RESPONSIBILITY OF G1:

FEEDBACK FROM THE STAKEHOLDER

Before the first feedback:

1. No housekeeping section is available

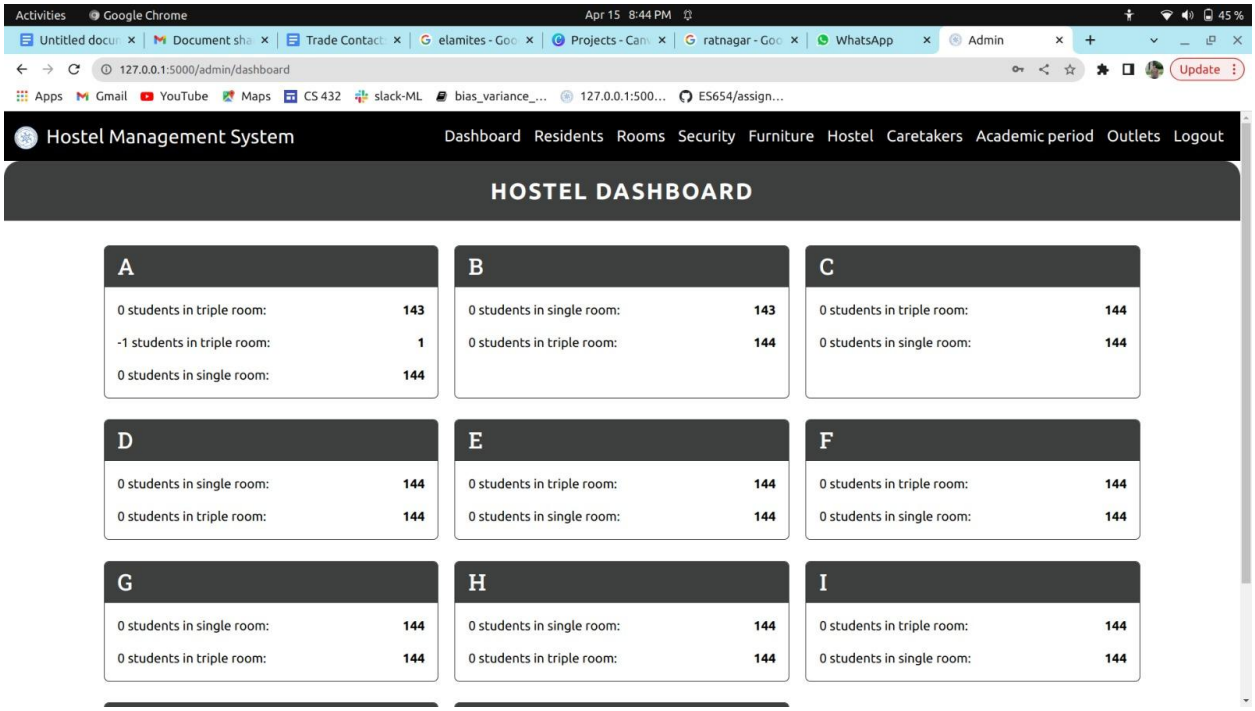


Fig.1

2. Import data from Excel is not included

