

Indian Institute of Technology Gandhinagar



Alumni Database Management System - Assignment 4 By CSE Group

CS432 Project Report

11th April 2023

Authors

Dhairya Shah 20110054
Bhavesh Jain 20110038
Inderjeet Singh 20110080
Rahul Chembakasseril 20110158
Joy Makwana 20110086
Kalash Kankaria 20110088
Kanishk Singhal 20110091
Harshvardhan Vala 20110075
Nokzendi Aier 20110173
Medhansh Singh 20110111

Under the guidance of
Professor. Mayank Singh

Responsibility of Group 1

We met Mr. Rabadiya, who works in the Alumni Relations of IIT Gandhinagar. We got some great insights and improvements from an actual employee's view. Here is the feedback he gave after the 1st demo of our portal:

- 1) Add search and filter
- 2) Import and export CSV of data
- 3) Mailing feature
- 4) Dropdowns
- 5) Description of tables instead of schema visible to student role
- 6) Remove the right block in the student view because no operations are allowed for that role.

After getting the first feedback, we worked on the tips and implemented the below features, which turned out very useful and intuitive for the stakeholders. In the second feedback, we got **only** one more recommendation from the stakeholder: to implement bulk WhatsApp messaging to alumni from the database. Because we don't have a WhatsApp Business Account, we did not implement this feature due to technical difficulties.

The screenshots of the web application before the first feedback are already present in the README write-up of our GitHub repository. The screenshots containing updated features are present in this document.

We have successfully implemented all the plausible features and changes suggested by the stakeholders given in the initial and final feedback forms. We have written below the documentation of the updated features.

Please find the repository to our codebase here: https://github.com/RahulVC02/dbms_a3

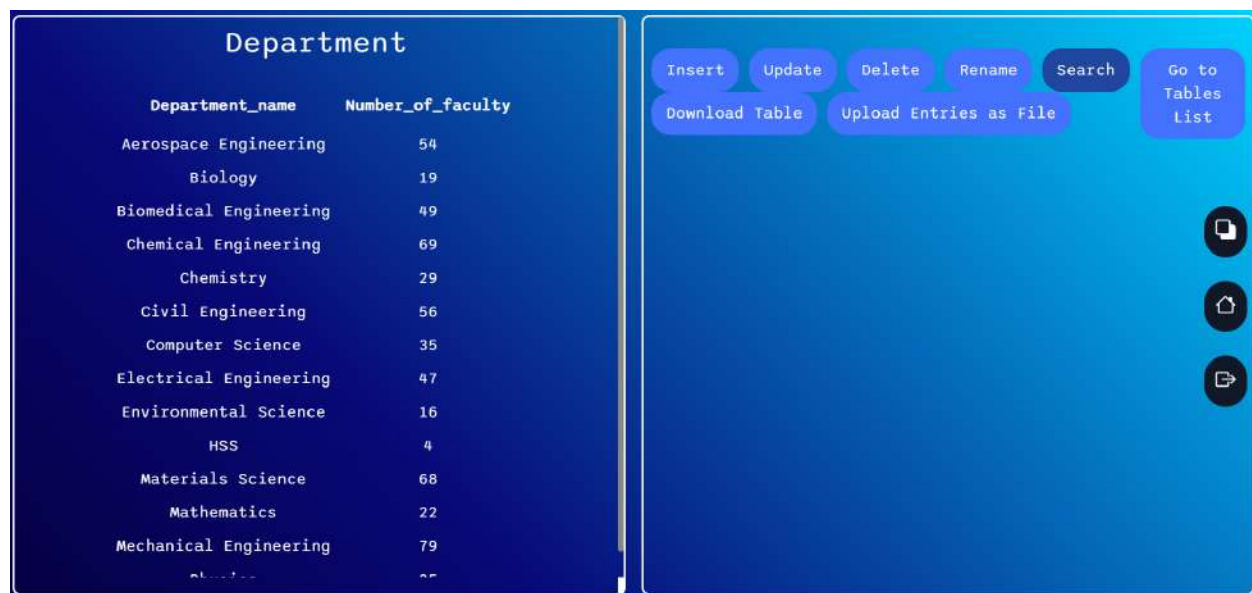
Documentation of features Implemented by Group-2:

1) Search

- Along with Insert, Delete, Update, and Rename, we added a feature to search in a table using keyword inputs. This provided users with an easy selection without writing queries in SQL. This is available to the admin and the employee roles.
- In the front end, this feature is a Search button and an input text bar- that outputs those tuples containing the specified keyword.
- In the backend, this feature first concatenates the fields in all the attributes of the tuples in a tuple. Then regular expression-based matching is done, and the matched tuples are returned as the output to the user.

- Screenshots

Interface



The screenshot displays a database management interface. On the left, a table titled 'Department' is shown with two columns: 'Department_name' and 'Number_of_faculty'. The table contains 15 rows of data. On the right, a sidebar contains several buttons for database operations: 'Insert', 'Update', 'Delete', 'Rename', 'Search', 'Go to Tables List', 'Download Table', and 'Upload Entries as File'. There are also three circular icons at the bottom of the sidebar: a document icon, a home icon, and a folder icon.

Department_name	Number_of_faculty
Aerospace Engineering	54
Biology	19
Biomedical Engineering	49
Chemical Engineering	69
Chemistry	29
Civil Engineering	56
Computer Science	35
Electrical Engineering	47
Environmental Science	16
HSS	4
Materials Science	68
Mathematics	22
Mechanical Engineering	79
Physics	25

Input - From the list of departments, we are searching for those departments which are related to "Engineering".

Department		Search the Table	
Department_name	Number_of_faculty	Engineering	Get Search Results
Aerospace Engineering	54		
Biology	19		
Biomedical Engineering	49		
Chemical Engineering	69		
Chemistry	29		
Civil Engineering	56		
Computer Science	35		
Electrical Engineering	47		
Environmental Science	16		
HSS	4		
Materials Science	68		
Mathematics	22		
Mechanical Engineering	79		

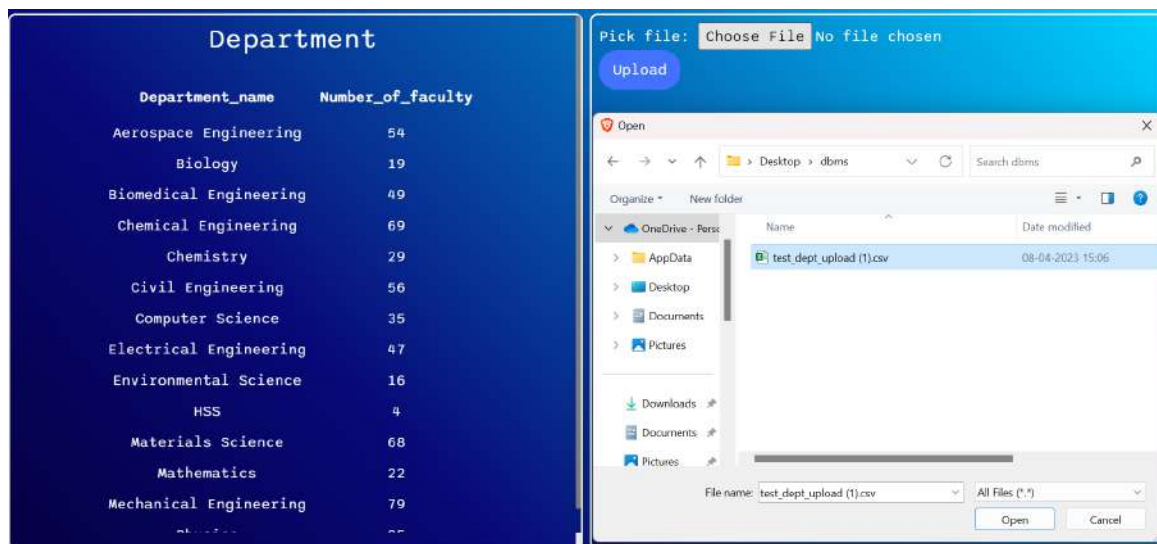
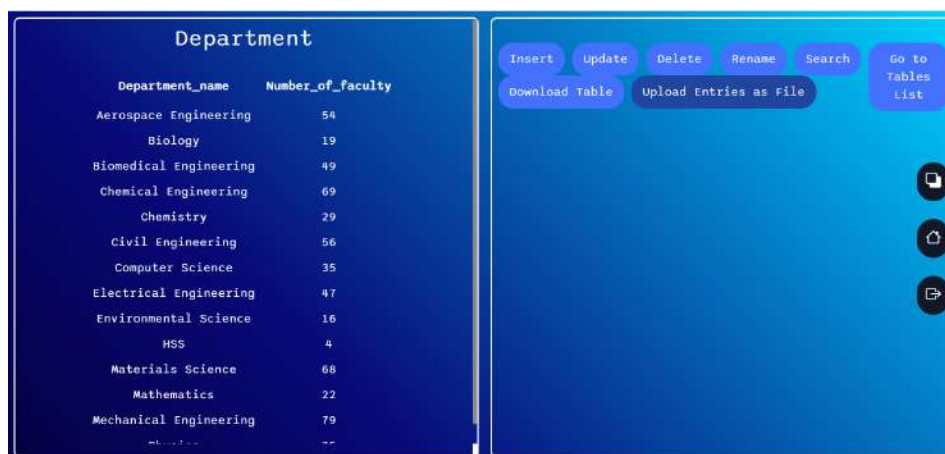
Output- The query outputs only those tuples which are related to "Engineering".

The Results For "Engineering" Are:			
Department_name	Number_of_faculty	Go to Tables List	
Aerospace Engineering	54	Go Back to department table	
Biomedical Engineering	49	Download Search Results	
Chemical Engineering	69		
Civil Engineering	56		
Electrical Engineering	47		
Mechanical Engineering	79		

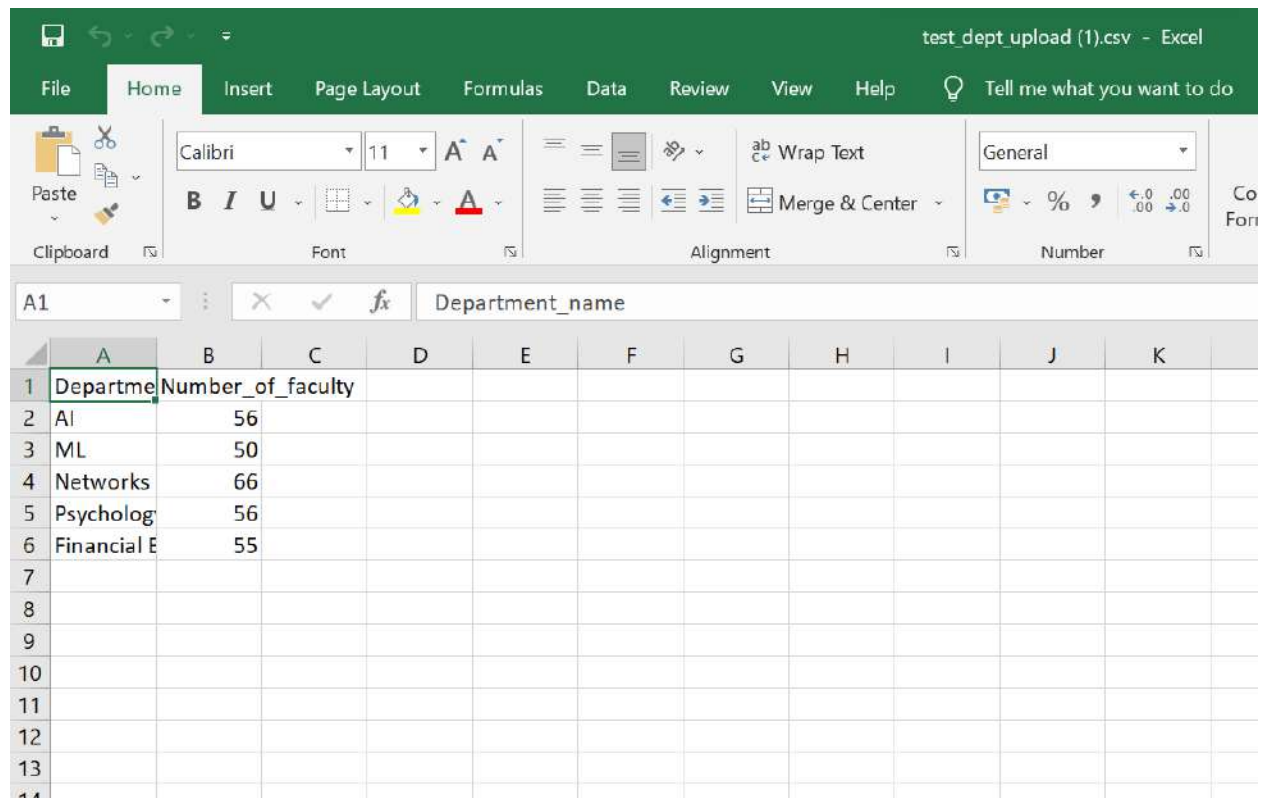
2) Upload

- In our interactions with the Alumni Relations office, they suggested that many times their data entry happens in bulk- using .csv files received from the Academic Office and Student Affairs office. They wanted a way to upload entries as csv files along with the original way of inserting entries one at a time.
 - In the frontend, we have provided an Upload button and an option to choose a file to upload. Post uploading, the user can execute and add each tuple to the relevant tuple.
 - In the backend, this works by reading the .csv file, converting it to a tabular form line-by-line and then uploading each row using an Insert operation.
- Screenshots

