

RESPONSIBILITY OF G1:

FEEDBACK FROM THE STAKEHOLDER

Before the first feedback:

1. No housekeeping section is available

The screenshot shows a Google Chrome browser window with the URL 127.0.0.1:5000/admin/dashboard. The page title is "Hostel Management System". The main content area is titled "HOSTEL DASHBOARD" and contains nine boxes labeled A through I, each representing a room status:

Room Label	0 students in triple room:	1 students in triple room:	0 students in single room:	1 students in single room:
A	143	-1	144	
B	143		144	
C	144		144	
D	144		144	
E	144		144	
F	144		144	
G	144		144	
H	144		144	
I	144		144	

Fig.1

2. Import data from Excel is not included

The screenshot shows a web-based Hostel Management System interface. At the top, there's a navigation bar with links for Dashboard, Residents, Rooms, Security, Furniture, Hostel, Caretakers, Academic period, Outlets, and Logout. Below the navigation is a search bar labeled "Add a New Resident". The main area is titled "Resident Details" and contains a table with columns: ID, Resident Type, Hostel, Program, Branch, and Gender. Each column has a dropdown menu set to "All". Below the table are filters for "Joining Year" (dropdown with "All" selected), "Currently allocated" (checkbox checked), "Fee pending" (checkbox checked), and "Due pending" (checkbox checked). A blue "Apply Filters" button is located to the right of these filters. Below the filters is a table listing four residents with columns: ID, Name, Gender, Room Number, Hostel, and Email. Each resident row has an "Edit" button on the far right.

ID	Name	Gender	Room Number	Hostel	Email
20000000	residentfn1 residentln1	F	120	b	residentemail1@iitgn.ac.in
20000001	residentfn2 residentln2	M	102	j	residentemail2@iitgn.ac.in
20000002	residentfn3 residentln3	M	103	j	residentemail3@iitgn.ac.in
20000003	residentfn4 residentln4	M	104	j	residentemail4@iitgn.ac.in

Fig.2

3. Details of the resident guardian are not available

This screenshot shows the "Resident Details" update form for the resident with ID 20000000. The form includes fields for First Name (residentfn1), Middle Name (residentmn1), Last Name (residentln1), Gender (F), Blood Group (B+), Email id (residentemail1@iitgn.ac.in), City (city1), Postal Code (100000), Home Contact (5000000000), Resident Type (faculty), and a checkbox for "Confirm before update". At the bottom is a large "Update" button. Below the form is a section titled "Phone Numbers".

Fig.3

Features included after the first feedback:

1. **Inclusion of the housekeeping:** Along with the hostel staff, like security and caretaker, the housekeeping section is also included, where the admin can add, edit and delete the details of a member of the housekeeping staff.

The screenshot shows a web-based Admin interface for a Hostel Management System. The top navigation bar includes links for Dashboard, Residents, Rooms, Security, Housekeeping, Furniture, Hostel, Caretakers, Academic period, Outlets, and Logout. A sub-navigation menu for 'Housekeeping' is visible. Below the navigation is a search bar with placeholder text 'Add Housekeeper'. The main content area displays a table titled 'Housekeeper Details' with columns for ID, Name, Hostel, and House Keeping Type. There are five entries in the table, each with an 'Edit' button. The table includes filters for ID, Hostel, and House Keeping Type, and an 'Apply Filters' button. The bottom of the screen shows a Windows taskbar with various icons and system status indicators.

Fig.4: Admin view of the housekeeping section.

This screenshot shows the same Admin interface as Fig.4, but with a modal window open over the 'Housekeeper Details' table. The modal is titled 'Add Housekeeper' and contains fields for ID, First Name, Middle Name, Last Name, Gender (with a dropdown set to 'F'), and Phone Number. Below these fields is an 'Add' button. The background table and its data remain visible through the semi-transparent modal.

Fig.5: Adding of a housekeeping staff member

Housekeeper Details

ID	First Name	Middle Name	Last Name
10000001	hkfname1	hkmname1	hklname1

Gender: M

Update

Phone Numbers

9000000000	Delete
9000000001	Delete

Add Phone

Current Allocation

a	Room	Delete
---	------	--------

Allocate

Fig.6: Updation, deletion, and current room allocation

2. Admin can import resident data from the Excel sheets.

Hostel Management System

Add a New Resident **Deallocate all students**

Import Resident Data

Import Allocation Data

Resident Details

ID:	Resident Type	Hostel	Program	Branch	Gender
ID	All	All	All	All	All
Joining Year	All	<input type="checkbox"/> Currently allocated	<input type="checkbox"/> Fee pending	<input type="checkbox"/> Due pending	Apply Filters

ID	Name	Gender	Room Number	Hostel	Email
2000000	residentfn1 residentln1	F	101	a	residentemail1@itgn.ac.in
2000001	residentfn2 residentln2	M	102	j	residentemail2@itgn.ac.in

Fig.7: An option to include the resident in the form of a CSV file

The screenshot shows the 'Resident Details' section of the Hostel Management System. At the top, there are two input fields: 'Choose file' and 'No file chosen' for 'Import Resident Data', and 'Choose file' and 'No file chosen' for 'Import Allocation Data'. Below these are several dropdown filters: 'ID', 'Resident Type' (set to 'All'), 'Hostel' (set to 'All'), 'Program' (set to 'All'), 'Branch' (set to 'All'), and 'Gender' (set to 'All'). A 'Joining Year' dropdown is also present, set to 'All'. There are three checkboxes: 'Currently allocated' (unchecked), 'Fee pending' (unchecked), and 'Due pending' (unchecked). A blue 'Apply Filters' button is located to the right of the checkboxes. The main table below has columns: ID, Name, Gender, Room Number, Hostel, and Email. The table is currently empty.

Fig.8: Before importing data

The screenshot shows the same 'Resident Details' section as Fig.8, but with a modal dialog box in the center. The dialog has a title 'Success' and a message 'Students added successfully'. It includes an 'Ok' button at the bottom right. The rest of the interface is identical to Fig.8, including the filters and the empty table below.

Fig.9: A pop to denote the successful importing of data

The screenshot shows the 'Resident Details' section of the Hostel Management System. At the top, there are buttons for 'Add a New Resident', 'Import Resident Data', 'Import Allocation Data', and 'Deallocate all students'. Below these are dropdown filters for ID, Resident Type, Hostel, Program, Branch, and Gender, all set to 'All'. There are also filters for 'Joining Year' (set to 'All') and checkboxes for 'Currently allocated', 'Fee pending', and 'Due pending', with a 'Apply Filters' button. The main table lists two residents:

ID	Name	Gender	Room Number	Hostel	Email
20000000	residentfn1 residentln1	M			residentemail1@itgn.ac.in
20000001	residentfn2 residentln2	M			residentemail2@itgn.ac.in