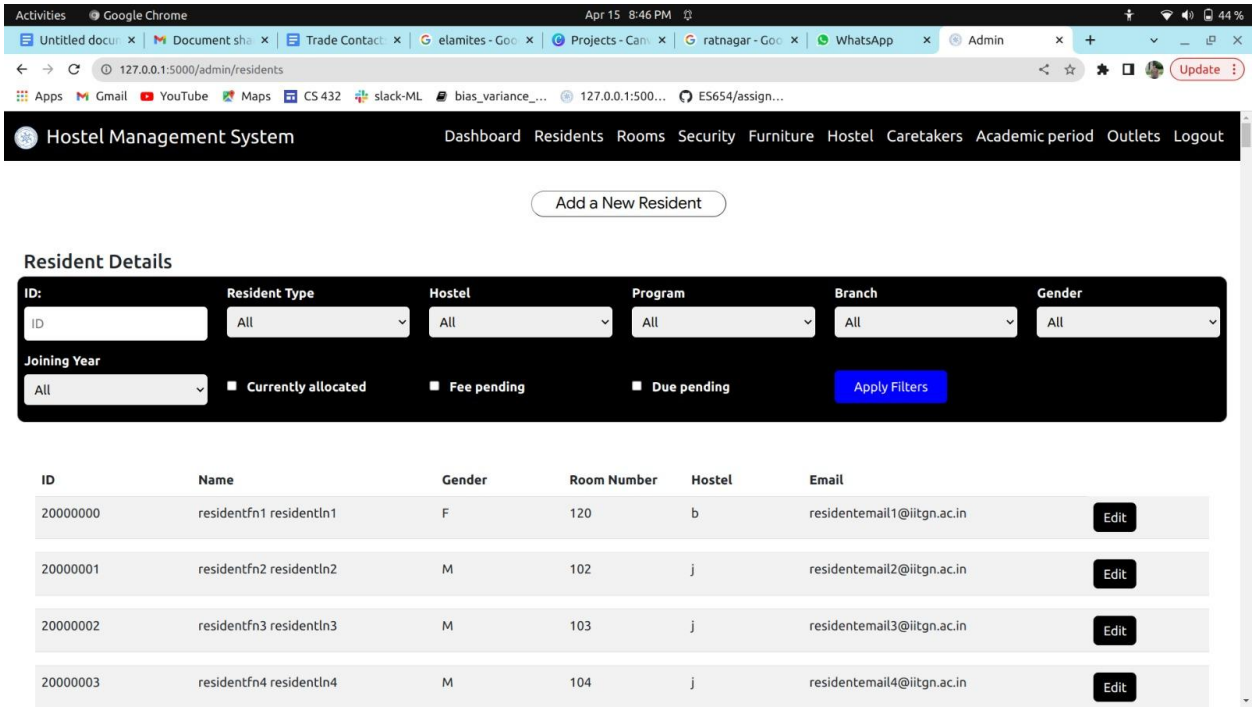


Fig.2

3. Details of the resident guardian are not available

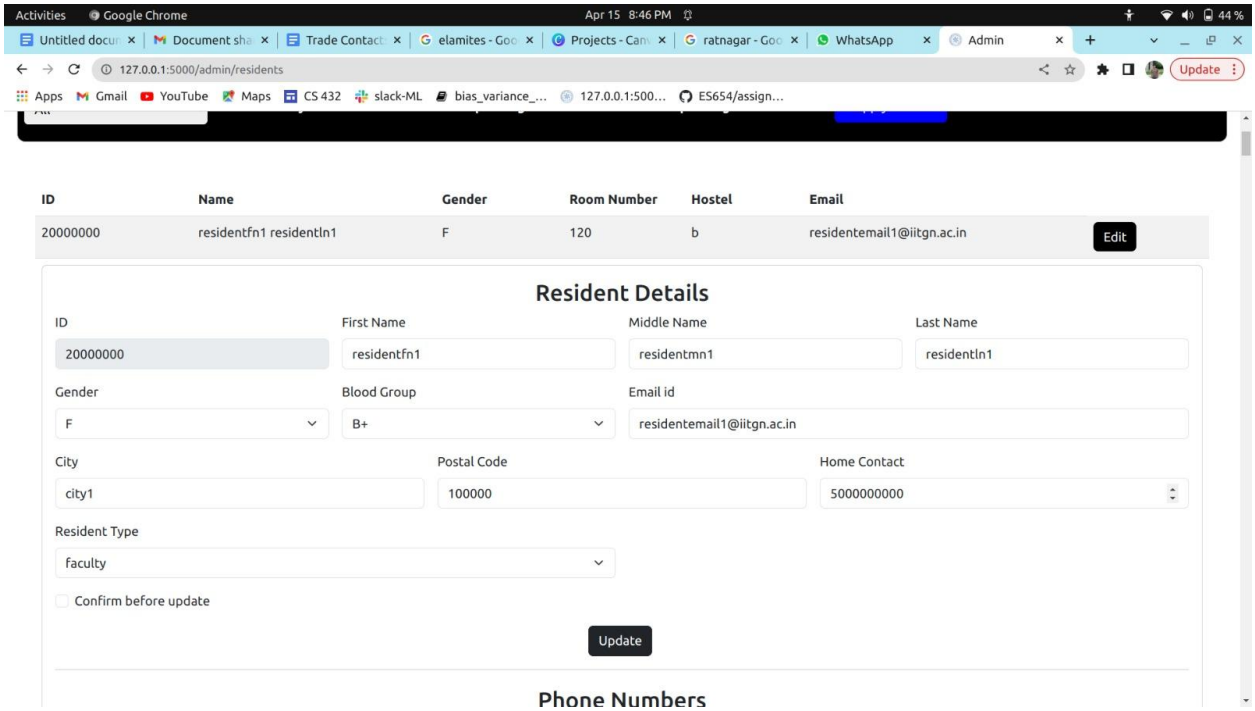


Fig.3

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.