Interface and Input

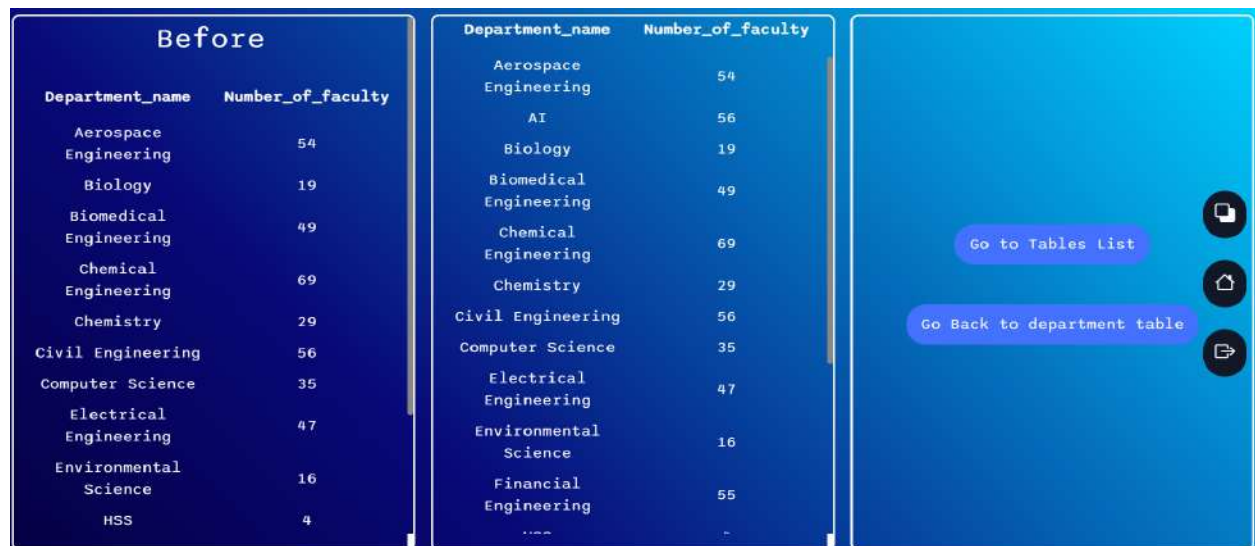


File View



	A	B	C	D	E	F	G	H	I	J	K
1	Departme	Number_of_faculty									
2	AI	56									
3	ML	50									
4	Networks	66									
5	Psychology	56									
6	Financial E	55									
7											
8											
9											
10											
11											
12											
13											
14											

Output - Observe that the "AI" and "Financial Engineering" departments can be seen in the "After" table and were not there in the "Before" table- the file's entries have been added to the database.



Before	
Department_name	Number_of_faculty
Aerospace Engineering	54
Biology	19
Biomedical Engineering	49
Chemical Engineering	69
Chemistry	29
Civil Engineering	56
Computer Science	35
Electrical Engineering	47
Environmental Science	16
HSS	4

Department_name	Number_of_faculty
Aerospace Engineering	54
AI	56
Biology	19
Biomedical Engineering	49
Chemical Engineering	69
Chemistry	29
Civil Engineering	56
Computer Science	35
Electrical Engineering	47
Environmental Science	16
Financial Engineering	55

[Go to Tables List](#)
[Go Back to department table](#)




Updated Table View

Aerospace Engineering	54
AI	56
Biology	19
Biomedical Engineering	49
Chemical Engineering	69
Chemistry	29
Civil Engineering	56
Computer Science	35
Electrical Engineering	47
Environmental Science	16
Financial Engineering	55
HSS	4
Materials Science	68
Mathematics	22
Mechanical Engineering	79
ML	50
Networks	66

InsertUpdateDeleteRenameSearch

Download TableUpload Entries as File

Go to Tables List



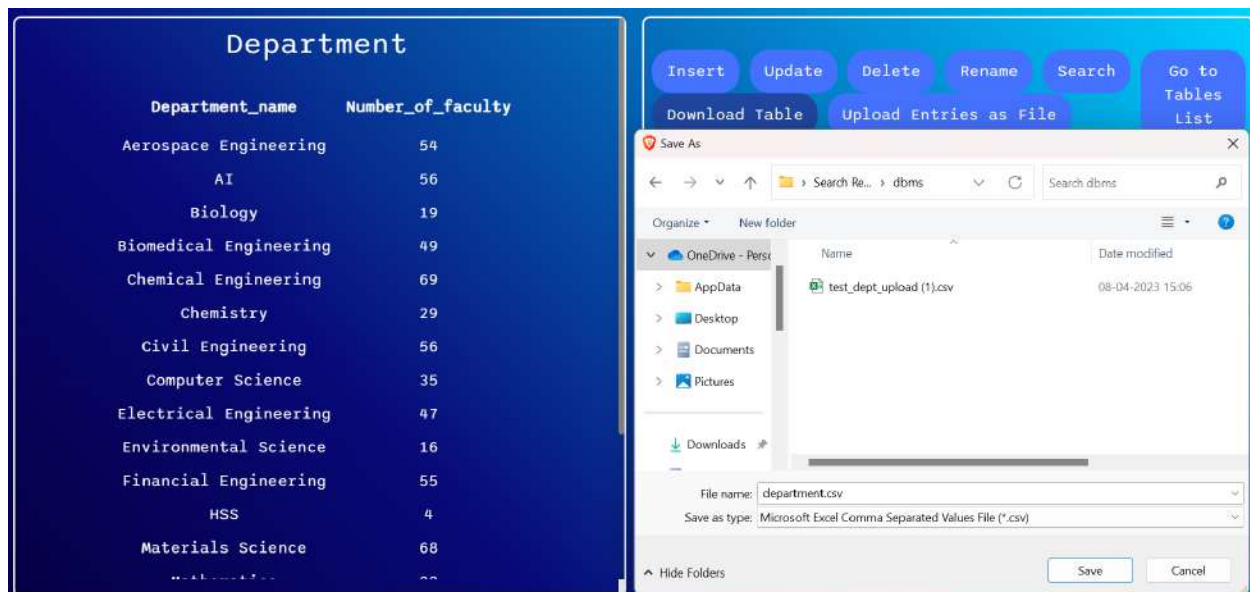
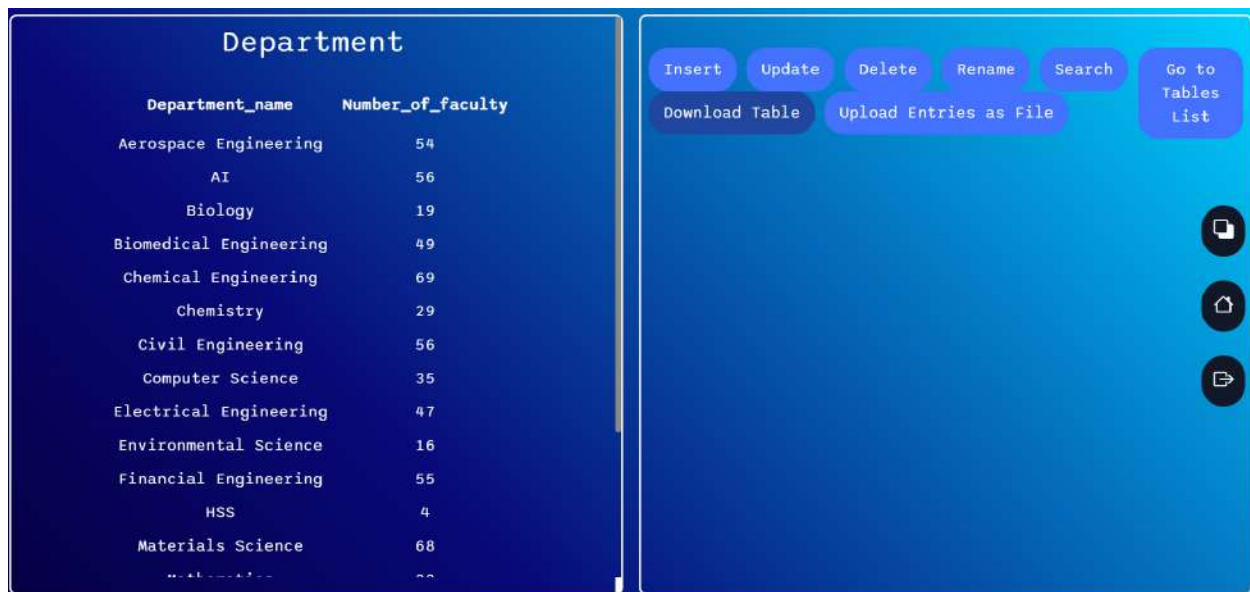
3) Download

- We also received feedback from the Alumni Relations team that they have use for data in .csv files, and so the web-app should have an option to export data from the database as .csv files. In the initial implementation, this feature was not there and we added it post this feedback.
- This download feature works for both- downloading all the data from a table as well as performing a search operation and then downloading the output data of the search operation.
- In the frontend, we provide a Download button to the Employee and Admin roles only. Along with the output of a search query, we provide a button to download the output as a .csv file.

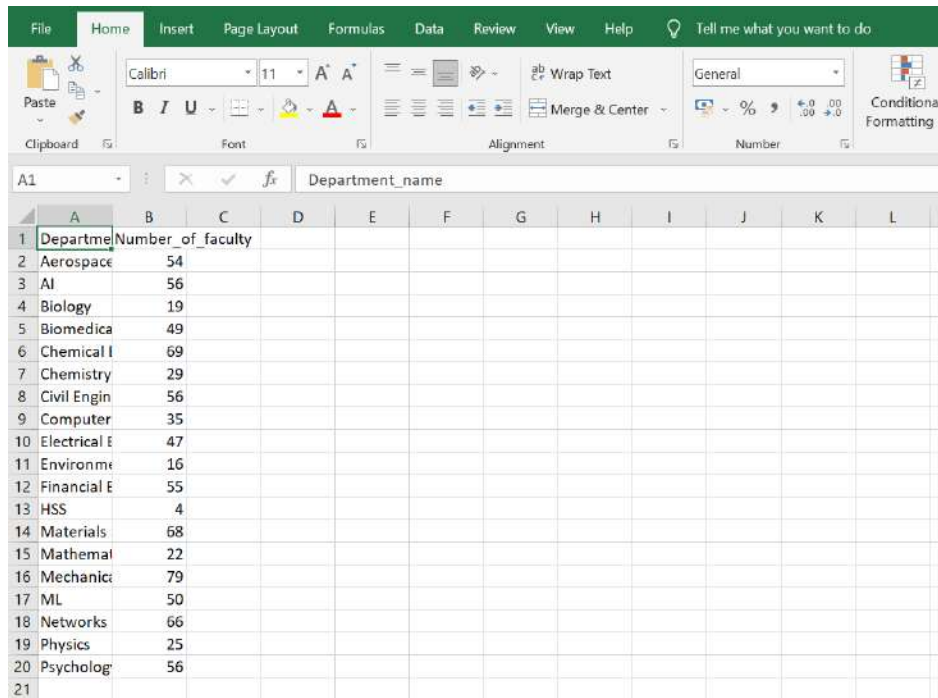
- In the backend, this works by fetching the data from the database using SELECT operations, then converting it into a csv file- row-by-row. Then the file object is returned to the user as a Response.

- Screenshots

Interface



Output File View



The screenshot shows the Microsoft Excel interface with the 'Home' tab selected. The active cell is A1, which contains the text 'Department_name'. The table data is as follows:

	A	B	C	D	E	F	G	H	I	J	K	L
1	Department_name	Number_of_faculty										
2	Aerospace	54										
3	AI	56										
4	Biology	19										
5	Biomedical	49										
6	Chemical E	69										
7	Chemistry	29										
8	Civil Engin	56										
9	Computer	35										
10	Electrical E	47										
11	Environme	16										
12	Financial E	55										
13	HSS	4										
14	Materials	68										
15	Mathemat	22										
16	Mechanics	79										
17	ML	50										
18	Networks	66										
19	Physics	25										
20	Psycholog	56										
21												

Download for Search Results

Input



The screenshot shows a web application interface with a dark blue background. On the left, there is a table titled 'Department' with two columns: 'Department_name' and 'Number_of_faculty'. The table contains the same data as the Excel screenshot. On the right, there is a search interface with a text input field labeled 'Search the Table' containing the text 'Chem'. Below the input field is a button labeled 'Get Search Results'. On the far right, there are three circular icons: a document, a home icon, and a folder.

Department_name	Number_of_faculty
Aerospace Engineering	54
AI	56
Biology	19
Biomedical Engineering	49
Chemical Engineering	69
Chemistry	29
Civil Engineering	56
Computer Science	35
Electrical Engineering	47
Environmental Science	16
Financial Engineering	55
HSS	4
Materials Science	68
Mathematics	22
Mechanics	79
ML	50
Networks	66
Physics	25
Psychology	56

Output




The Results For "Chem" Are:

Department_name	Number_of_faculty
Chemical Engineering	69
Chemistry	29

Go to Tables List

Go Back to department table

Download Search Results



Output File View




The Results For "Chem" Are:

Department_name	Number_of_faculty
Chemical Engineering	69
Chemistry	29

Go to Tables List

Go Back to department table

Download Search Results



Save As

Desktop > dbms

Search dbms

Organize New folder

Name	Date modified
department.csv	08-04-2023 15:30
test_dept_upload (1).csv	08-04-2023 15:06

File name: department_Chem.csv

Save as type: Microsoft Excel Comma Separated Values File (*.csv)

Hide Folders Save Cancel

	A	B	C	D	E	F	G	H	I	J	K
1	Department	Number_of_faculty									
2	Chemical I	69									
3	Chemistry	29									
4											
5											
6											
7											
8											
9											

4) Mailing

- In our discussions with the team, they also mentioned that a very common use-case for them is to filter alumni based on some common factor and then mass-mail these alumni for upcoming events/donations for college-related activities.
- So, we added a mailing feature specifically for the alumni table where a user can search for alumni using the previously implemented search feature. Then, the user gets an option to mail the alumni and a mailing list is displayed.
- The user also gets an option to add/remove people from the mailing list as many times as needed.
- The user gets text inputs for writing the subject as well as the body of the email and finally, a send button to dispatch the emails to everyone on the mailing list.
- In the frontend, we provide the above mentioned features.
- In the backend, we have set up a temporary email account and using Google App Passwords, we successfully integrated the Flask Mail SMTP service to this account. This can then be updated to the official email of IITGN Alumni Relations. The server receives the subject and the body of the mail, and the mailing list is stored on the server-side and updated as per user-inputs.
- The mailing list is then passed as the recipients list to the flask-mail API and the mails are dispatched.
- Example - Say we want to email all the alumni living in Mumbai for an upcoming Reunion event planned in Mumbai. We can do this using our web-app.