Fig.10: After importing the resident data

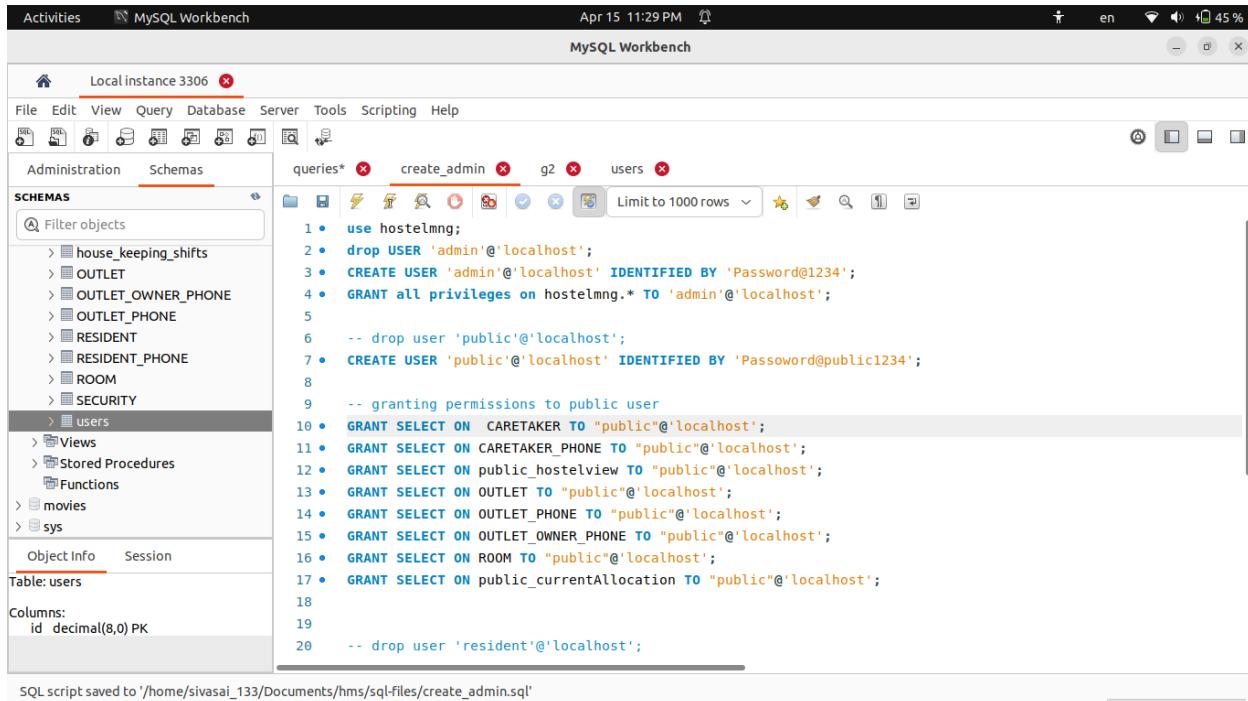
3. Adding a section to store the Guardian details and Emergency contact of the resident:

The screenshot shows the 'Resident Details' page for a specific resident. The resident's ID is 20000000. The form includes fields for First Name (residentfn1), Middle Name (residentmn1), Last Name (residentln1), Gender (M), Blood Group (O+), Email id (residentemail1@itgn.ac.in), City (city1), Postal Code (100000), Resident Type (faculty), and a section for 'Guardian Name' (guardianfn1), 'Guardian Type' (Father), and 'Home Contact' (500000000). A green box highlights the 'Guardian Name', 'Guardian Type', and 'Home Contact' fields. Below the form is a 'Phone Numbers' section with an 'Add Phone' button, and a 'Current Allocation' section with an 'Allocate' button.

Fig.11

Different Views

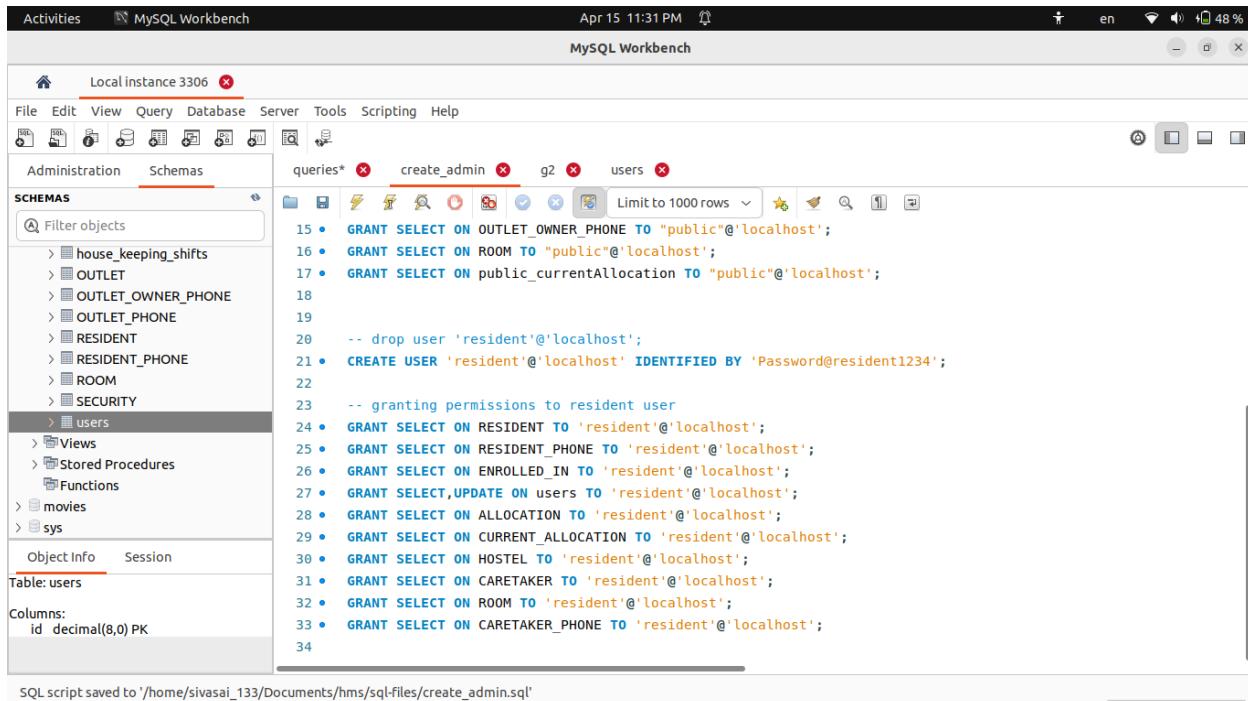
Generating multiple users and views and granting permissions to them.



The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL script:

```
1 • use hostelmng;
2 • drop USER 'admin'@'localhost';
3 • CREATE USER 'admin'@'localhost' IDENTIFIED BY 'Password@1234';
4 • GRANT all privileges on hostelmng.* TO 'admin'@'localhost';
5
6 -- drop user 'public'@'localhost';
7 • CREATE USER 'public'@'localhost' IDENTIFIED BY 'Passoword@public1234';
8
9 -- granting permissions to public user
10 • GRANT SELECT ON CARETAKER TO "public"@"localhost";
11 • GRANT SELECT ON CARETAKER_PHONE TO "public"@"localhost";
12 • GRANT SELECT ON public_hostelview TO "public"@"localhost";
13 • GRANT SELECT ON OUTLET TO "public"@"localhost";
14 • GRANT SELECT ON OUTLET_PHONE TO "public"@"localhost";
15 • GRANT SELECT ON OUTLET_OWNER_PHONE TO "public"@"localhost";
16 • GRANT SELECT ON ROOM TO "public"@"localhost";
17 • GRANT SELECT ON public_currentAllocation TO "public"@"localhost";
18
19
20 -- drop user 'resident'@'localhost';
```

SQL script saved to '/home/sivasai_133/Documents/hms/sql-files/create_admin.sql'