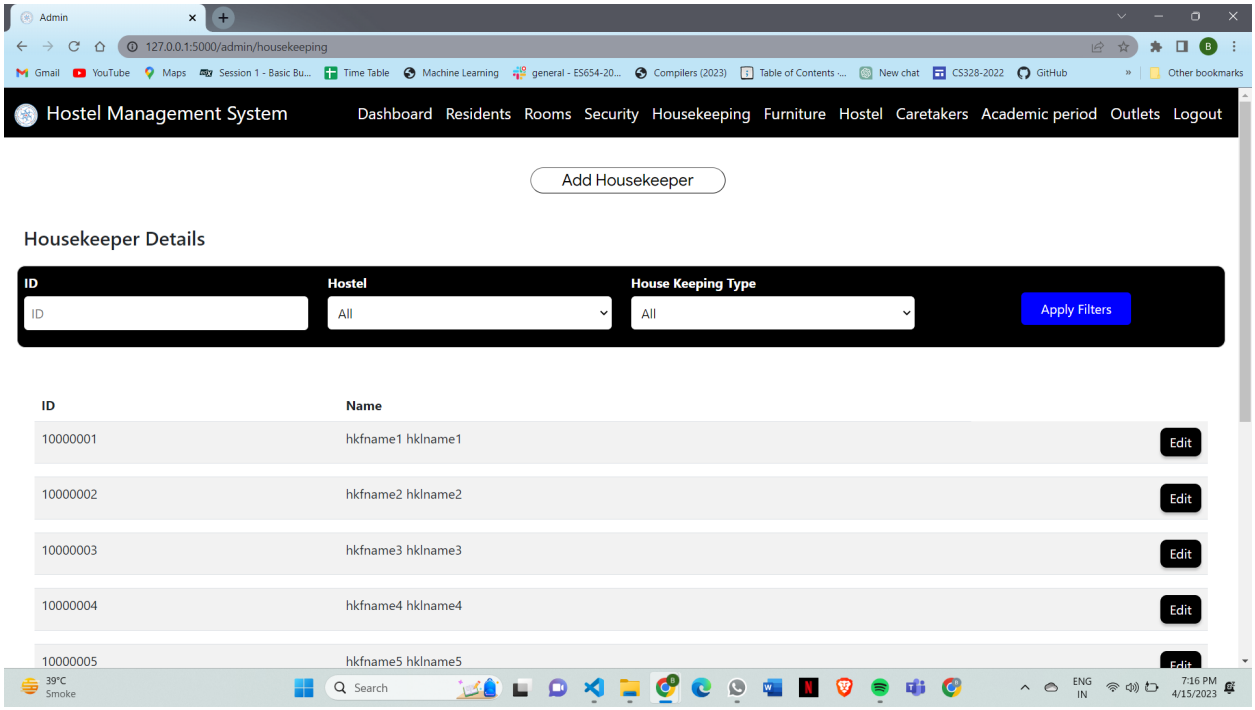


Fig.4: Admin view of the housekeeping section.