- Full Demo




Search

Alumni

Roll_Number	Full_Name	CPI	Contact_Number
10096079	Gregory Nelson	4.16	9464948451148
10165745	Lindsey Williams	6.78	0237554497245
10223578	Mark Sharp	0.85	6851779727576
10311874	Lindsey Martin	3.27	9862531380932
10316544	Jeremy Drake	3.60	4287307993068
10447530	David Morales	2.87	1710086566772
10459620	Craig Mata	6.00	6569529974445

Search the Table

[Get Search Results](#)

Mail Option

The Results For "Mumbai" Are:

Roll_Number	Full_Name	CPI	Contact_
14869225	Rahul Chembakasseril	4.00	1.23E
14869226	Kanishk Singhal	4.00	1.23E
14869227	Dhairya Shah	4.00	1.23E

[Go to Tables List](#)

[Go Back to alumni table](#)

[Download Search Results](#)

[Mail These Alumni](#)





Initial Mailing List

Mailing List




Name	Email
Rahul Chembakasseril	rahul.chembakasseril@iitgn.ac.in
Kanishk Singhal	kanishk.singhal@iitgn.ac.in
Dhairya Shah	dhairya.shah@iitgn.ac.in

Add Remove

Subject

Body

Send The Mail



Add

Mailing List

Name	Email
Rahul Chembakasseril	rahul.chembakasseril@iitgn.ac.in
Kanishk Singhal	kanishk.singhal@iitgn.ac.in
Dhairya Shah	dhairya.shah@iitgn.ac.in
Bhavesh Jain	bhavesh.jain@iitgn.ac.in




Add Remove

Alumnus Name

Alumnus Email

Execute

Go Back to Mail Page



Remove

Mailing List

Name	Email
Rahul Chembakasseril	rahul.chembakasseril@iitgn.ac.in
Kanishk Singhal	kanishk.singhal@iitgn.ac.in
Dhairya Shah	dhairya.shah@iitgn.ac.in
Bhavesh Jain	bhavesh.jain@iitgn.ac.in




Add Remove

Kanishk Singhal

Kanishk.singhal@iitgn

Execute

Go Back to Mail Page



Mailing List

Name	Email
Rahul Chembakasseril	rahul.chembakasseril@iitgn.ac.in
Dhairya Shah	dhairya.shah@iitgn.ac.in
Bhaves Jain	bhaves.jain@iitgn.ac.in

Add Remove

Subject

Body

Send The Mail

Send Mail

Mailing List

Name	Email
Rahul Chembakasseril	rahul.chembakasseril@iitgn.ac.in
Dhairya Shah	dhairya.shah@iitgn.ac.in
Bhaves Jain	bhaves.jain@iitgn.ac.in

Add Remove

Alumni Reunion in Mumbai | 10th April, 2023.

You're invited!

Send The Mail

Output on Mail Client



5) Dropdowns

- On the basis of our discussion with the team, they told us that writing SQL queries would be difficult for the team, so implementing drop-downs makes the attribute selection much more user-friendly. So, we added this feature for the Employee role of the database.
- Two dropdowns are added to delete entries from the database, one to select the attribute and the other to select the value of the selected attribute.
- In the frontend, we added the dropdowns for deleting an entry from the database for Employee role only.
- In the backend, we used javascript to implement the dropdowns because there are 2 dropdowns. The first dropdown is used to select the particular attribute from the table and the second dropdown is used to select the particular entry of that attribute. So, we used javascript which waits till the value of the first dropdown is selected and then appends the entries to the second dropdown on the basis of the first dropdown.
- Screenshots

Interface

The screenshot displays a web application interface for database management. It is divided into two main sections: a data table on the left and a query execution panel on the right.

Left Panel: Department Table

Department_name	Number_of_faculty
Aerospace Engineering	54
AI	56
Biology	19
Biomedical Engineering	49
Chemical Engineering	69
Chemistry	29
Civil Engineering	56
Computer Science	35
Electrical Engineering	47
Environmental Science	16
Financial Engineering	55
HSS	4
Materials Science	68

Right Panel: SQL Query Builder

The right panel is titled "DELETE FROM department" and "WHERE". It contains a "Select an attribute" dropdown menu. The dropdown is currently open, showing a list of attributes: "Department_name", "Department_name", and "Number_of_faculty". The first "Department_name" is selected. Below the dropdown, there is a text input field containing "Aerospace Engineering". To the right of the input field is a small icon of a document with a checkmark. Below the input field is a blue "Execute" button. To the right of the button are three circular icons: a home icon, a lock icon, and a refresh icon.

Input - Deleting the "AI" entry

Department

Department_name	Number_of_faculty
Aerospace Engineering	54
AI	56
Biology	19
Biomedical Engineering	49
Chemical Engineering	69
Chemistry	29
Civil Engineering	56
Computer Science	35
Electrical Engineering	47
Environmental Science	16
Financial Engineering	55
HSS	4
Materials Science	68

```
DELETE FROM department
WHERE
```

Select an attribute

Department_name

Select the value

Aerospace Engineering

Aerospace Engineering

AI

Biology

Biomedical Engineering

Chemical Engineering

Chemistry

Civil Engineering

Computer Science

Electrical Engineering

Environmental Science

Financial Engineering

HSS

Materials Science

Mathematics

Mechanical Engineering

ML

Networks

Physics

Psychology

Output - The "AI" entry is deleted.

Before

Department_name	Number_of_faculty
Aerospace Engineering	54
AI	56
Biology	19
Biomedical Engineering	49
Chemical Engineering	69
Chemistry	29
Civil Engineering	56
Computer Science	35
Electrical Engineering	47
Environmental Science	16

After

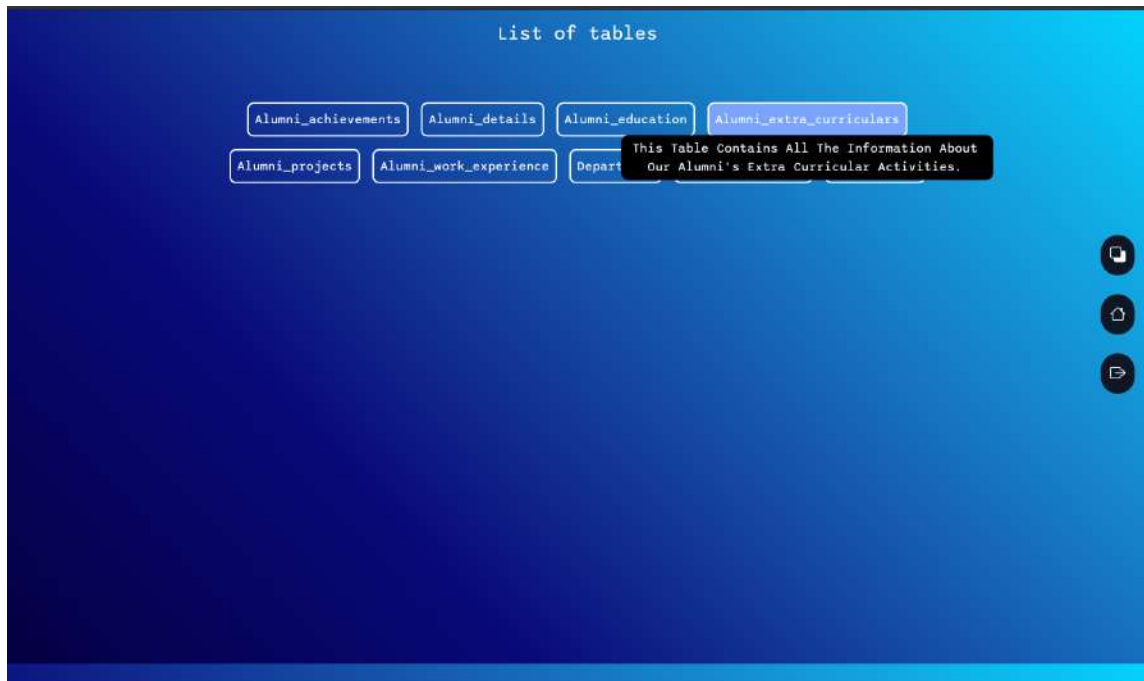
Department_name	Number_of_faculty
Aerospace Engineering	54
Biology	19
Biomedical Engineering	49
Chemical Engineering	69
Chemistry	29
Civil Engineering	56
Computer Science	35
Electrical Engineering	47
Environmental Science	16
Financial	55

Go to Tables List

Go Back to department table

6) UI/UX Changes

- As per the feedback, the schema details are no more visible to a user with 'student' role. Instead, a preview of a small description of the table is seen when the cursor is hovered over the table name in the 'List of Tables'
- For the employee and admin roles, hovering the cursor on the table name still shows the schema details of that table.



- Also, for student role, since none of the actions like 'Insert', 'Update', and 'Delete' were available to the user, the right partition of the page, which was empty before, was removed to give a single display of the respective table. Like:

The screenshot shows the 'Alumni_details' table view with a dark blue background. The table has the following columns: Roll_Number, Full_Name, Contact_Number, Branch, Degree, and an unnamed column. The data is as follows:

Roll_Number	Full_Name	Contact_Number	Branch	Degree	
10096679	Gregory Nelson	9464948451148	Environmental Science	MTech	xrusse
10165745	Lindsey Williams	0237554497245	Chemistry	BTech	rosarIode
10223578	Mark Sharp	6851779727576	Environmental Science	PHD	joannaza
10311874	Lindsey Martin	9862531380932	Biology	MTech	roymurr
10310544	Jeremy Drake	4287387993068	Chemical Engineering	PHD	mollynel
10447530	David Morales	1710086566772	Civil Engineering	BTech	solomonvict
10459620	Craig Mata	6569529974445	Environmental Science	BTech	marycoler
10785670	Anne Watkins	8624011058255	Computer Science	MA	kelsey
10785105	Michael Wilson	5222844420854	Electrical Engineering	MSc	kelsey
10795722	Troy Lucas III	1213758010884	Materials Science	PHD	lburch
			Chemical		

On the right side, there are three circular icons: a document, a home, and a list.

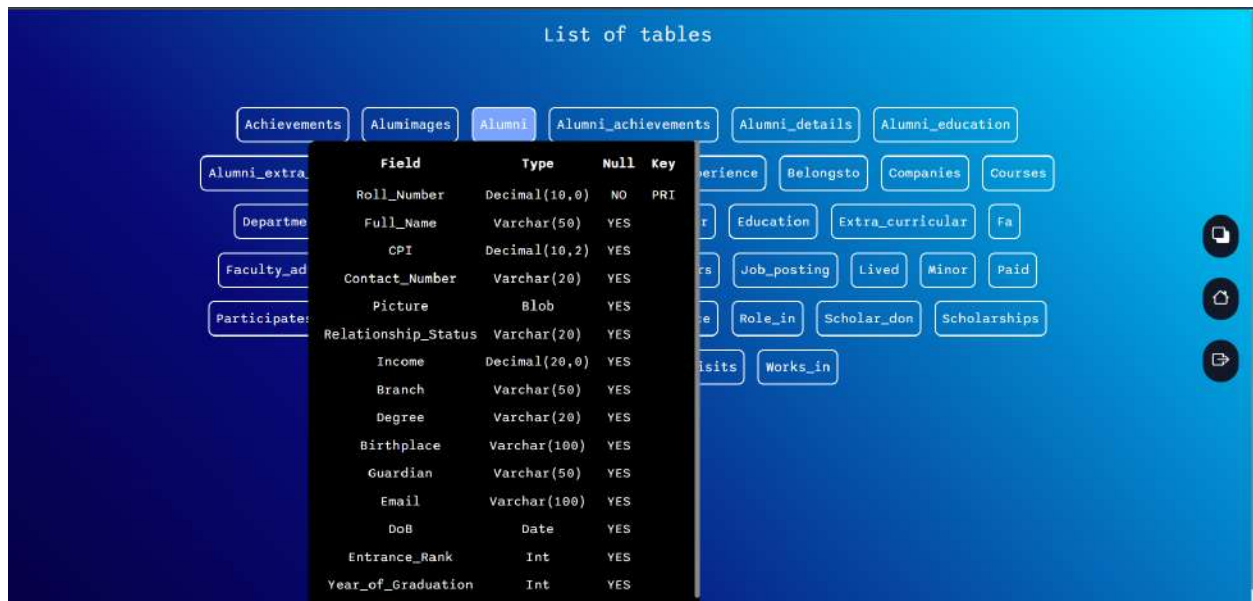
Screenshots of different views of the database seen by different users

Admin

List of tables for admin.



Schema: Admin can view the schema of all the tables by hovering.



Operations: Admin can insert, update and delete entries in any table. Rename is only available for the admin role. Search, download table, and upload entries as files are available for admin and employee.

Instructor_ID	Instructor_Name	Work_email
1008	Kristin Martinez	catherine27@example.net
1009	Amanda Martinez	kgilmore@example.net
1017	Leonard Rodriguez	russellemily@example.net
1028	Christopher Garrison	malonestephanie@example.com
1039	Jessica Blake	sharon37@example.org
1044	Nicole Roberts	andersonheather@example.com
1046	Stephanie Wilson	jessica28@example.net
1048	Jennifer Keith	rhayes@example.org
1053	James Hebert	jfowler@example.net
1063	Richard Adams	debra98@example.org
1065	Mackenzie Padilla	gwilliams@example.org
1070	Jennifer Torres	vsmith@example.net
1088	Benjamin Lucero	williamsleslie@example.net
1092	Christopher Smith	brian73@example.com
1096	Mr. Johnathan Gay	nicole29@example.net
1101	Laura Lewis	latoya33@example.com
1130	Ashley Benitez	hollowaycarmen@example.com

Emailing Alumni: Admin can bulk email to the alumni by filtering from the alumni table. Employee can also do the same but student is not allowed to do so.