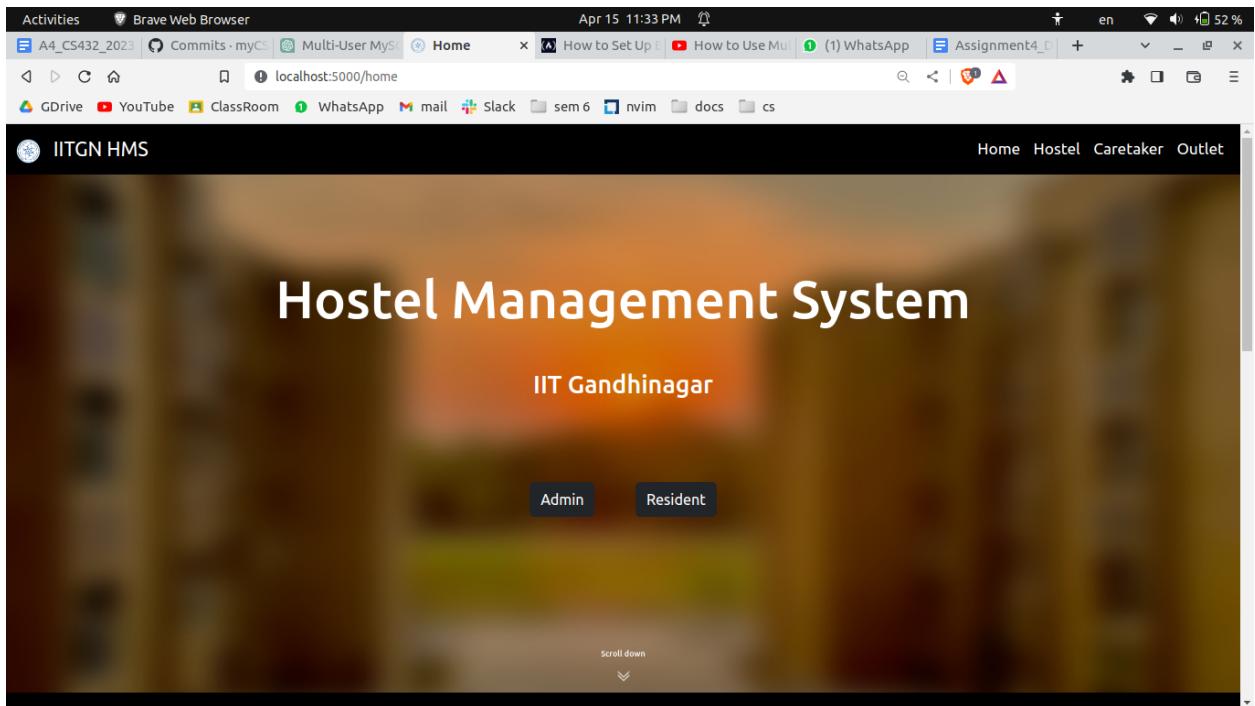


The screenshot shows the MySQL Workbench interface with a query editor containing the following SQL script:

```
15 • GRANT SELECT ON OUTLET_OWNER_PHONE TO "public"@"localhost";
16 • GRANT SELECT ON ROOM TO "public"@"localhost";
17 • GRANT SELECT ON public_currentAllocation TO "public"@"localhost";
18
19
20 -- drop user 'resident'@'localhost';
21 • CREATE USER 'resident'@'localhost' IDENTIFIED BY 'Password@resident1234';
22
23 -- granting permissions to resident user
24 • GRANT SELECT ON RESIDENT TO 'resident'@'localhost';
25 • GRANT SELECT ON RESIDENT_PHONE TO 'resident'@'localhost';
26 • GRANT SELECT ON ENROLLED_IN TO 'resident'@'localhost';
27 • GRANT SELECT,UPDATE ON users TO 'resident'@'localhost';
28 • GRANT SELECT,UPDATE ON ALLOCATION TO 'resident'@'localhost';
29 • GRANT SELECT ON CURRENT_ALLOCATION TO 'resident'@'localhost';
30 • GRANT SELECT ON HOSTEL TO 'resident'@'localhost';
31 • GRANT SELECT ON CARETAKER TO 'resident'@'localhost';
32 • GRANT SELECT ON ROOM TO 'resident'@'localhost';
33 • GRANT SELECT ON CARETAKER_PHONE TO 'resident'@'localhost';
34
```

SQL script saved to '/home/sivasai_133/Documents/hms/sql-files/create_admin.sql'

Public view



The screenshot shows a web browser window titled "Brave Web Browser" with the URL "localhost:5000/home/caretaker_details". The page has a dark background with a blurred image of a building. At the top, there is a navigation bar with links for "Home", "Hostel", "Caretaker", and "Outlet". Below the navigation bar, the text "Hostel Management System" is displayed. The main title "Caretaker Details" is centered in large white font. Below the title is a table with the following data:

ID	Name	Office Number	Email ID	Hostel Name	Hostel Contact	Caretaker Phono
10000001	caretakername1 caretakerlname1	a126	caretakeremail1@iitgn.ac.in	a	1000000000	7000000001
10000002	caretakername2 caretakerlname2	b163	caretakeremail2@iitgn.ac.in	b	1000000001, 7000000002, 7000000003, 7000000004, 7000000005	
10000003	caretakername3 caretakerlname3	c168	caretakeremail3@iitgn.ac.in	c	1000000002	7000000006, 7000000007, 7000000008
10000004	caretakername4 caretakerlname4	d130	caretakeremail4@iitgn.ac.in	d	1000000003	7000000009, 7000000010
10000005	caretakername5 caretakerlname5	e155	caretakeremail5@iitgn.ac.in	e	1000000004	7000000011, 7000000012, 7000000013, 7000000014
10000006	caretakername6 caretakerlname6	f184	caretakeremail6@iitgn.ac.in	f	1000000005	7000000015, 7000000016, 7000000017, 7000000018
10000007	caretakername7 caretakerlname7	g197	caretakeremail7@iitgn.ac.in	g	1000000006	7000000019, 7000000020, 7000000021
10000001	caretakername1 caretakerlname1	a126	caretakeremail1@iitgn.ac.in	h	1000000007	7000000001
10000002	caretakername2 caretakerlname2	b163	caretakeremail2@iitgn.ac.in	i	1000000008	7000000002, 7000000003, 7000000004, 7000000005
10000003	caretakername3 caretakerlname3	c168	caretakeremail3@iitgn.ac.in	j	1000000009	7000000006, 7000000007, 7000000008
10000004	caretakername4 caretakerlname4	d130	caretakeremail4@iitgn.ac.in	k	1000000010	7000000009, 7000000010
10000005	caretakername5 caretakerlname5	e155	caretakeremail5@iitgn.ac.in	l	1000000011	7000000011, 7000000012, 7000000013, 7000000014

Admin view

Activities Brave Web Browser April 15 11:32 PM en 50 %

A4_CS432_2023 Commits - myCS Multi-User MySQL Admin x How to Set Up YouTube How to Use MySQL (1) WhatsApp Assignment4... +

localhost:5000/admin/dashboard

GDrive YouTube ClassRoom WhatsApp Gmail Slack sem 6 nvim docs cs

Hostel Management System Dashboard Residents Rooms Security Housekeeping Furniture Hostel Caretakers Academic period Outlets Logout

HOSTEL DASHBOARD

A	B	C
2 students in triple room: 4	0 students in triple room: 144	0 students in triple room: 144
0 students in triple room: 67	0 students in single room: 144	0 students in single room: 144
0 students in single room: 70		
1 students in triple room: 73		
D	E	F
0 students in single room: 144	0 students in single room: 144	0 students in single room: 144
0 students in triple room: 144	0 students in triple room: 144	0 students in triple room: 144
G	H	I
0 students in triple room: 144	0 students in triple room: 144	0 students in triple room: 144
0 students in single room: 144	0 students in single room: 144	0 students in single room: 144

