Department: EE
CPI: 9.9

Sarah Wilson
Department: CE
CPI: 8.2

Mihir Taylor
Department: CSE
CPI: 9.9

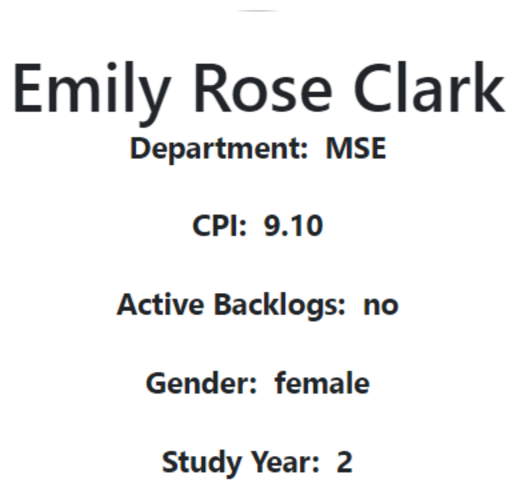
Emily Clark
Department: MSE
CPI: 9.1

nywxd gpsxd
Department: ME
CPI: 9

rpwac jmdeo
Department: ME
CPI: 7

othek dntrh
Department: ME
CPI: 8

byahc wzpeh
Department: ME
CPI: 8


Emily Rose Clark

Department: MSE

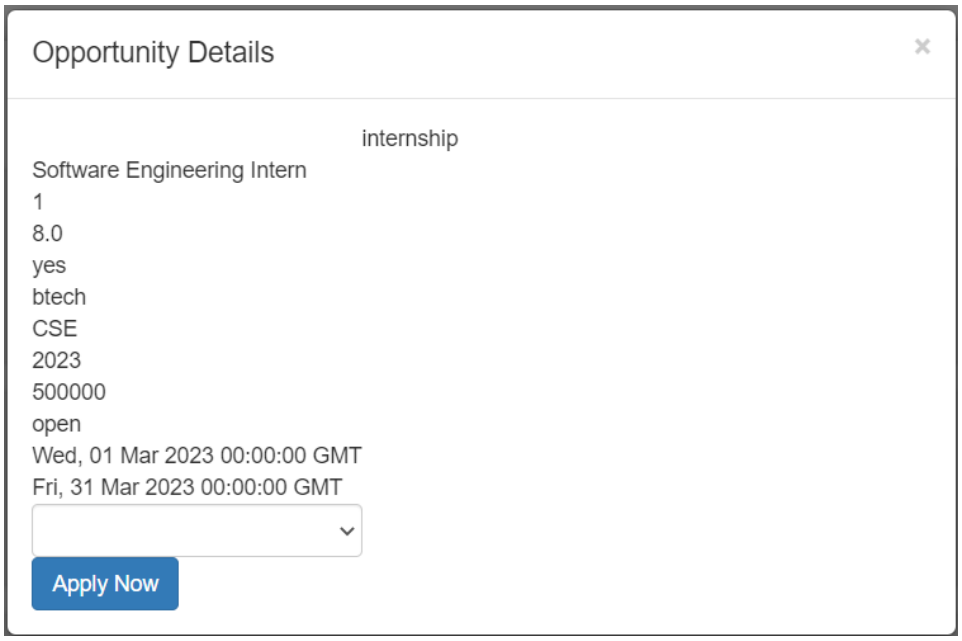
CPI: 9.10

Active Backlogs: no

Gender: female

Study Year: 2

Similarly, all the other constraints mensioned in assignment 1 are present in the website and database, and can be seen by interacting with the website.
Students can view their details but can’t edit it, whereas CDS can.
Similarly, CDS can view, add and delete opportunities, whereas students can only view the opportunities for applying.

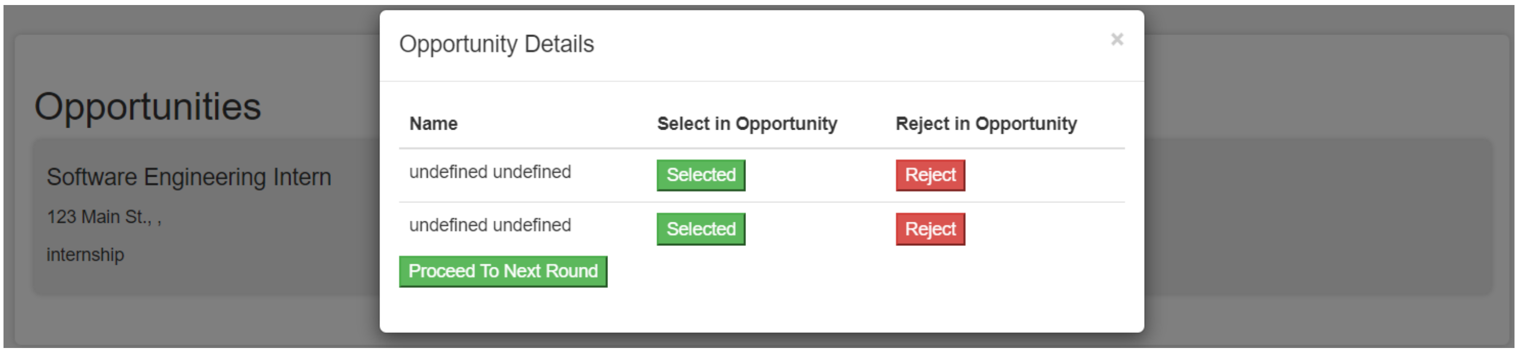

Opportunity Details

internship

Software Engineering Intern
1
8.0
yes
btech
CSE
2023
500000
open
Wed, 01 Mar 2023 00:00:00 GMT
Fri, 31 Mar 2023 00:00:00 GMT