Search

Roll_Number	Full_Name	CPI	Contact_Number
10096079	Gregory Nelson	4.16	9464948451148
10165745	Lindsey Williams	6.78	0237554497245
10223578	Mark Sharp	0.85	6851779727576
10311874	Lindsey Martin	3.27	9862531380932
10316544	Jeremy Drake	3.60	4287307993068
10447530	David Morales	2.87	1710086566772
10459620	Craig Mata	6.00	6569529974445

Mail Option

The Results For "Mumbai" Are:

Roll_Number	Full_Name	CPI	Contact_
14869225	Rahul Chembakasseril	4.00	1.23E
14869226	Kanishk Singhal	4.00	1.23E
14869227	Dhairya Shah	4.00	1.23E

Go to Tables List

Go Back to alumni table

Download Search Results

Mail These Alumni



Initial Mailing List

Mailing List




Name	Email
Rahul Chembakasseril	rahul.chembakasseril@iitgn.ac.in
Kanishk Singhal	kanishk.singhal@iitgn.ac.in
Dhairya Shah	dhairya.shah@iitgn.ac.in

Add Remove

Subject

Body

Send The Mail



Output on Mail Client

Alumni Reunion in Mumbai | 10th April, 2023. External Inbox x 🔗 🖨 📧

 alumni.dbms.g2@gmail.com 4:01PM (0 minutes ago) ★ ↩ ⋮

to dhairya.shah, bhaveshjain, me ▾

You're invited!

↩ Reply ↩ Reply all ➡ Forward 🔗 Get link

Employee

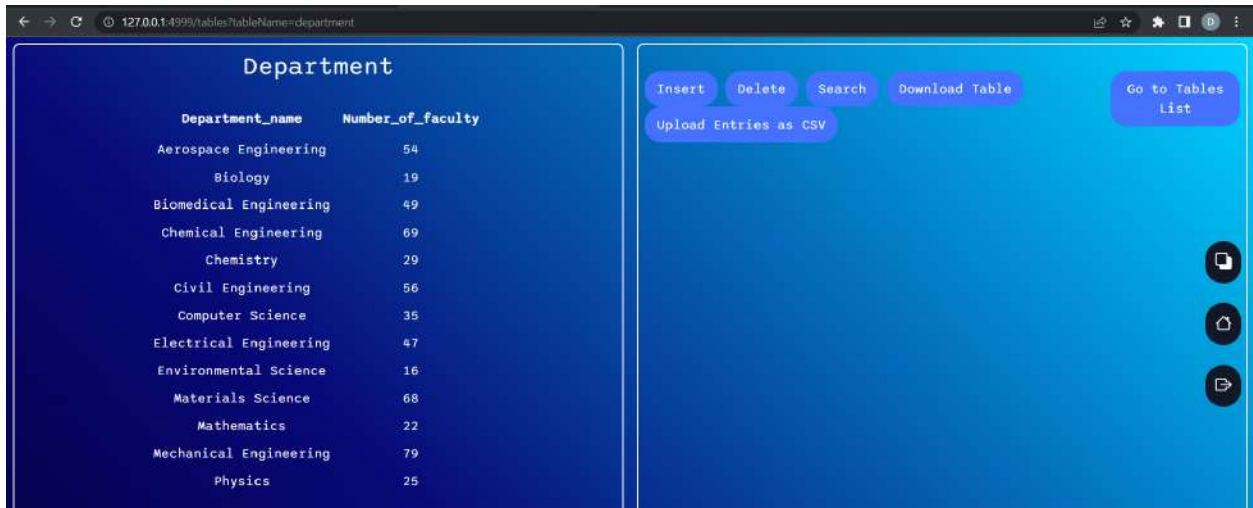
List of tables in employee role



Schema: Employee can see the schema of all the tables by hovering.



Operations: Employee can insert, delete, and search entries in the tables. Employee can also download tables, and upload entries as CSV. Rename is not allowed for employee role, but employee can bulk email the alumni from the alumni table as mentioned earlier.



The screenshot shows a web application interface for a database table named "Department". The interface is divided into two main sections. The left section displays a table with two columns: "Department_name" and "Number_of_faculty". The right section contains several buttons for table operations: "Insert", "Delete", "Search", "Download Table", "Upload Entries as CSV", and "Go to Tables List". The browser's address bar shows the URL "127.0.0.1:4399/tables?tableName=department".

Department_name	Number_of_faculty
Aerospace Engineering	54
Biology	19
Biomedical Engineering	49
Chemical Engineering	69
Chemistry	29
Civil Engineering	56
Computer Science	35
Electrical Engineering	47
Environmental Science	16
Materials Science	68
Mathematics	22
Mechanical Engineering	79
Physics	25

Student

List of Tables in student role



Schemas: Student cannot see the schemas by hovering; instead a description of the table is displayed.



No operations are allowed for the student role except viewing the above tables.

Alumni_education					
Roll_Number	Student_Name	Institute_Name	Degree	Discipline	Institute_c
72958940	Frank Sanchez	Alyssa Allen	10	Art	Lake Rebeccashi
58340650	Stephanie Terry	Alyssa Allen	10	Art	Lake Rebeccashi
95408004	Scott Fisher	Alyssa Allen	10	Art	Lake Rebeccashi
58557967	Steve Guerrero	Alyssa Allen	10	Art	Lake Rebeccashi
19258380	Deborah Nelson	Amy Casey	10	Art	Andrewchest
81034393	Edwin Hardy	Amy Casey	10	Art	Andrewchest
12693066	Craig Graham	Amy Casey	10	Art	Andrewchest
38010511	Emily Gomez	Amy Casey	10	Art	Andrewchest
55852336	Lori Mejia	April Welch	10	Art	Colemanbur
70641576	Kathleen Smith	April Welch	10	Art	Colemanbur
64532126	Samantha Pham	April Welch	10	Art	Colemanbur
49954150	Sara Hicks	April Welch	10	Art	Colemanbur
63891713	Heather Drake	April Welch	10	Art	Colemanbur
83038765	Robert Smith	April Welch	10	Art	Colemanbur

Other Responsibilities of Group 2

Multiple user access with different roles

Here we needed to ensure that multiple users across different roles can access the database concurrently and retrieve the information they want without any issue. We also needed to ensure that no 2 users working on the database together can make changes to the same table as that might break the constraints of our database and cause other synchronization issues.

This problem was tackled by the in-built implementation of Flask itself. Flask handles the task of allowing multiple users to concurrently look at the database. Along with this, the error handling implemented in our web application from the start regarding database constraints, tackle this problem of 2 users editing the same item very smoothly.

Here is an example of multiple users accessing the database together:

Here we can see that we have 2 users with IPs - 10.7.35.65 and 10.7.51.107 who are logged in to our database hosted by one of those clients itself.

This is the terminal log of both users logging in to our database along with the time:

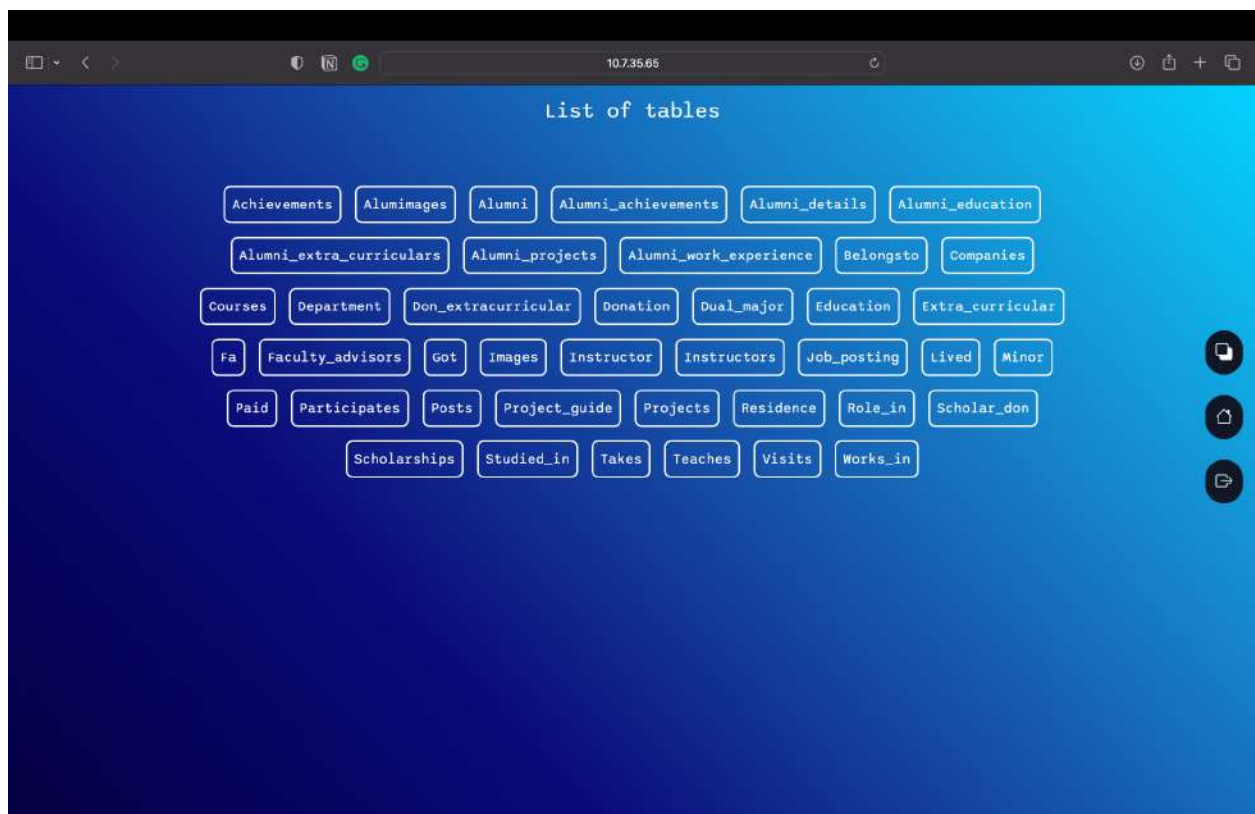
```
PS D:\College Work\Databases\Assignment-3\dbms_a3> python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSG
* Running on http://10.7.35.65:5000
Press CTRL+C to quit
* Restarting with watchdog (windowsapi)
* Debugger is active!
* Debugger PIN: 138-837-014
10.7.35.65 - - [14/Apr/2023 00:08:30] "GET / HTTP/1.1" 200 -
10.7.35.65 - - [14/Apr/2023 00:08:30] "GET /static/styles/style.css HTTP/1.1" 304 -
10.7.51.107 - - [14/Apr/2023 00:09:31] "GET / HTTP/1.1" 200 -
10.7.51.107 - - [14/Apr/2023 00:09:32] "GET /static/styles/style.css HTTP/1.1" 304 -
10.7.35.65 - - [14/Apr/2023 00:09:43] "POST / HTTP/1.1" 302 -
10.7.35.65 - - [14/Apr/2023 00:09:43] "GET /tables HTTP/1.1" 200 -
10.7.35.65 - - [14/Apr/2023 00:09:43] "GET /static/styles/style.css HTTP/1.1" 304 -
10.7.51.107 - - [14/Apr/2023 00:09:50] "POST / HTTP/1.1" 302 -
10.7.51.107 - - [14/Apr/2023 00:09:51] "GET /tables HTTP/1.1" 200 -
10.7.51.107 - - [14/Apr/2023 00:09:51] "GET /static/styles/style.css HTTP/1.1" 304 -
```

Here User-1 with IP 10.7.35.65 has logged in with an Employee role,
And User-2 with IP 10.7.51.107 has logged in with an Admin role.

Table view for User-1:



Table view for User-2:



Now, both Users will go to the "Department" table and try to Insert an entry in the table.

The terminal log shows that both the users enter the insert page of "Department" table around the same time:

```
10.7.35.65 - - [14/Apr/2023 00:15:37] "GET /tables?tableName=department HTTP/1.1" 200 -
10.7.35.65 - - [14/Apr/2023 00:15:37] "GET /static/styles/style.css HTTP/1.1" 304 -
10.7.35.65 - - [14/Apr/2023 00:15:37] "GET /static/group.png HTTP/1.1" 304 -
10.7.51.107 - - [14/Apr/2023 00:15:42] "GET /tables?tableName=department HTTP/1.1" 200 -
10.7.51.107 - - [14/Apr/2023 00:15:42] "GET /tables?tableName=department HTTP/1.1" 200 -
10.7.51.107 - - [14/Apr/2023 00:15:42] "GET /static/styles/style.css HTTP/1.1" 304 -
10.7.51.107 - - [14/Apr/2023 00:15:44] "POST /tables/edit HTTP/1.1" 200 -
10.7.51.107 - - [14/Apr/2023 00:15:44] "GET /static/styles/style.css HTTP/1.1" 304 -
10.7.35.65 - - [14/Apr/2023 00:15:44] "POST /tables/edit HTTP/1.1" 200 -
10.7.35.65 - - [14/Apr/2023 00:15:44] "GET /static/styles/style.css HTTP/1.1" 304 -
10.7.35.65 - - [14/Apr/2023 00:16:50] "POST /tables/edit/insert HTTP/1.1" 200 -
10.7.35.65 - - [14/Apr/2023 00:16:50] "GET /static/styles/style.css HTTP/1.1" 304 -
10.7.51.107 - - [14/Apr/2023 00:16:51] "POST /tables/edit/insert HTTP/1.1" 200 -
10.7.51.107 - - [14/Apr/2023 00:16:51] "GET /static/styles/style.css HTTP/1.1" 304 -
```

Now, both the Users will try to edit the department table around the same time.