Activities Brave Web Browser April 15 11:33 PM en 51 %

A4_CS432_2023 Commits - myCS Multi-User MySQL Admin x How to Set Up YouTube How to Use MySQL (1) WhatsApp Assignment4... +

localhost:5000/admin/residents

GDrive YouTube ClassRoom WhatsApp Gmail Slack sem 6 nvim docs cs

Hostel Management System Dashboard Residents Rooms Security Housekeeping Furniture Hostel Caretakers Academic period Outlets Logout

Add a New Resident **Deallocate all students**

Resident Details

ID:	Resident Type	Hostel	Program	Branch	Gender
<input type="text" value="ID"/>	All	All	All	All	All
Joining Year					
All	<input type="checkbox"/> Currently allocated	<input type="checkbox"/> Fee pending	<input type="checkbox"/> Due pending	Apply Filters	

Import Resident Data **Import Allocation Data**

ID	Name	Gender	Room Number	Hostel	Email	Action
20000000	residentfn1 residentln1	F	101	a	residentemail1@iitgn.ac.in	Edit
20000001	residentfn2 residentln2	F	102	a	residentemail2@iitgn.ac.in	Edit

Resident view

The screenshot shows a web browser window titled "Brave Web Browser" with the URL "localhost:5000/resident/profile". The page is titled "Resident Profile" and displays a table of resident information. At the top right, there is a "Change Password" button. The table data is as follows:

Resident Id	20000001
First name	residentfn2
Middle name	residentmn2
Last name	residentln2
Gender	F
Blood group	O-
Email id	residentemail2@iitgn.ac.in
Guardian first name	guardianfn2
Guardian middle name	guardianmn2
Guardian last name	guardianln2
Guardian type	Father
City	city2
Postal code	100001

At the top right of the browser window, there are links for "Profile", "Current allocation", "History", and "Logout". The browser's toolbar at the top includes "Activities", "Brave Web Browser", "Apr 15 11:35 PM", "en", "54 %", and various system icons.

RESPONSIBILITY OF G1&G2:

Before Locks

1)

```
try:
    # Adding the select and where clause
    query_string = f"UPDATE RESIDENT SET {query_string} WHERE resident_id = {resident_id};"

    # Executing the query and committing the changes
    cur.execute(query_string)
    mysql.connection.commit()
    return {"success": True, "reload":True, "message": "Successfully updated details"}

except Exception as e:
    return {"success": False, "reload":False, "message":e.args[1]}
```

2)

```
query_string = f"INSERT INTO CURRENT_ALLOCATION(resident_id, {text_string}) VALUES ({resident_id}, {query_string});"
try:
    # Executing the query and committing the changes
    cur.execute(query_string)
    mysql.connection.commit()
    return {"success": True, "reload":True, "message": "Successfully added allocation"}
except Exception as e:
    return {"success": False, "reload":False, "message":e.args[1]}
```

After Locks

1)

```
try:
    #Locking table resident
    cur.execute("LOCK TABLES RESIDENT WRITE")
    # Adding the select and where clause
    query_string = f"UPDATE RESIDENT SET {query_string} WHERE resident_id = {resident_id};"

    # Executing the query and committing the changes
    cur.execute(query_string)
    mysql.connection.commit()
    return {"success": True, "reload":True, "message": "Successfully updated details"}

except Exception as e:
    mysql.connection.rollback()
    return {"success": False, "reload":False, "message":e.args[1]}

finally:    You, 9 seconds ago * Uncommitted changes
#Unlocking table resident
cur.execute("UNLOCK TABLES")
cur.close()
```

2)

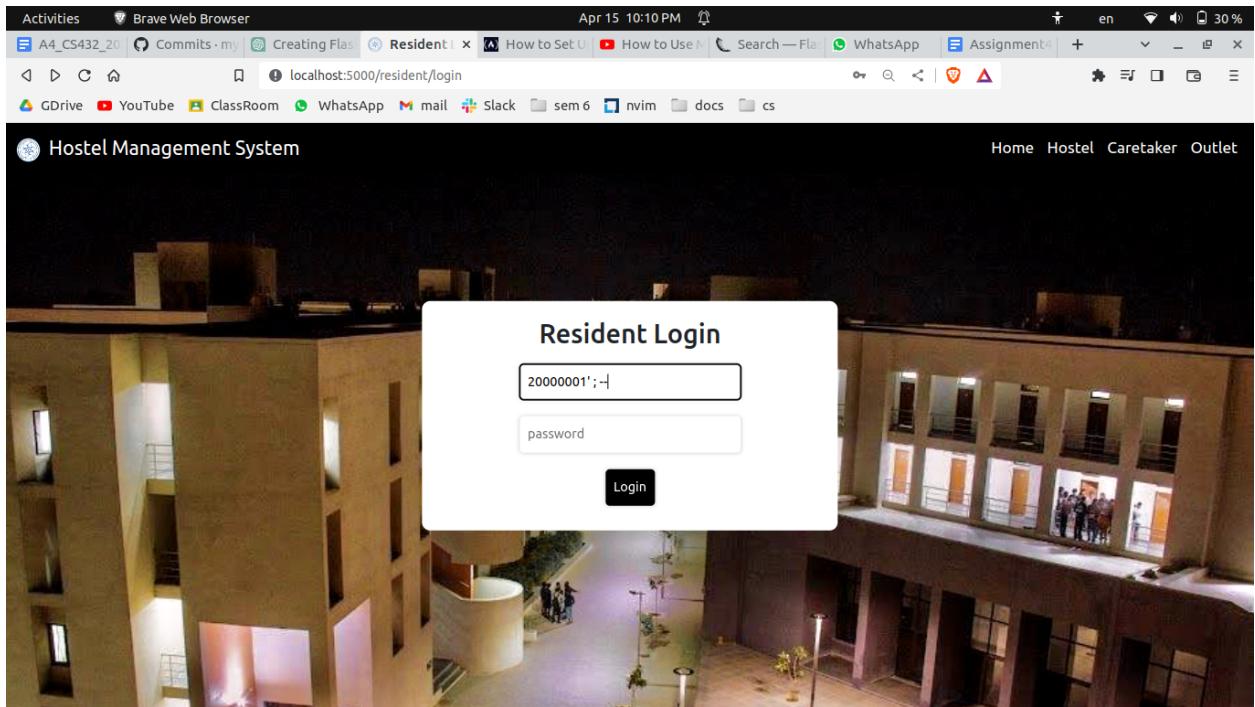
```
try:
    #Locking table Current_allocation
    cur.execute("LOCK TABLES CURRENT_ALLOCATION WRITE")
    # Executing the query and committing the changes
    cur.execute(query_string)
    mysql.connection.commit()
    return {"success": True, "reload":True, "message": "Successfully added allocation"}
except Exception as e:
    mysql.connection.rollback()
    return {"success": False, "reload":False, "message":e.args[1]}

finally:
    #Unlocking table current_allocation      You, 1 second ago * Uncommitted changes
    cur.execute("UNLOCK TABLES")
    cur.close()
```