Student view ⬆️

Similarly, CDS can view, add and delete opportunities whereas POC can only see their opportunities, and cannot add or delete, they only have permission to select / reject students.

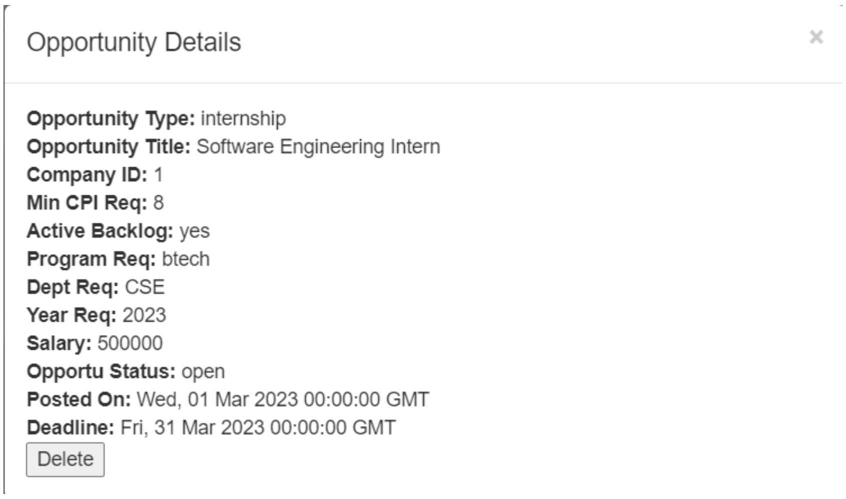

Opportunities

Software Engineering Intern
123 Main St., ,
internship

Opportunity Details

Name	Select in Opportunity	Reject in Opportunity
undefined undefined	Selected	Reject
undefined undefined	Selected	Reject

POC view ⬆️


Opportunity Details

Opportunity Type: internship
Opportunity Title: Software Engineering Intern
Company ID: 1
Min CPI Req: 8
Active Backlog: yes
Program Req: btech
Dept Req: CSE
Year Req: 2023
Salary: 500000
Opportu Status: open
Posted On: Wed, 01 Mar 2023 00:00:00 GMT
Deadline: Fri, 31 Mar 2023 00:00:00 GMT

CDS view ⬆️

Similarly, all the other privileges cane be seen from part of G1 Q2.

Responsibilities:

G1:

Mihir Sutariya (20110208)
- Made changes into the web application after the first feedback (round details page, login page, etc).
- Changed the point of contact selection page to make it more intuitive with added functionality.
- Made changes into the database as per requirements
- Wrote additional SQL queries to facilitate certain features

Pushpendra Pratap Singh (20110151)
- Made changes into the web application after the first feedback (round details page, login page, etc).
- Changed the point of contact selection page to make it more intuitive with added functionality.
- Documented the changes made after the first feedback from the stakeholders.
- Fixed UI bugs

Shruhrid Banthia (20110198)
- Made changes into the web application after the first feedback (round details page, login page, etc).
- Changed the point of contact selection page to make it more intuitive with added functionality.
- Made changes into the database as per requirements
- Wrote additional SQL queries to facilitate certain features

Sanskar Sharma (20110185)
- Made changes into the web application after the first feedback (round details page, login page, etc).
- Changed the point of contact selection page to make it more intuitive with added functionality.
- Documented the changes made after the first feedback from the stakeholders.
- Fixed UI bugs

Tanvi Dixit (20110212) (dropped)

Utkarsh Mishra (20110218)
- Documented the changes in the database as per the feedback received from stakeholders and documentation of the report.
- Documented the changes in the database as per the feedback received from stakeholders.

G2:

Dhyey Kumar Thummar (20110059)
- Documentation and screenshots of a total of 2 attacks [SQL Injection and XSS] performed and the defenses against those attacks.
- Documentation and screenshots of Additional attacks.

Ksheer Agrawal (20110098)
- Applied locks to the tables in SQL for handling concurrent multi user access, documented it well.
- Applied locks to the tables in SQL for handling concurrent multi user access.

R Yeeshu Dhurandhar (20110152)
- Applied locks to the tables in SQL for handling concurrent multi user access.

- 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.

Sahil Agrawal (20110178)

- Applied locks to the tables in SQL for handling concurrent multi user access.
- 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.

Saatvik Rao (20110175)

- Applied locks to the tables in SQL for handling concurrent multi user access, documented it well.
- Documented the changes in the database as per the feedback received from stakeholders.