Insert Page as seen by User-1:

Not secure | 10.7.35.65:5000/tables/edit

Department

Department_name	Number_of_faculty
Aerospace Engineering	54
Biology	19
Biomedical Engineering	49
Chemical Engineering	69
Chemistry	29
Civil Engineering	56
Computer Science	35
Electrical Engineering	47
Environmental Science	16
Materials Science	68
Mathematics	22
Mechanical Engineering	79
Physics	25

INSERT INTO department

Department_name

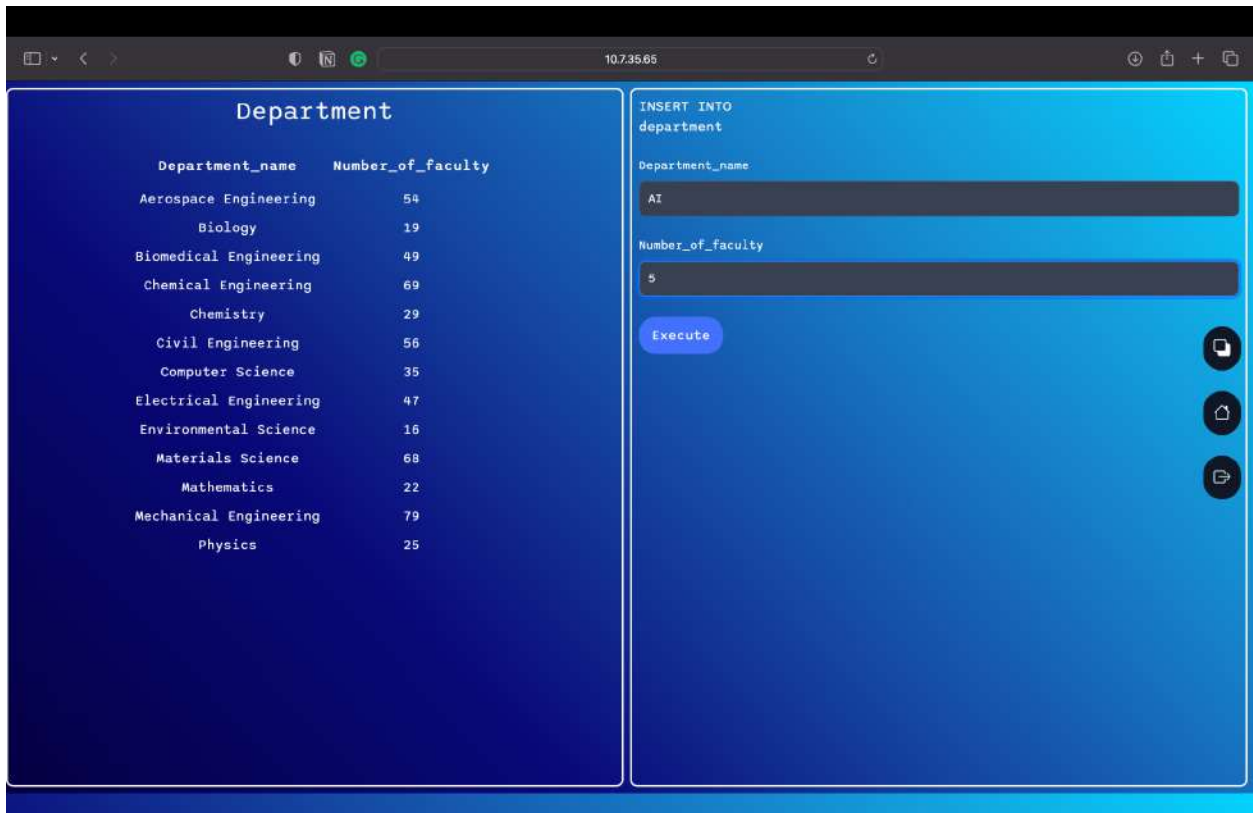
AI

Number_of_faculty

12

Execute

Insert Page as seen by User-2:



The screenshot shows a web application interface with a dark blue header and a light blue body. On the left, there is a table titled "Department" with two columns: "Department_name" and "Number_of_faculty". The table lists various departments and their corresponding faculty counts. On the right, there is a form titled "INSERT INTO department" with two input fields: "Department_name" and "Number_of_faculty". The "Department_name" field contains the text "AI" and the "Number_of_faculty" field contains the text "5". Below the input fields is a blue button labeled "Execute". To the right of the "Execute" button are three circular icons: a document, a home, and a refresh.

Department_name	Number_of_faculty
Aerospace Engineering	54
Biology	19
Biomedical Engineering	49
Chemical Engineering	69
Chemistry	29
Civil Engineering	56
Computer Science	35
Electrical Engineering	47
Environmental Science	16
Materials Science	68
Mathematics	22
Mechanical Engineering	79
Physics	25

INSERT INTO department

Department_name

AI

Number_of_faculty

5

Execute

We can see that both the Users can access the same table concurrently and view the information at the same time without any issue. The above terminal log and photos show that both the users are at the same table at the same time.

Now, User-1 and User-2 will simultaneously try to edit the same table with the entries seen above, but User-1 (Employee) executed the edit before User-2, and hence User-2 will not be able to execute his/her edit. This can be seen from what both the Users saw after they executed their Insert command.

For User-1:

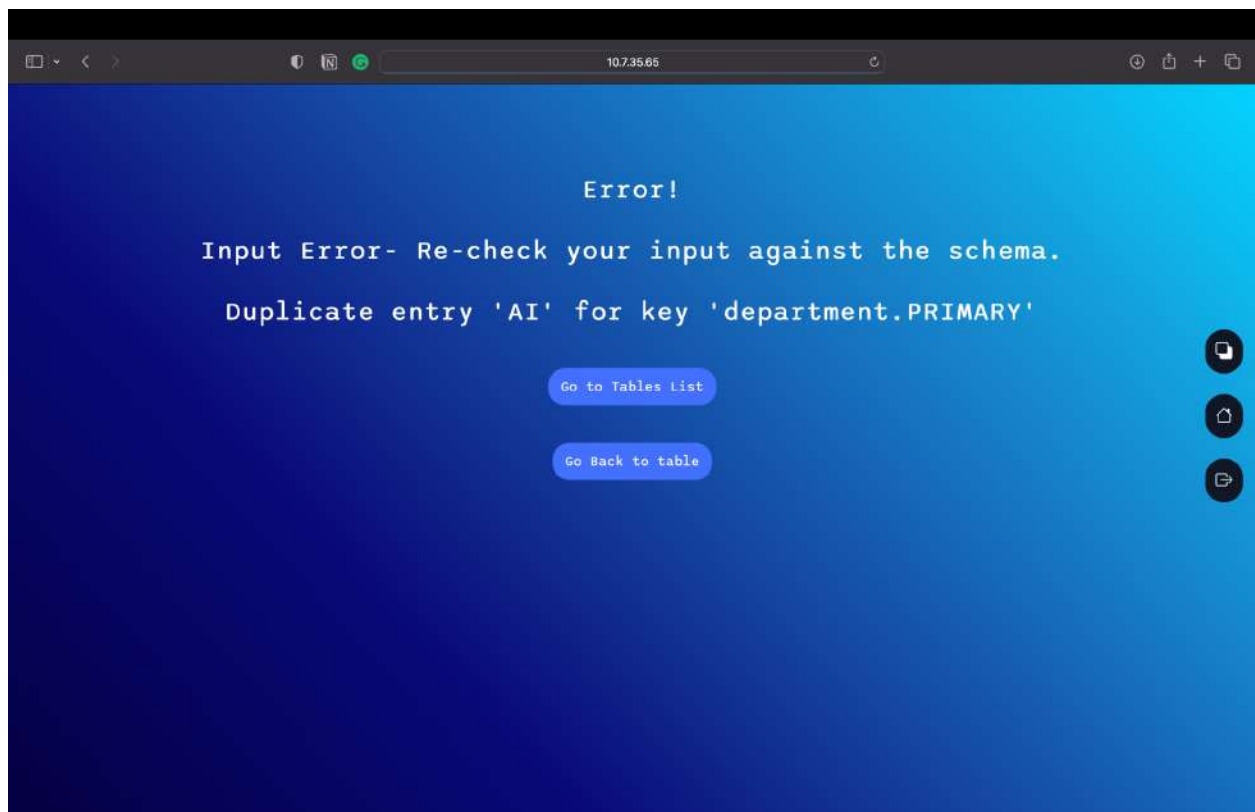


Before		After	
Department_name	Number_of_faculty	Department_name	Number_of_faculty
Aerospace Engineering	54	Aerospace Engineering	54
Biology	19	AI	12
Biomedical Engineering	49	Biology	19
Chemical Engineering	69	Biomedical Engineering	49
Chemistry	29	Chemical Engineering	69
Civil Engineering	56	Chemistry	29
Computer Science	35	Civil Engineering	56
Electrical Engineering	47	Computer Science	35
Environmental Science	16	Electrical Engineering	47
Materials Science	68	Environmental Science	16
Mathematics	22	Materials Science	68
Mechanical Engineering	79	Mathematics	22
Physics	25	Mechanical Engineering	79
		Physics	25

Go to Tables List

Go Back to department table

For User-2:



Error!

Input Error- Re-check your input against the schema.

Duplicate entry 'AI' for key 'department.PRIMARY'

Go to Tables List

Go Back to table

This shows that User-2 was not able to execute his/her edit on the same item. Hence, it can be seen that multiple users are able to work on the database concurrently, but they cannot update the same item together, as was required from our web application.

If the Users would have worked on/updated different items, then our application would give no error and hence they can do that concurrently. But, while editing the same item, only one of the Users could execute his/her edits as desired.

Responsibility of Group 1&2

Attacks

We have performed 4 attacks, namely, SQL Injection, XSS, stored XSS, and URL attack.

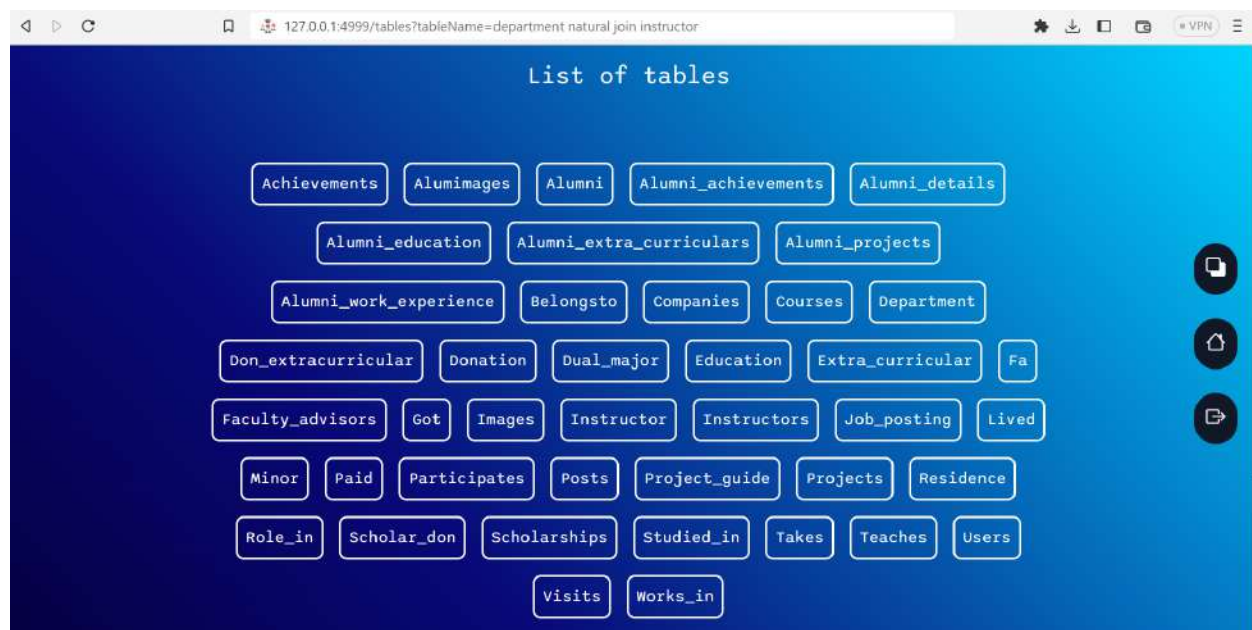
1) SQL Injection:

To perform an SQL injection, we need to insert some SQL code in an input which is not sanitized. If we can get the backend server to execute the “injected” SQL code despite the application not giving us explicit functionality to execute such an SQL code, then our SQL injection is successful.

Example 1

A user can perform a natural join and view the results of the natural join without the application even providing such a functionality. Here, I am injecting the tableName value in the URL with “department natural join instructor” instead of a table name such as “alumni” or “department” or “instructor”. This gives the user the ability to perform a natural join using SQL injection when the application doesn’t provide the explicit functionality for this.

Input



Output

Department Natural Join Instructor			
Psychology	56	1008	Kristin Martinez catherine27@example.net
Physics	25	1008	Kristin Martinez catherine27@example.net
Networks	66	1008	Kristin Martinez catherine27@example.net
ML	50	1008	Kristin Martinez catherine27@example.net
Mechanical Engineering	79	1008	Kristin Martinez catherine27@example.net
Mathematics	22	1008	Kristin Martinez catherine27@example.net
Materials Science	68	1008	Kristin Martinez catherine27@example.net
Financial Engineering	55	1008	Kristin Martinez catherine27@example.net

This is the output of the SQL query of “natural join” manually injected in the URL bar which is not typical behavior expected of a user. The output corresponds to the natural join of the department and the instructor table.

Example 2

The system we’ve built has multiple users- several admins, employees and students who have usernames and passwords which are validated during login time. These credentials are stored in a separate “users” database. All our other data is stored in the “alumni” database and we only display the tables and views from this “alumni” database inside our web application.