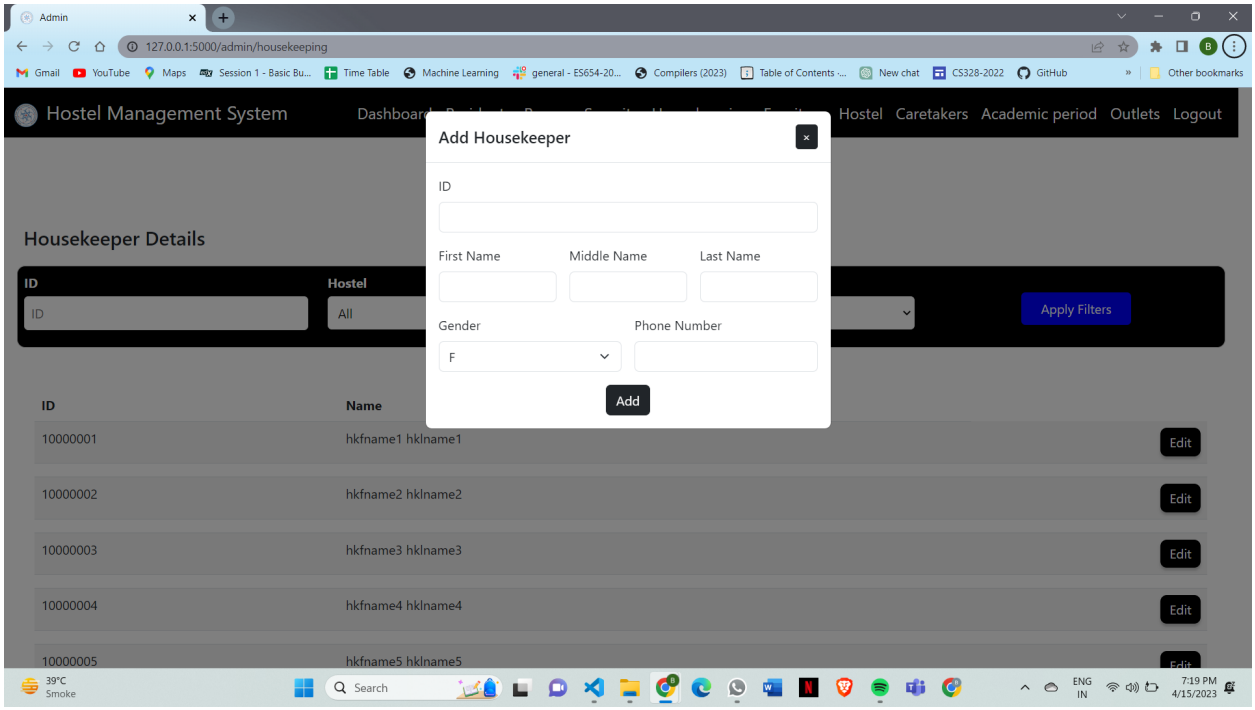


Fig.5: Additing of a housekeeping staff member

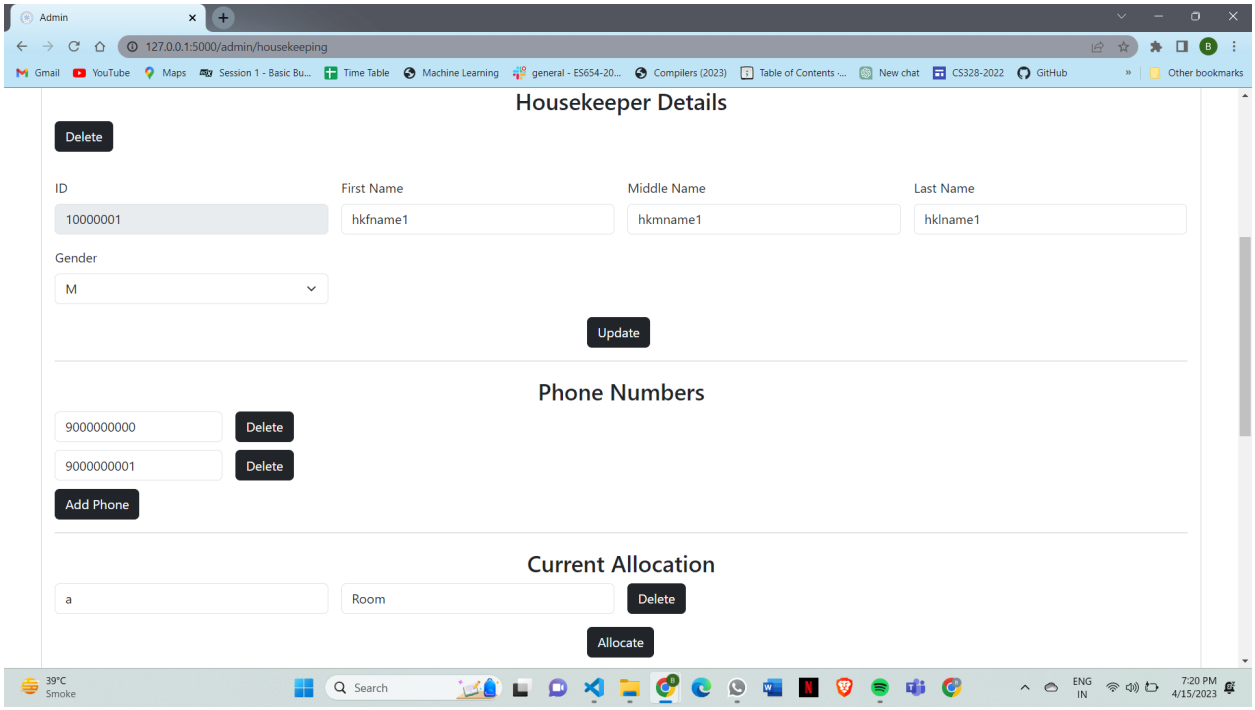


Fig.6: Updation, deletion, and current room allocation