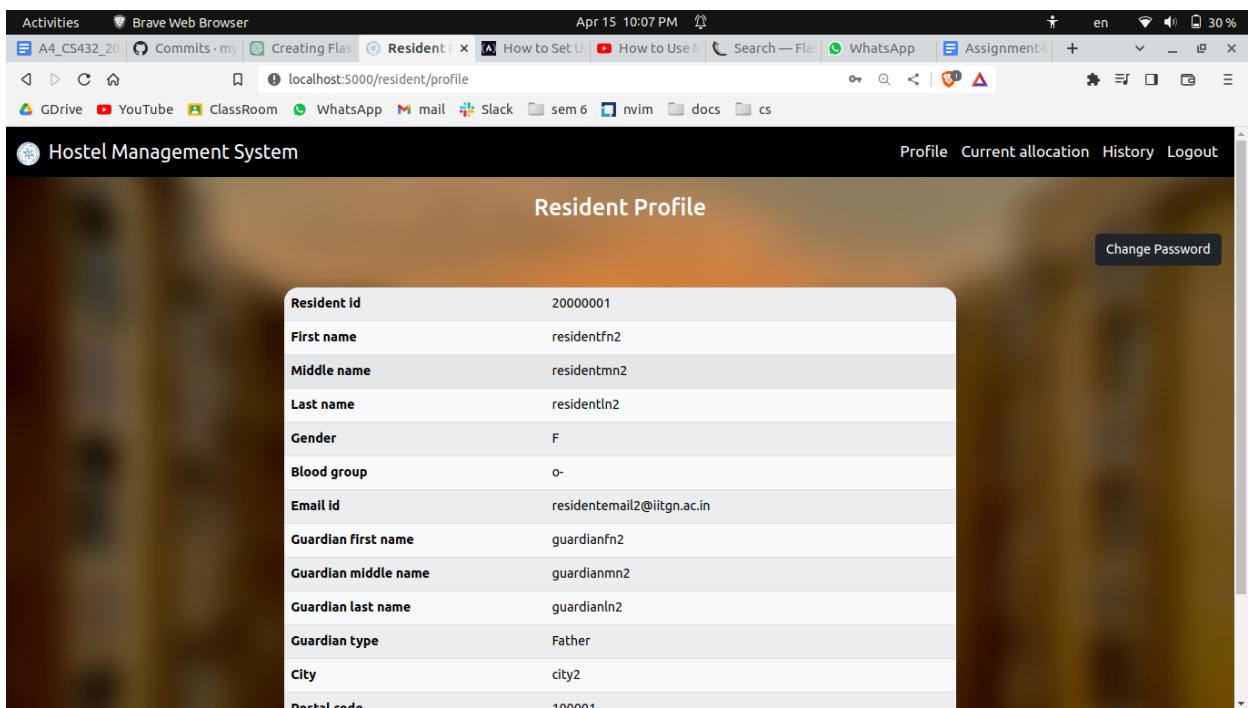
Locks have been implemented at all UPDATE, INSERT, DELETE, and other SQL queries to support concurrency.

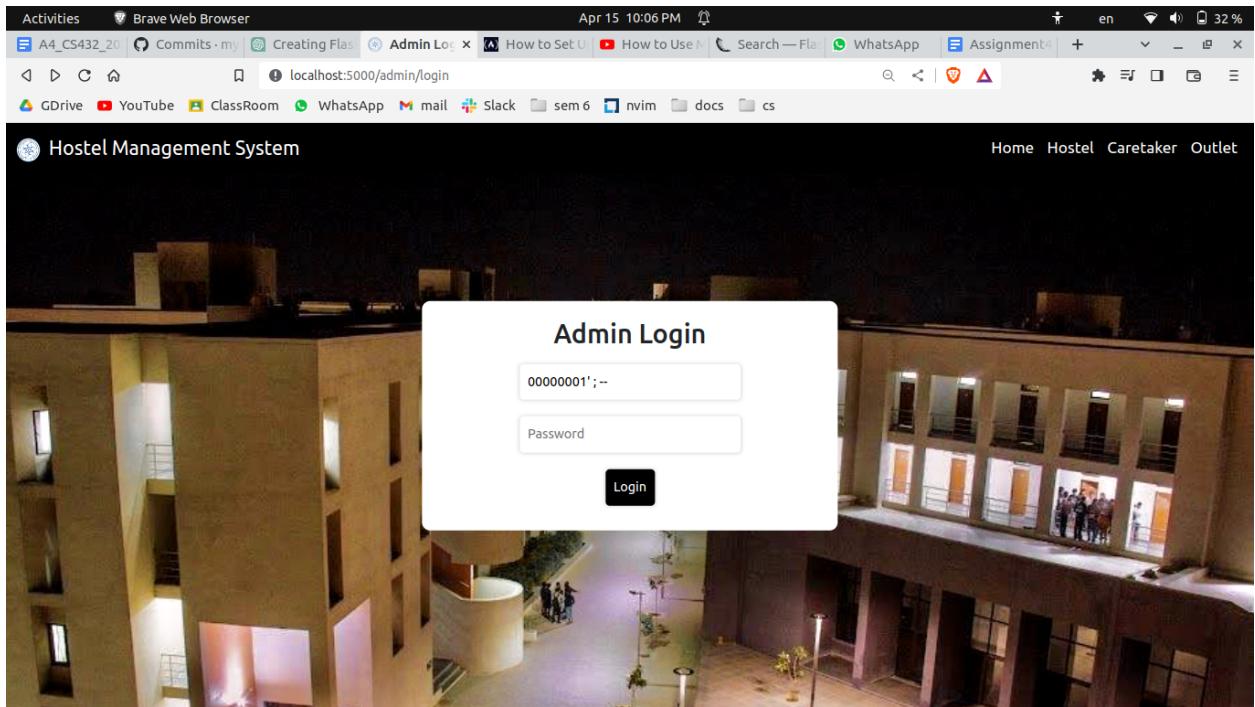
RESPONSIBILITY OF G1&G2:

SQL Injection:



Logging in the resident using SQL injection. Injection used: 20000001' ; --



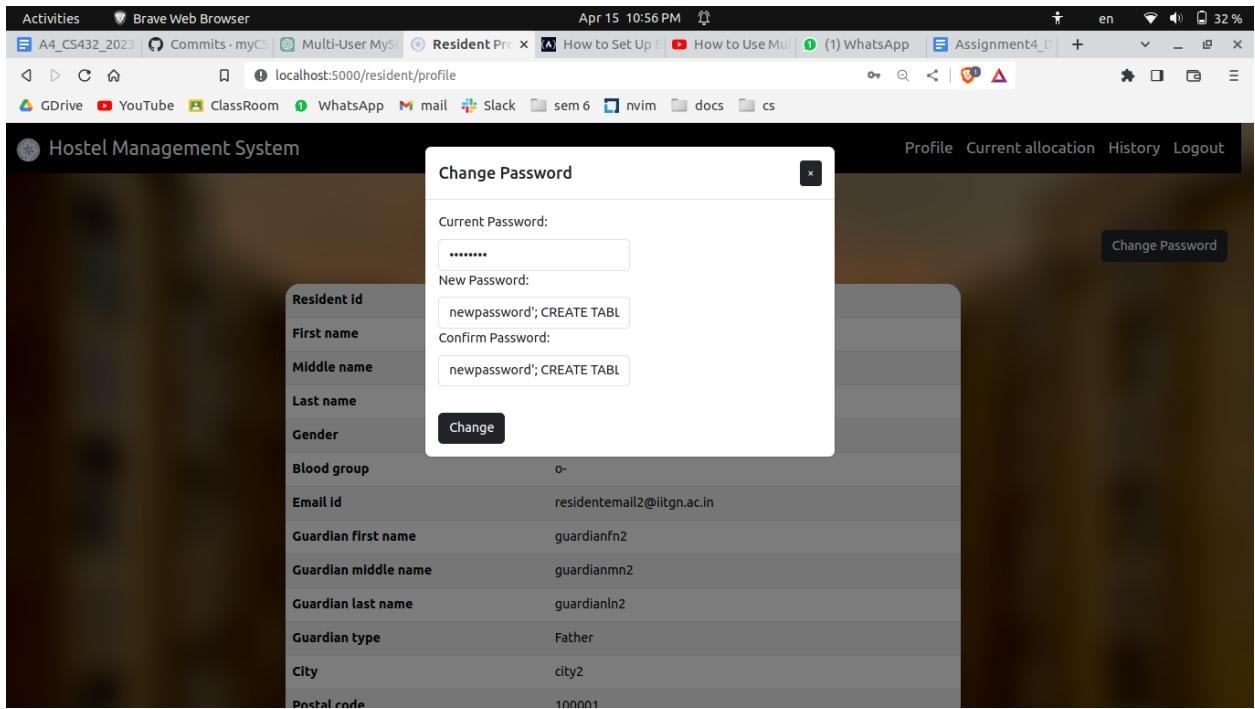


Logging in the admin using sql injection

Injection used: 00000001'; --

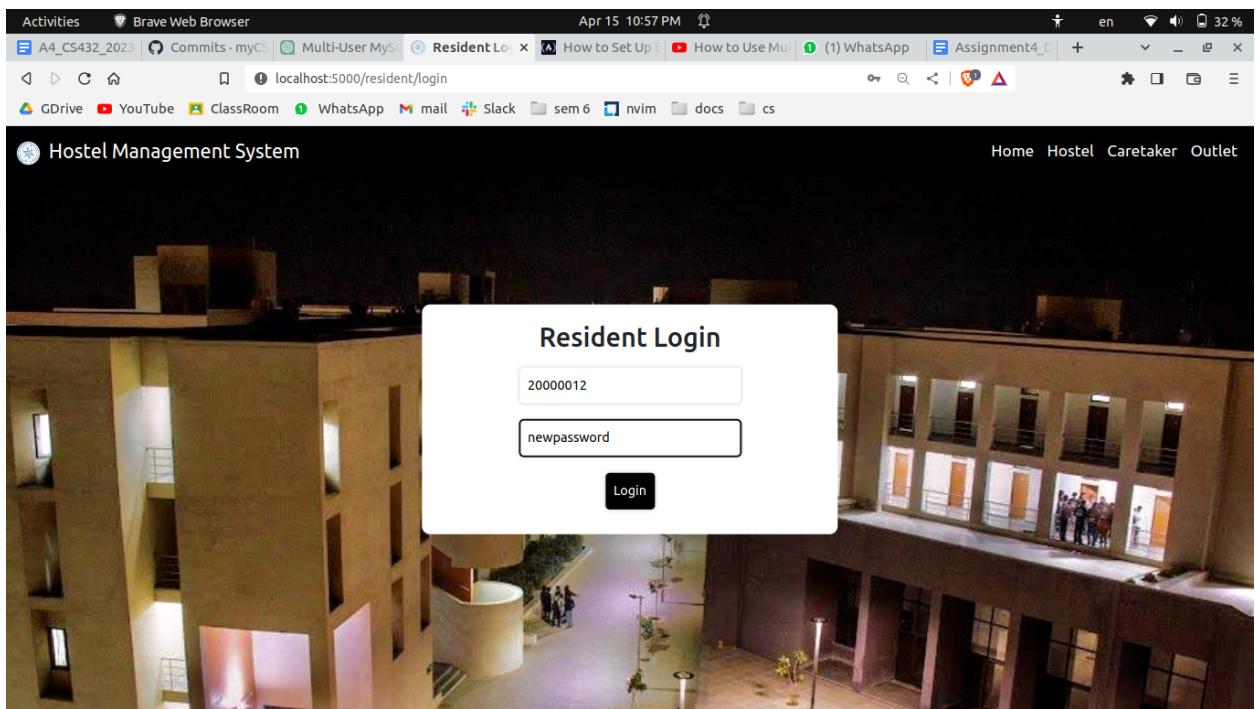
Card	2 students in triple room:	0 students in triple room:	0 students in single room:	1 students in triple room:
A	4	144	144	73
B	0	144	144	0
C	0	144	144	0
D	144	144	144	0
E	0	144	144	0
F	0	144	144	0
G	144	144	144	0
H	0	144	144	0
I	0	144	144	144

After logging in with correct credentials, you can change passwords of all users using the below sql injection script



Injection used: newpassword'; CREATE TABLE buffer; SELECT * FROM users JOIN buffer ON '1' = '1

Changing passwords of all users



Hostel Management System

Resident Profile

[Change Password](#)

Resident Id	20000012
First name	residentfn13
Middle name	residentmn13
Last name	residentln13
Gender	M
Blood group	a-
Email id	residentemail13@iitgn.ac.in
Guardian first name	guardianfn13
Guardian middle name	guardianmn13
Guardian last name	guardianln13
Guardian type	Father
City	city13
Postal code	100012

A screenshot of a web browser showing a Resident Login page. The page has a white background with a central form. At the top center, it says "Resident Login". Below that are two input fields: the first contains the number "20000111" and the second contains "newpassword". At the bottom of the form is a black "Login" button. The background of the entire screen is a photograph of a modern, multi-story building at night, likely a dormitory or hostel, with lights on in some windows and people visible on the balconies. The browser's address bar shows "localhost:5000/resident/login".

The screenshot shows a web browser window titled "Resident Profile" from the "Hostel Management System". The browser's address bar displays "localhost:5000/resident/profile". The page header includes links for "Profile", "Current allocation", "History", and "Logout". A "Change Password" button is located in the top right corner of the main content area. The main content is a table listing resident information:

Resident id	20000111
First name	residentfn112
Middle name	residentmn112
Last name	residentln112
Gender	F
Blood group	a+
Email id	residentemail112@iitgn.ac.in
Guardian first name	guardianfn112
Guardian middle name	guardianmn112
Guardian last name	guardianln112
Guardian type	Father
City	city112
Postal code	100111

Defense:

We used hash generator to generate hash values for the passwords and these generated hash passwords are stored in the database.

XSS Attack:

The screenshot shows a web application for a Hostel Management System. A modal dialog titled "Add Caretaker" is open, prompting for Caretaker ID, Name, Contact, Hostel, and Actions. The "Name" field contains the value "10000001". The "Email ID" field contains the value "<script>alert(document.coc)". The "Actions" table lists five entries, each with a "Delete" button. The "Actions" column for the first entry contains the value "a,h". The developer tools' Elements tab is open, showing the HTML structure of the "Email ID" input field. The computed styles for the "form-control" class are visible on the right.

```
<div class="col-md-4"> </div>
<div class="col-md-4"> </div>
<div class="col-md-4"> </div>
<div class="col-md-6"> </div>
<div class="col-md-6"> </div>
<div class="col-md-6">
  <label form="email_id" class="form-label">Email ID:</label>
  <input type="text" class="form-control" name="email_id" id="email_id" placeholder="Enter email address" required> <span>50</span>
</div>
<!-- Hostel -->
<div class="col-md-6" id="Hostel"> </div>
<div class="col-md-6"> </div>
<div class="col-md-6"> </div>
<div class="col-md-12" style="text-align: center;"> </div>
</form>
```

Styles Computed Layout Event Listeners

Filter :has(.cls)

element.style { }

.form-control { _form-control.scss:6

display: block; width: 100%; padding: 0.375rem 0.75rem; font-size: 1rem; font-weight: 400; line-height: 1.5; color: #var(--bs-body-color); background-color: #var(--bs-form-control-bg); background-clip: padding-box; border: 1px var(--bs-border-width) solid #var(--bs-border-color); }