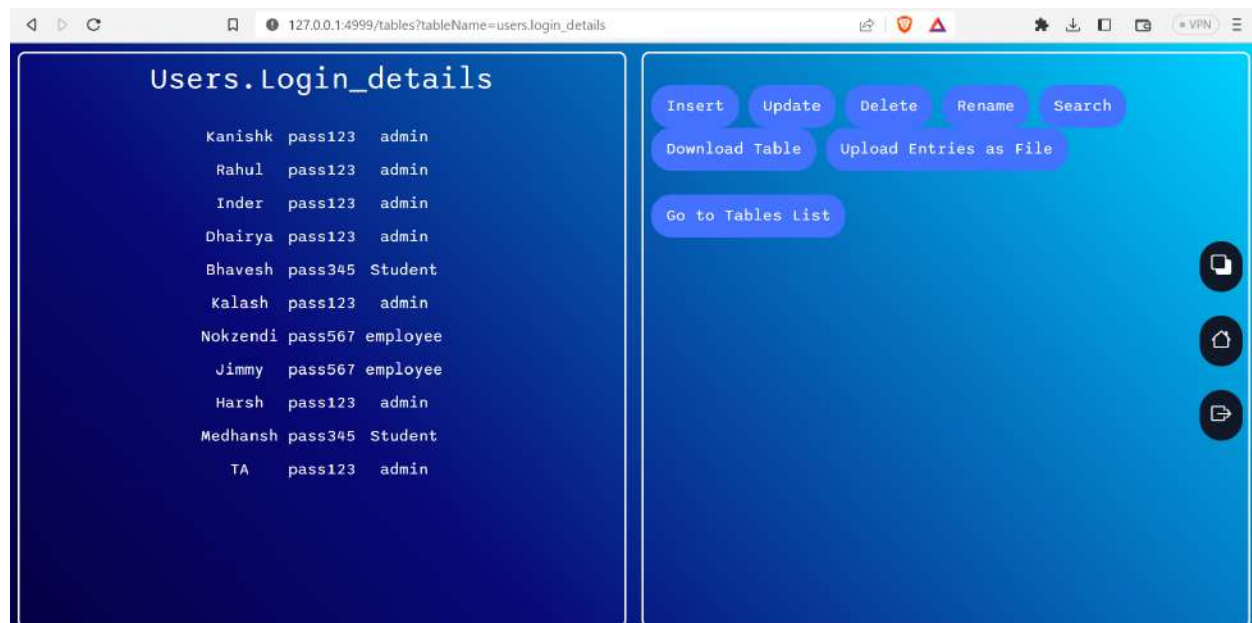
No user should be able to view other user’s credentials from the web application- not even the admin. This should only be possible via the Workbench. Moreover, the web application doesn’t even display the “users” database where our login credentials are stored. So, no access to the “users” database should be possible from this web application made for the “alumni” database.

But we can access it using the same SQL injection in the unsanitized URL input.

Input



Output



This kind of display behavior shouldn't be possible from our web application especially when we have a distributed database with multiple concurrent users. Also, the entries can directly be updated here and deleted from the web-app itself giving users the ability to log out other users arbitrarily. This behavior is unsafe.

Before			After		
name	password	role	name	password	role
Kanishk	pass123	admin	Kanishk	pass123	admin
Rahul	pass123	admin	Rahul	pass123	admin
Inder	pass123	admin	Inder	pass123	admin
Dhairya	pass123	admin	Dhairya	pass123	admin
Bhavesh	pass345	Student	Bhavesh	pass345	Student
Kalash	pass123	admin	Kalash	pass123	admin
Nokzendi	pass567	employee	Nokzendi	pass567	employee
Jimmy	pass567	employee	Jimmy	pass567	employee
Harsh	pass123	admin	Harsh	pass123	admin
Medhansh	pass345	Student	Medhansh	pass345	Student
TA	pass123	admin	TA	pass123	admin
			hacker	hacked_now	admin

[Go to Tables list](#)
[Go Back to users.login_details table](#)

Above, the hacker can insert his own credentials and gain admin access.

Solution-

The defense against such an SQL injection is to sanitize the input whenever it is received from the user. We shouldn't directly take user input as it is. We should first do some checks and only display data if the input is validated and secure.

The check I'm performing is whether the tableName input is in the list of tables of the "alumni" database- this way I throw an error whenever I receive a tableName value which is not a valid name of a table in the alumni database.

```
@app.route('/tables', methods=['GET'])
def tables():
    if session.get('logged_in', False) == False:
        return redirect('/')
    args = request.args

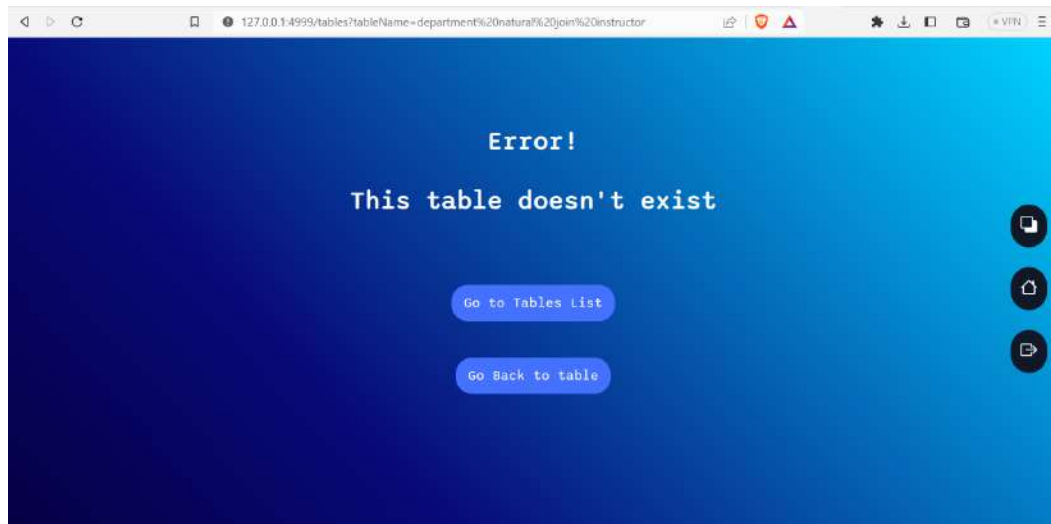
    cursor = mysql.get_db().cursor()
    cursor.execute("SHOW TABLES")
    tables_in_db = cursor.fetchall()
    cursor.close()
    table_names_in_db = []

    for i in range(len(tables_in_db)):
        table_names_in_db.append(tables_in_db[i][0])

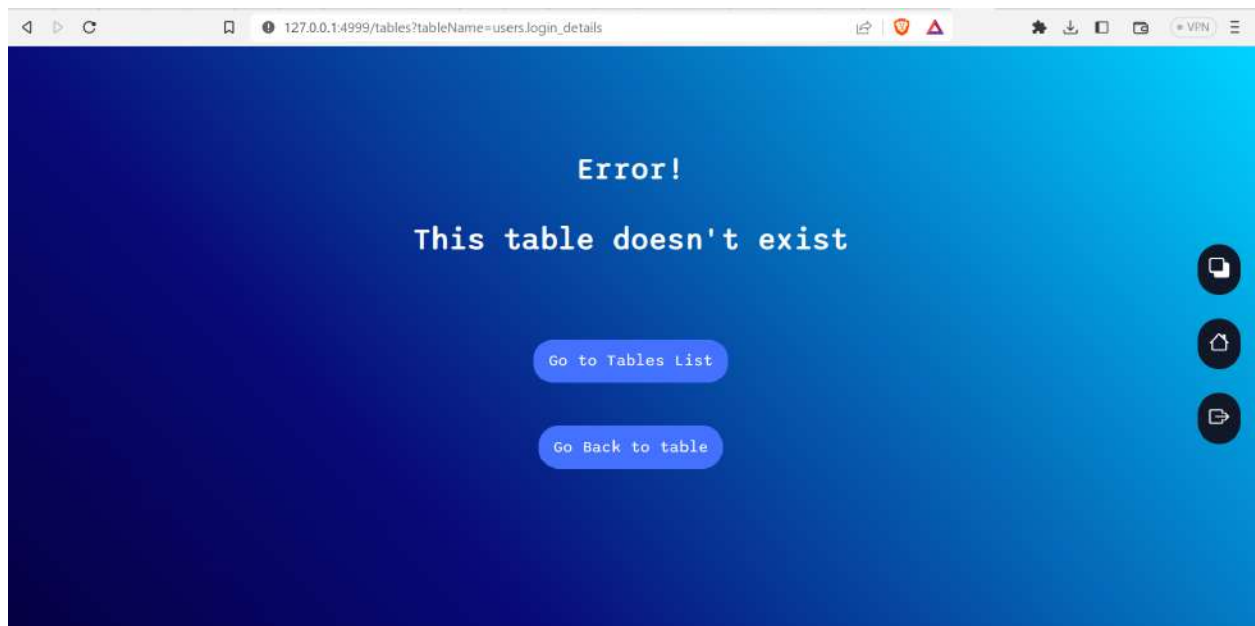
    table_name = args.get('tableName', default=None, type=str)
    if(table_name is not None and table_name not in table_names_in_db):
        return render_template('errors.html', errorMessage="This table doesn't exist")
```

After this change, both the SQL injections don't work anymore. No access to the login database is possible from the alumni database web application. User credentials are not displayed on the web application anymore.

Solved 1



Solved 2

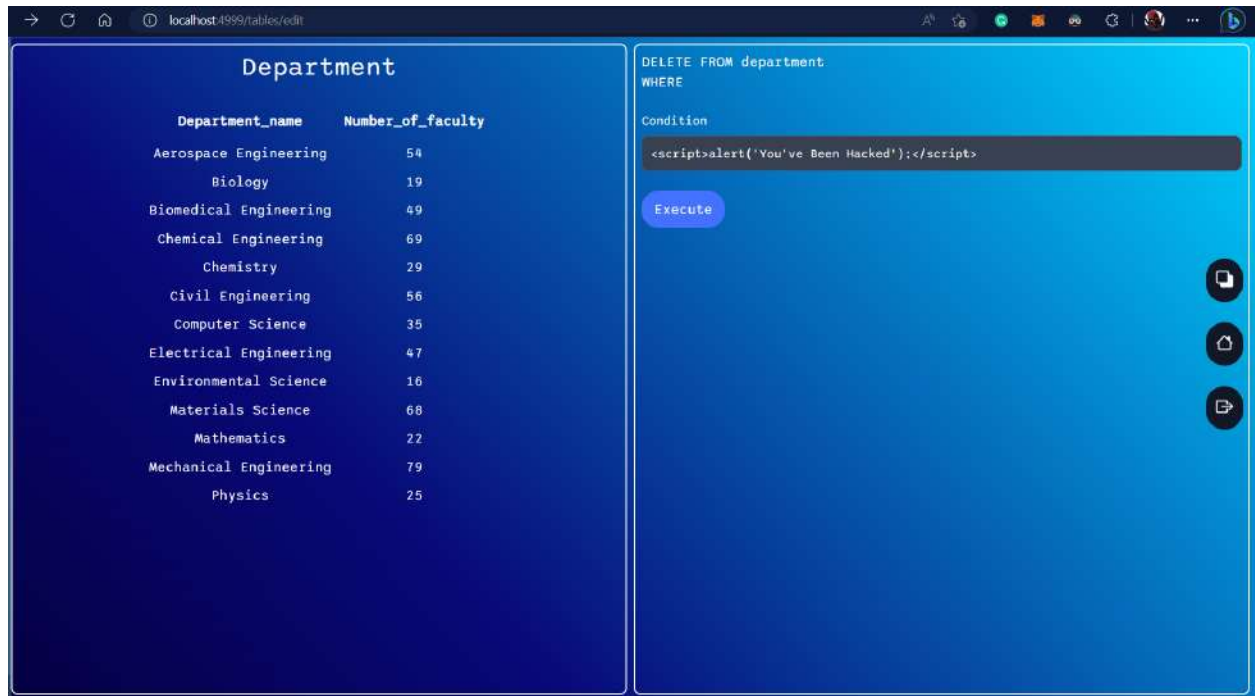


2) Cross Site Scripting(XSS):

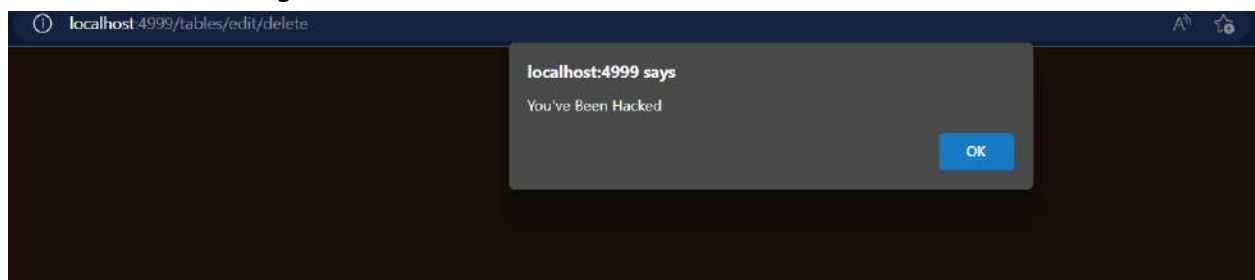
XSS is an attack in which we input some malicious JavaScript in the input box which then gets embedded in the HTML and later gets executed when the infected page is opened.

If we enter the following script in the Input Box-

`<script>alert('You've Been Hacked');</script>`



We are redirected to the error page where we are received with this prompt. This shows that JS can be executed on the server using the input box. Which means DOM elements and other data can be accessed using XSS.