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

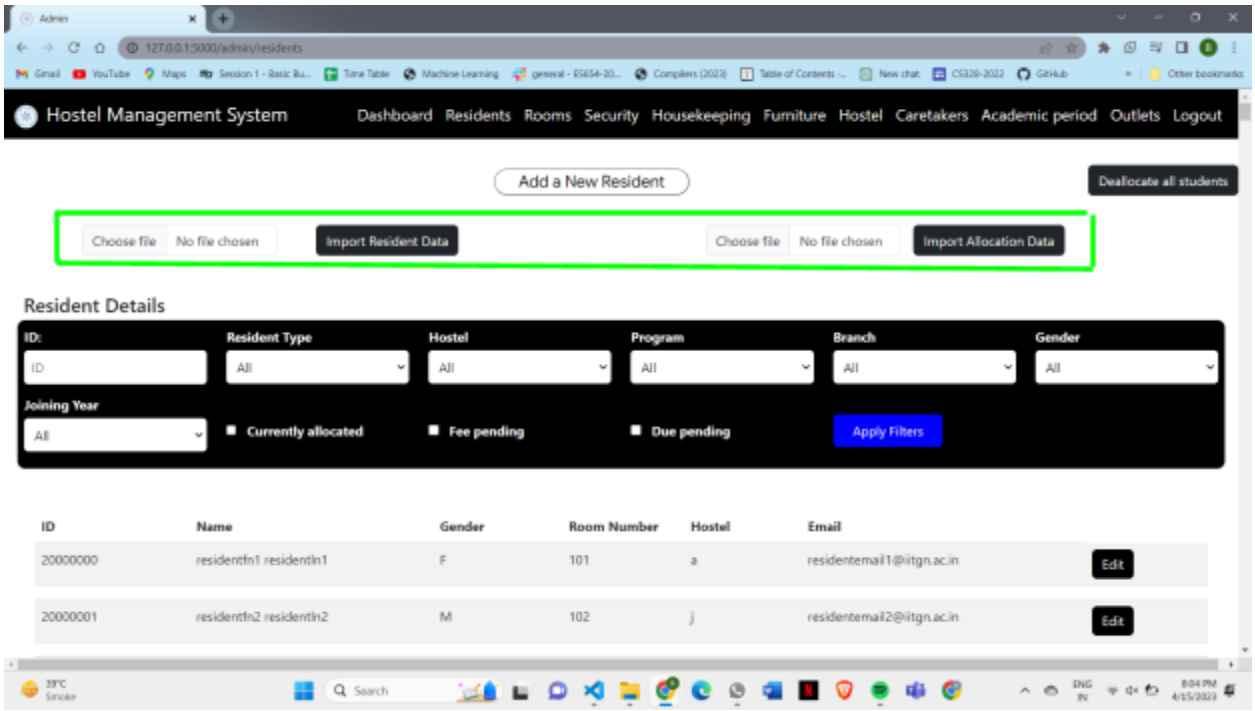


Fig.7: An option to include the resident in the form of a CSV file