The screenshot shows a modal dialog with the message "127.0.0.1:5000 says" followed by a long session ID. An "OK" button is at the bottom right. The session ID is: eyJpZCI6IjAwMDAwMDAxIiwibG9nZ2VkX2lujp0cnVlCJuYW1lIjoiYWRtaW4ifQ.ZDrIWw.U3kUr6zOSlamybVvBkAc-LYAuo

Before

A screenshot of Visual Studio Code showing the file structure and content of `app.py`. The code includes MySQL configuration and page definitions. A specific line of code is highlighted:

```
10 import time
11
12 app = Flask(__name__)
13
14 # setting the secret key
15 app.secret_key = 'DONT TELL ANYONE'
16
17 # MySQL configurations
18 app.config['MYSQL_HOST'] = 'localhost'
19 app.config['MYSQL_USER'] = "admin"
20 app.config["MYSQL_PASSWORD"] = "Password@1234"
21 app.config["MYSQL_DB"] = "hostelmng"
22 app.config['SESSION_COOKIE_HTTPONLY'] = True
23
24 mysql = MySQL(app)
25
26 # Defining the pages to support
27 pages = {
28     "Home" : "/home",
29     "Hostel" : "/home/hostel_details",
30     "Caretaker" : "/home/caretaker_details",
31     "Outlet" : "/home/outlet_details",
32 }
33
34 admin_pages = {
```

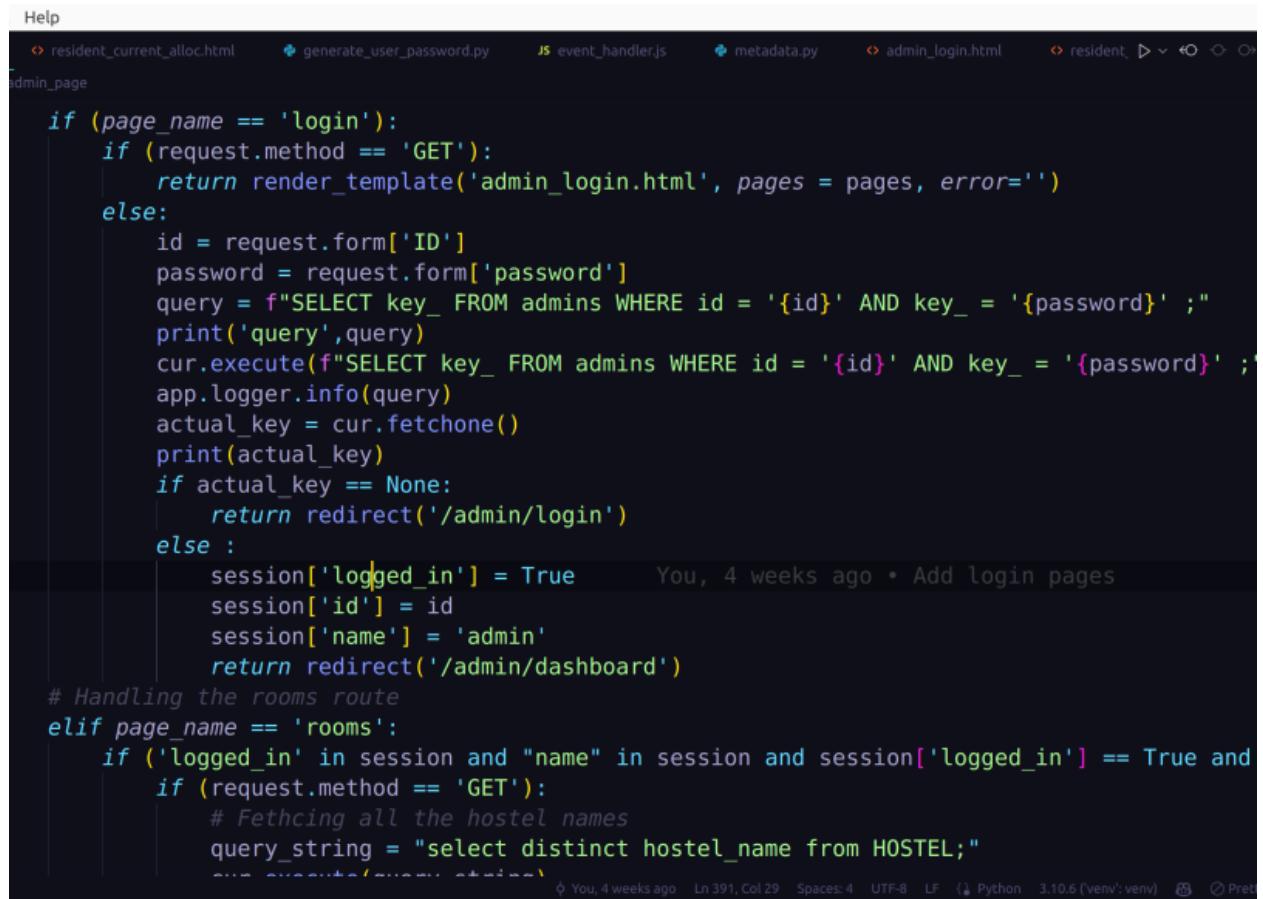
After

A screenshot of Visual Studio Code showing the same `app.py` file as before, but with a different line of code highlighted. This line adds a session cookie configuration to the `app.config` dictionary.

```
10 import time
11
12 app = Flask(__name__)
13
14 # setting the secret key
15 app.secret_key = 'DONT TELL ANYONE'
16
17 # MySQL configurations
18 app.config['MYSQL_HOST'] = 'localhost'
19 app.config['MYSQL_USER'] = "admin"
20 app.config["MYSQL_PASSWORD"] = "Password@1234"
21 app.config["MYSQL_DB"] = "hostelmng"
22 app.config['SESSION_COOKIE_HTTPONLY'] = True
23
24 mysql = MySQL(app)
25
26 # Defining the pages to support
27 pages = {
28     "Home" : "/home",
29     "Hostel" : "/home/hostel_details",
30     "Caretaker" : "/home/caretaker_details",
31     "Outlet" : "/home/outlet_details",
32 }
33
34 admin_pages = {
```

Dictionary attack:

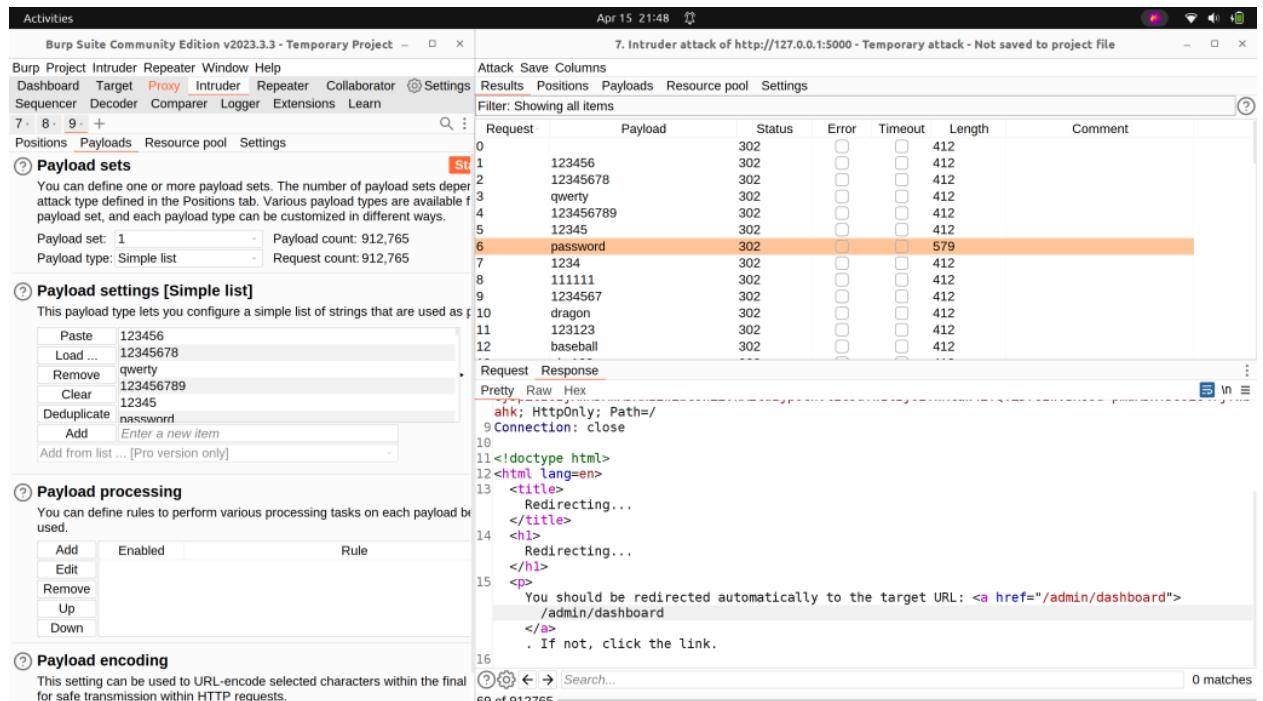
A. Attack : The unresolved code for the dictionary attack



```
if (page_name == 'login'):
    if (request.method == 'GET'):
        return render_template('admin_login.html', pages = pages, error='')
    else:
        id = request.form['ID']
        password = request.form['password']
        query = f"SELECT key_ FROM admins WHERE id = '{id}' AND key_ = '{password}' ;"
        print('query',query)
        cur.execute(f"SELECT key_ FROM admins WHERE id = '{id}' AND key_ = '{password}' ;")
        app.logger.info(query)
        actual_key = cur.fetchone()
        print(actual_key)
        if actual_key == None:
            return redirect('/admin/login')
        else :
            session['logged_in'] = True      You, 4 weeks ago • Add login pages
            session['id'] = id
            session['name'] = 'admin'
            return redirect('/admin/dashboard')

# Handling the rooms route
elif page_name == 'rooms':
    if ('logged_in' in session and "name" in session and session['logged_in'] == True and
        if (request.method == 'GET'):

        # Fetching all the hostel names
        query_string = "select distinct hostel_name from HOSTEL;"
```



Burp Suite Community Edition v2023.3.3 - Temporary Project - □ ×

Attack Save Columns

Results Positions Payloads Resource pool Settings

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
0	123456	302	<input type="checkbox"/>	<input type="checkbox"/>	412	
1	12345678	302	<input type="checkbox"/>	<input type="checkbox"/>	412	
2	qwerty	302	<input type="checkbox"/>	<input type="checkbox"/>	412	
3	123456789	302	<input type="checkbox"/>	<input type="checkbox"/>	412	
4	12345	302	<input type="checkbox"/>	<input type="checkbox"/>	412	
5	12345	302	<input type="checkbox"/>	<input type="checkbox"/>	412	
6	password	302	<input type="checkbox"/>	<input type="checkbox"/>	579	
7	1234	302	<input type="checkbox"/>	<input type="checkbox"/>	412	
8	111111	302	<input type="checkbox"/>	<input type="checkbox"/>	412	
9	1234567	302	<input type="checkbox"/>	<input type="checkbox"/>	412	
10	dragon	302	<input type="checkbox"/>	<input type="checkbox"/>	412	
11	123123	302	<input type="checkbox"/>	<input type="checkbox"/>	412	
12	baseball	302	<input type="checkbox"/>	<input type="checkbox"/>	412	

② Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.

Payload set: 1 Payload count: 912,765

Payload type: Simple list Request count: 912,765

② Payload settings [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste	123456
Load ...	12345678
Remove	qwerty
Clear	123456789
Deduplicate	password
Add	Enter a new item

Add from list ... [Pro version only]

② Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Add	Enabled	Rule
Up		
Down		

② Payload encoding

This setting can be used to URL-encode selected characters within the final URL for safe transmission within HTTP requests.

16

17 `ahk; HttpOnly; Path=/`

18 `Connection: close`

19

20 `11<!DOCTYPE html>`

21 `12<html lang=en>`

22 `13 <title>`

23 `Redirecting...`

24 `</title>`

25 `14 <h1>`

26 `Redirecting...`

27 `</h1>`

28 `15 <p>`

29 `You should be redirected automatically to the target URL: `

30 `/admin/dashboard`

31 ``

32 `. If not, click the link.`

33 `</p>`

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

B. Defense: The resolved code with the defense for the dictionary attack

```

    return {"success": False, "reload": False, "message": str(e.args[1])}

if (page_name == 'login'):
    if (request.method == 'GET'):
        return render_template('admin_login.html', pages = pages, error='')
    else:
        id = request.form['ID']
        password = request.form['password']
        import time
        # Preventing multiple logins(after 3 attempts) for 10 mins
        if (id in resident_login_attempts):
            if (resident_login_attempts[id][0] >= 3):
                if (time.time() - resident_login_attempts[id][1] < 600):
                    return render_template('admin_login.html', pages = pages, error='Too many attempts')
                else:
                    resident_login_attempts[id] = [0, time.time()]
            else:
                resident_login_attempts[id] = [0, time.time()]
                resident_login_attempts[id][1] = time.time()
                resident_login_attempts[id][0] += 1

        else:
            resident_login_attempts[id] = [0, time.time()]
        if resident_login_attempts[id][0] < 3:
            query = f"SELECT key_ FROM admins WHERE id = '{id}' AND key_ = '{password}' ;"
            print(query)
            cur.execute(f"SELECT key_ FROM admins WHERE id = '{id}' AND key_ = '{password}' ;")

```

Request	Payload	Status	Error	Timeout	Length	Comment
0	123456	302			412	
1	12345678	302			412	
2	qwerty	200			5985	
3	123456789	200			5985	
4	12345	200			5985	
5	password	200			5985	
6	1234	200			5985	
7	111111	200			5985	
8	1234567	200			5985	
9	dragon	200			5985	
10	123123	200			5985	
11	200			5985	

Attack Save Columns
Results Positions Payloads Resource pool Settings
Filter: Showing all items

② Payload sets
You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, and each payload type can be customized in different ways.
Payload set: 1 Payload count: 912,765 Request count: 912,765
Payload type: Simple list

② Payload settings [Simple list]
The payload type lets you configure a simple list of strings that are used as payloads.
Paste 123456 Load ... 12345678 Remove qwerty Clear 12345 Deduplicate password Add Enter a new item Add from list ... [Pro version only]

② Payload processing
You can define rules to perform various processing tasks on each payload before it is used.
Add Enabled Rule
Edit Remove Up Down

② Payload encoding
This setting can be used to URL-encode selected characters within the final payload for safe transmission within HTTP requests.

Note: The scheme is attached in the github repo.

CONTRIBUTIONS:

G1:

Kareena
Bhavini
Hamsini
Rajesh
Gnana Sai
Chaitanya

G2:

Sriman
Manish
Sunny
Siva Sai
Karthik
Rishab