To prevent XSS we have made a variable name "autoescape" true in all the HTML files where input entered is rendered to the site. This is done in the following way-

At the starting of the HTML file add
`{% autoescape true %}`

```
{% autoescape true %}  
<!DOCTYPE html>  
<html lang="en">|
```

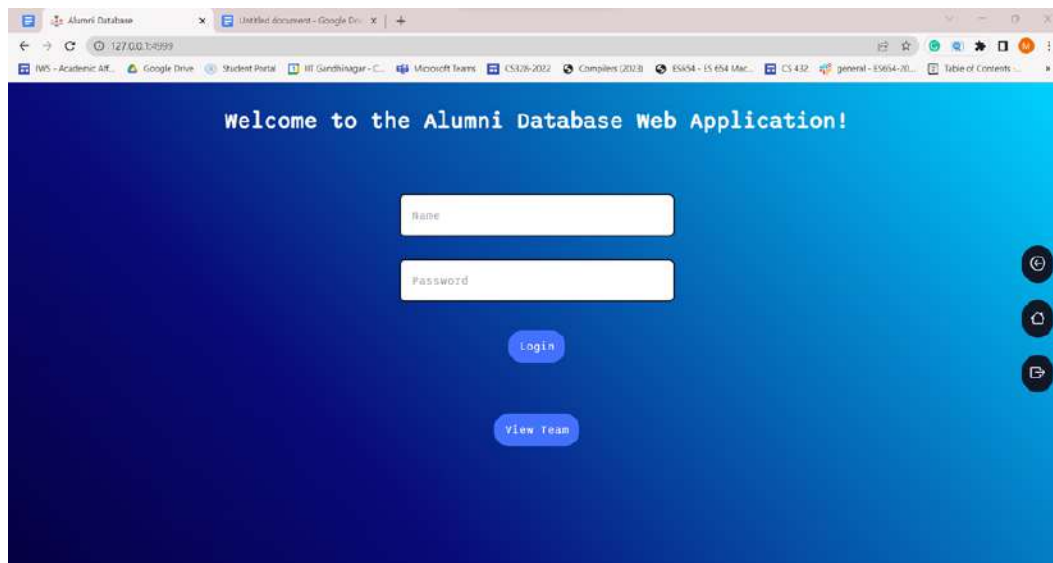
At the end of HTML file add
`{% endautoescape %}`

```
</html>  
{% endautoescape %}
```

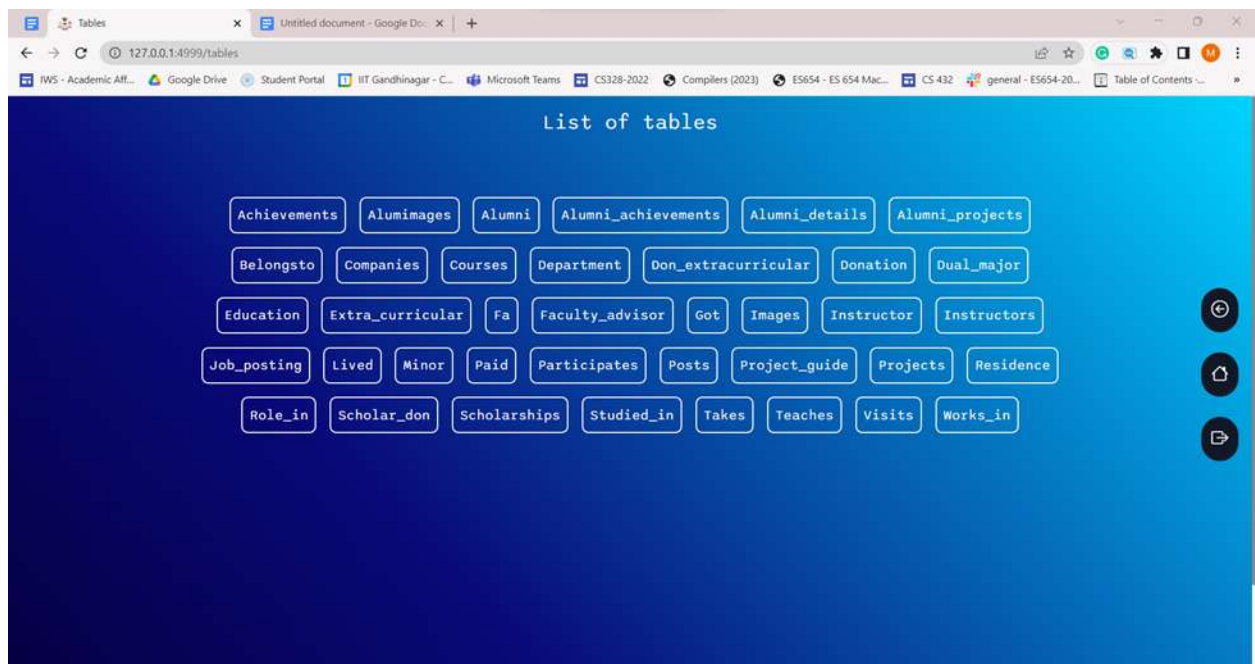
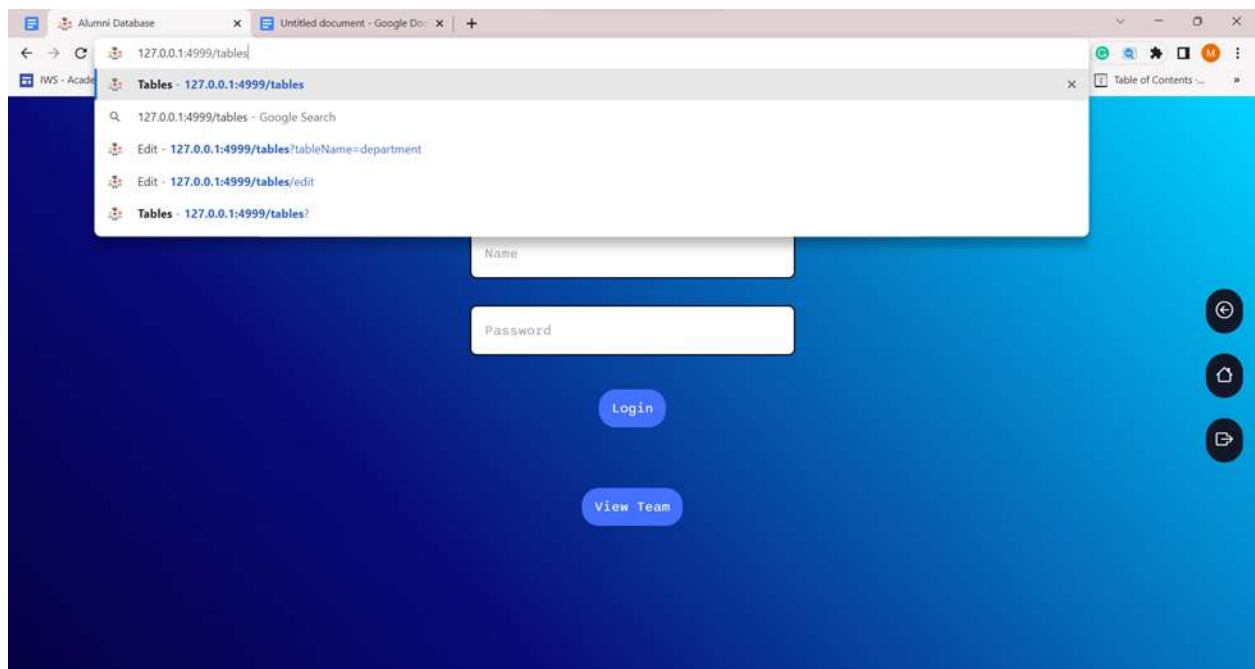

3) URL attack

```
# rendering the list of tables in the database
@app.route('/tables', methods=['GET'])
def tables():

    args = request.args
    table_name = args.get('tableName', default=None, type=str)
    if table_name is None:
        cursor = mysql.get_db().cursor()
        cursor.execute("SHOW TABLES")
```



The login details are asked on the welcome page, but if an attacker directly writes a URL that leads to a page that can only be seen after authentication, it is allowed. As seen in the code piece above, no mechanism was put in place to stop this. For example, refer to the figure below. The attacker could add '/tables' in the URL and would have bypassed the login to access the database as admin directly, as the default role of our database is admin.



Here we can see that the attacker successfully accessed the database as admin.

```
# rendering the list of tables in the database
@app.route('/tables', methods=['GET'])
def tables():
    if session.get('logged_in', False) == False:
        return redirect('/')
    args = request.args
    table_name = args.get('tableName', default=None, type=str)
    if table_name is None:
        cursor = mysql.get_db().cursor()
        cursor.execute("SHOW TABLES")
```

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.

4) Stored XSS (Cross-Site Scripting Attack)

Stored XSS is similar to the normal XSS attacks we described above. Here too, we inject some malicious Javascript code in the input field.

The only difference is that this malicious code is now stored somewhere into the database. This means that it can affect every single user of the web application as the database is shared. Then later, when the query is run on the database into which a malicious code script (added as a text field in some row of some table) and the infected row is retrieved from the database, then we end up with this row being rendered by the browser.

When the browser comes across the infected row- it sees the `<script> {code} </script>` tag and executes the code. Another big difference is that a single script injected by an attacker affects every user because the same database is shared by all the users- so the same malicious code will be run for all the users.

Another difference is that it can be harder to detect because the malicious code lives in the database till either it is run by some user, or we have scanned the database and found this malicious code by ourselves.

In the demo attack, I will insert an entry into the department table. Because the input is not sanitized, the code is inserted as text into the table. Then, the user is directed to the results page and here, while loading the "Before" version of the table- there will be no error but when loading the "After" version of the table- the malicious code will be executed and the attacker will be successful.

Code Injection

The screenshot shows a web application interface. On the left, there is a table titled "Department" with two columns: "Department_name" and "Number_of_faculty". The table lists various departments and their corresponding faculty counts. On the right, there is a form titled "INSERT INTO department". It has two input fields: "Department_name" and "Number_of_faculty". The "Department_name" field contains the malicious SQL injection code: `<script>while(true){alert('You have been hacked again.')}</script>`. The "Number_of_faculty" field contains the value "111". Below the input fields is an "Execute" button. To the right of the "Execute" button are three circular icons: a home icon, a refresh icon, and a copy icon.

Department_name	Number_of_faculty
Aerospace Engineering	54
Biology	19
Biomedical Engineering	49
Chemical Engineering	69
Chemistry	29
Civil Engineering	56
Computer Science	35
Electrical Engineering	47
Environmental Science	16
Financial Engineering	55
HechEsEs	50
Materials Science	68
Mathematics	22
Mechanical Engineering	70

INSERT INTO
department

Department_name
`<script>while(true){alert('You have been hacked again.')}</script>`

Number_of_faculty
111

Execute

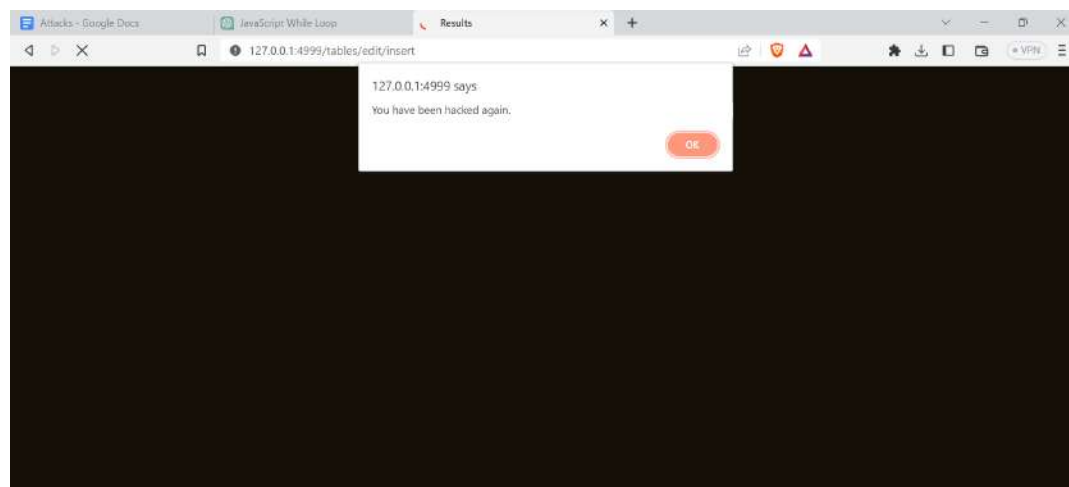
The input in the textbox is-

`<script>alert("You have been hacked again.");</script>`

This code is malicious and traps all the users in an infinite loop when they try to view the department table. This can not be fixed from the web-application as it is now stuck in an infinite loop- the app has broken.

The only way to fix it is using the SQL Workbench to delete the entry, and the same error will pop up for every user.

Attack Output



This alert will persist till we close the tab and patch the error in the database manually in the Workbench.

Defense

The defense remains the same- sanitize the user input properly before inserting it into the database, and providing it to the browser because unsanitized user inputs could contain malicious JS code.

This is handled by using the autoescape variable and setting it to true.

At the starting of the HTML file add

```
{% autoescape true %}
```

```
{% autoescape true %}  
<!DOCTYPE html>  
<html lang="en">
```

At the end of HTML file add

```
{% endautoescape %}
```

```
</html>  
{% endautoescape %}
```

Relations and their constraints, finalized after the second feedback, are present and valid as per the ER diagram constructed in Assignment 1

- 1) In the Achievements table, the Roll number is the primary key, so if I insert another entry with an existing roll number, it should give an error.

Achievements

Roll_number	Purpose	Achievement_Date	Description
10096079	Cultural	2022-12-20	Particular type great perform. Crime mind should music. Every future prove news ready.
10165745	Cultural	2022-12-12	Just model spring government. Thank number officer maintain claim today. Be sort nice book song.
10447530	Cultural	2022-02-15	Head service career stay item. Down cost imagine hand gun size.
10785305	Sports	2022-07-07	Third year report. Fast test current write seven bed hit newspaper. Likely skin no meet note put.
10852417	Sports	2022-03-20	Tv outside few run pass husband again. Executive blood smile property if else real.
10885496	Cultural	2022-02-01	Analysis once especially so. Wind same parent. Begin better that conference.
			Need federal his third expert.