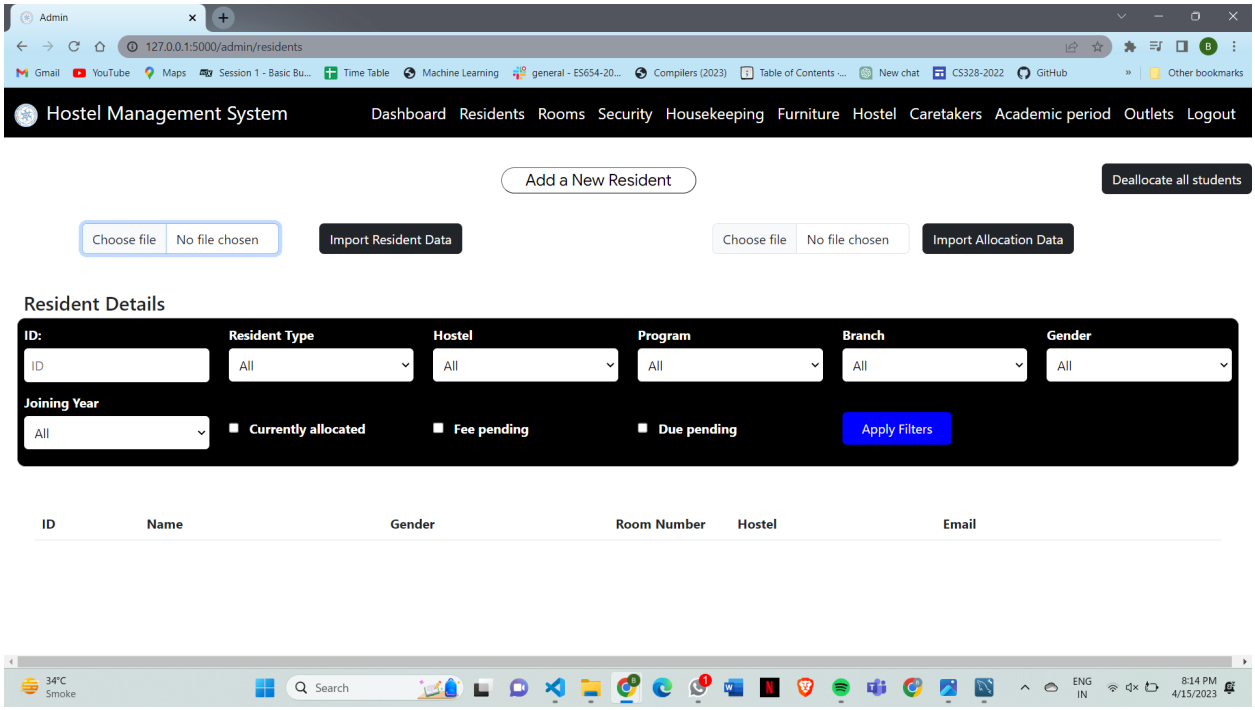


Fig.8: Before importing data

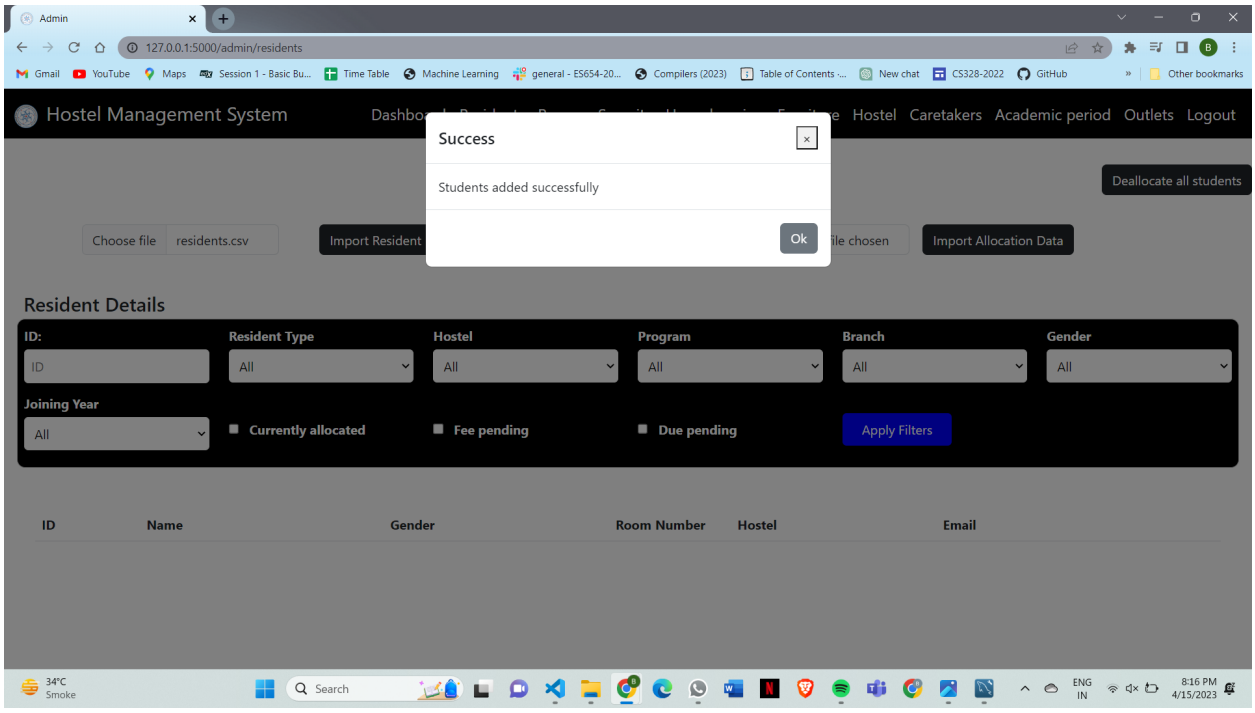


Fig.9: A pop to denote the