INSERT INTO achievements

Roll_number: 10096079

Purpose: Sports

Achievement_Date: 2022-12-20

Description: Third year report. Fast test current write seven bed hit newspaper. Likely skin

Execute

Error!

Input Error- Re-check your input against the schema.

Duplicate entry '10096079' for key 'achievements.PRIMARY'

Go to Tables List

Go Back to table

- 2) In the achievements table, the date is to be written in the format: YYYY-MM-DD, so we will get an error if I give a random string as the date.

The screenshot shows a web application interface with a table titled 'Achievements' and an 'INSERT INTO achievements' form. The table has four columns: Roll_number, Purpose, Achievement_Date, and Description. The form has four input fields: Roll_number (10096045), Purpose (Sports), Achievement_Date (hello), and Description (Third year report. Fast test current write seven bed hit newspaper. Likely skin). An 'Execute' button is at the bottom of the form.

Roll_number	Purpose	Achievement_Date	Description
10096079	Cultural	2022-12-20	Particular type great perform. Crime mind should music. Every future prove news ready.
10165745	Cultural	2022-12-12	Just model spring government. Thank number officer maintain claim today. Be sort nice book song.
10447530	Cultural	2022-02-15	Head service career stay item. Down cost imagine hand gun size.
10785305	Sports	2022-07-07	Third year report. Fast test current write seven bed hit newspaper. Likely skin no meet note put.
10852417	Sports	2022-03-20	Tv outside few run pass husband again. Executive blood smile property if else real.
10885496	Cultural	2022-02-01	Analysis once especially so. Wind same parent. Begin better that conference. Need federal his third expert.

The screenshot shows an error message on a blue background. The text reads: 'Error! Input Error- Re-check your input against the schema. Incorrect date value: 'hello' for column 'Achievement_Date' at row 1'. There are two buttons at the bottom: 'Go to Tables List' and 'Go Back to table'.

Error!

Input Error- Re-check your input against the schema.

Incorrect date value: 'hello' for column 'Achievement_Date' at row 1

[Go to Tables List](#)

[Go Back to table](#)

- 3) As FA is a many-to-one relationship between alumni and instructors, we can see a few repeated instructor ids corresponding to different roll numbers in FA.

Fa	
Roll_Number	Instructor_ID
22889429	1008
23364810	1008
23761215	1008
31724572	1008
34735201	1008
36670264	1008
54920306	1008
59989896	1008

- 4) Department_name is the primary key of the table Department, which is referenced in Dual Major relation as a foreign key. I should get an error if I delete any entry from the department relation which is referenced as a foreign key in the dual_major relation between Department and Alumni.

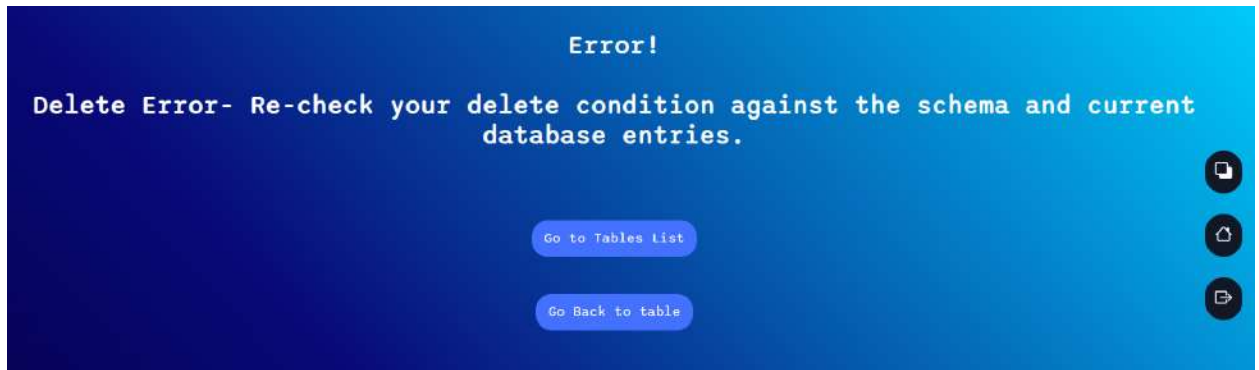
Department	
Department_name	Number_of_faculty
Aerospace Engineering	54
Biology	19
Biomedical Engineering	49
Chemical Engineering	69
Chemistry	29
Civil Engineering	56
Computer Science	35
Electrical Engineering	47
Environmental Science	16
Materials Science	68
Mathematics	22
Mechanical Engineering	79
Physics	25

```
DELETE FROM department
WHERE
```

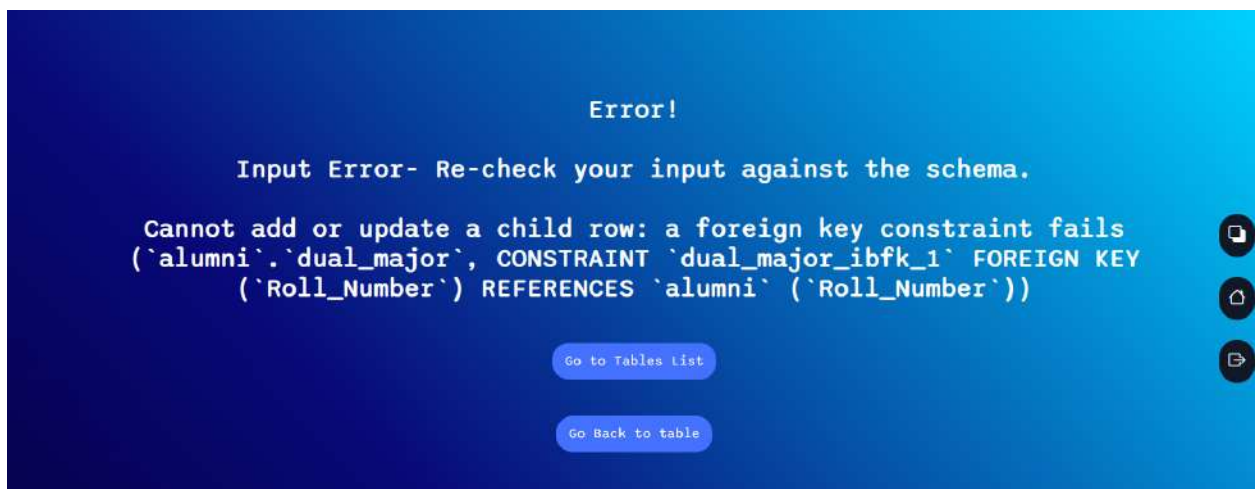
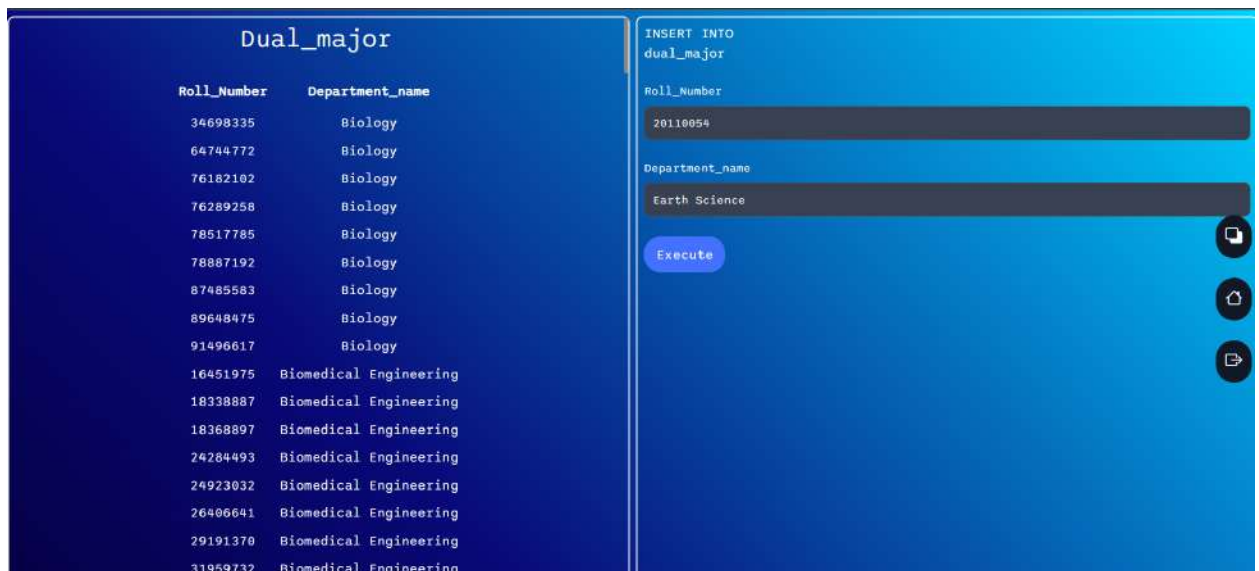
Condition

Department_name = 'Aerospace Engineering'

Execute



- 5) Department_name is the primary key of the table Department, which is referenced in Dual Major relation as a foreign key. So I should get an error if I add an entry with a department name that is not present in the department table.



- 6) In instructors table, you can observe that, the instructor ID and Department name cannot be NULL, so if we insert an entry with Department_name set as NULL, it should give an error.

Instructor_ID	Instructor_Name	Work_email	Department
1088	Benjamin Lucero	williamsleslie@example.net	Biolog
1144	Carla Neal	garciawendy@example.com	Biolog
1216	Michael Soto	odonovan@example.org	Biolog
1347	Sara Brown	leepatricia@example.net	Biolog
1402	Michael Wise	kpacheco@example.org	Biolog
1428	David Randall	victoria35@example.com	Biolog
1674	William Hill	sherrismith@example.com	Biolog
1684	Nicole Harvey	kevinwilson@example.org	Biolog
1693	Daniel Thompson	monica97@example.net	Biolog
1694	Michelle Lee	shaun09@example.net	Biolog
1732	Morgan Hardin	david25@example.net	Biolog
1815	Mark Hughes	wangethan@example.com	Biolog
1960	Devin Mcdonald	belindamoody@example.com	Biolog
2070	Tonya Hunter	tmahoney@example.com	Biolog
2279	Nathaniel Clark	alandelgado@example.net	Biolog
2455	Darrell Taylor	duffyrichard@example.com	Biolog
2489	Kevin Lee	ryan81@example.com	Biolog
2545	Kelsey Hunter	jeremy68@example.com	Biolog

INSERT INTO instructors

Instructor_ID: 1499

Instructor_Name: Dhairya Shah

Work_email: dhairya.shah@iitgn.ac.in

Department_name:

Execute

Error!

Input Error- Re-check your input against the schema.

Can not modify more than one base table through a join view 'alumni.instructors'

Go to Tables list

Go Back to table

- 7) In the Department table, we have allowed Number_of_faculty to be Decimal(4,0). So if you do something different, then it will give an error.

Field	Type	Null	Key
Department_name	Varchar(100)	NO	PRI
Number_of_faculty	Decimal(4,0)	YES	

Department

Department_name	Number_of_faculty
Aerospace Engineering	54
Biology	19
Biomedical Engineering	49
Chemical Engineering	69
Chemistry	29
Civil Engineering	56
Computer Science	35
Electrical Engineering	47
Environmental Science	16
Materials Science	68
Mathematics	22
Mechanical Engineering	79
Physics	25

INSERT INTO department

Department_name

Earth Science

Number_of_faculty

4938598

Execute

Error!

Input Error- Re-check your input against the schema.

Out of range value for column 'Number_of_faculty' at row 1

[Go to Tables List](#)

[Go Back to table](#)

Contributions

Dhairya Shah

- Collected timely feedback from the stakeholders and Prof. Singh.
- Documented Screenshots of different views and a write-up on their privileges.
- Tested 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.
- Helped improve the UI/UX after the first feedback and solve previous bugs in UI from the last assignment.
-

Bhavesht Jain

- Got the feedback from the stakeholders.
- Helped in researching vulnerabilities and worked on attacks.
- Worked on improving UX.

Inderjeet Singh

- Helped in taking feedback through various stakeholders regarding our database.
- Worked on the Implementation of some of those features, including changes to the UI.
- Changed the way tables look to the "Student" view as was given in the feedback, i.e. removed the redundant right block that was visible earlier.
- Worked on changing the tooltip text for each table for the "Student" view specifically.
- Showed exactly how multi-user concurrency works in our database and how 2 users cannot edit the same item simultaneously.

Rahul Chembakasseril

- Added the full search, upload, download, and mailing features as per the first feedback.
- Detected and solved two vulnerabilities- SQL injection via the unsanitized input in the address bar, Stored XSS injection via the unsanitized input in the 'Insert' input.

Joy Makwana

- Added the dropdowns for the delete feature in employee role as per the feedback.
- Improved the UI/UX for employee and admin role.
- Improved the UI of the Send Mail page, including add/remove button.

Kalash Kankaria

- Worked on the UI/UX improvements as per the feedback received. These changes include:
 - For the student role, changed the tooltip text in 'List of Tables' to show a short description of the particular table instead of the entire schema
 - Removed the right empty container box visible in the student role when any of the tables were opened.

- Exhibited the working of multi-user concurrency in our database and the restrictions on changing data simultaneously by 2 different users concurrently.

Kanishk Singhal

- Improved the UI/UX features mentioned in the feedback. Handled UI bugs occurring in the previous submission.
- Working on building SQL injection and XSS attack. Also built defenses against these attacks.
- Worked on the mailing feature provided in the search menu.
- Made a couple of presentations of the website to the stakeholders taking their feedback twice.

Harshvardhan Vala

- Helped in taking feedback from various stakeholders regarding our database, and worked on improving the UI/UX of the web app after the first feedback.
- Documented all the major constraints in our database with their screenshots, which are in sync with our ER diagram.

Nokzendi Aier

- Helped in finding vulnerabilities
- Worked on attacking the vulnerabilities and their fixes
- Helped in improving the overall UI/UX

Medhansh Singh

- Worked on carrying out various attacks on the website.
- Detected and solved a vulnerability - Bypassing login via URL attack.