successful importing of data

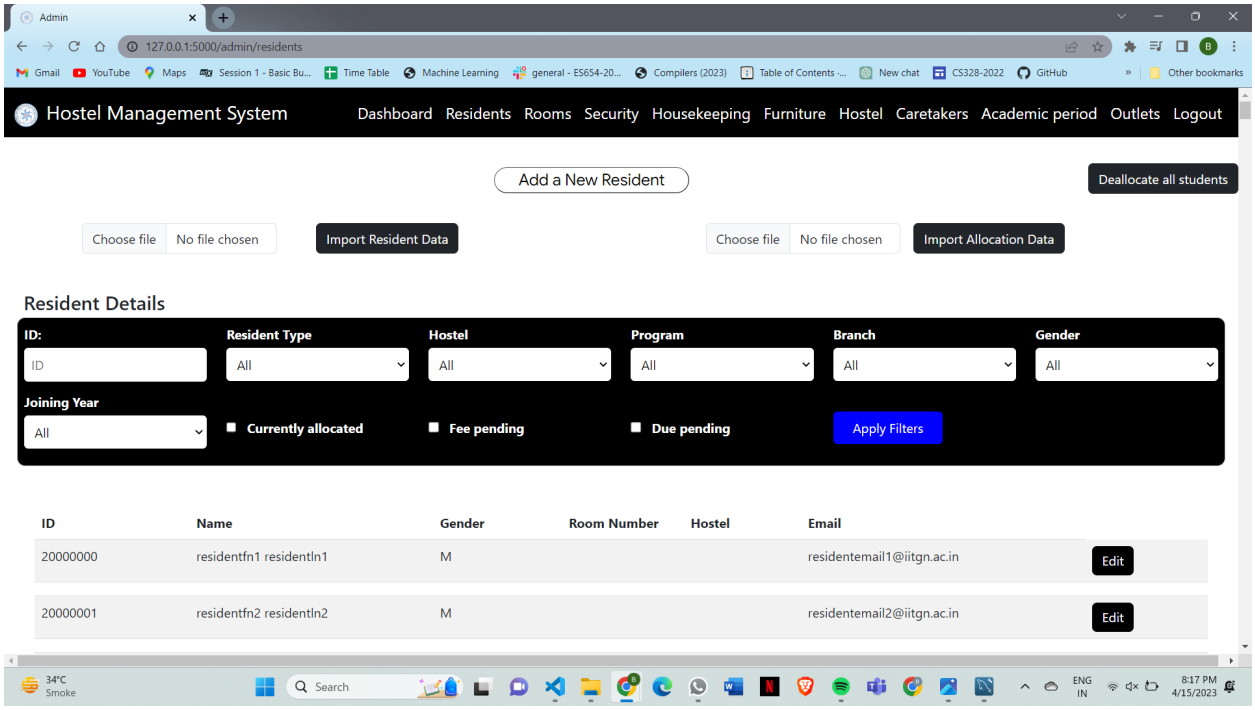


Fig.10: After importing the resident data

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

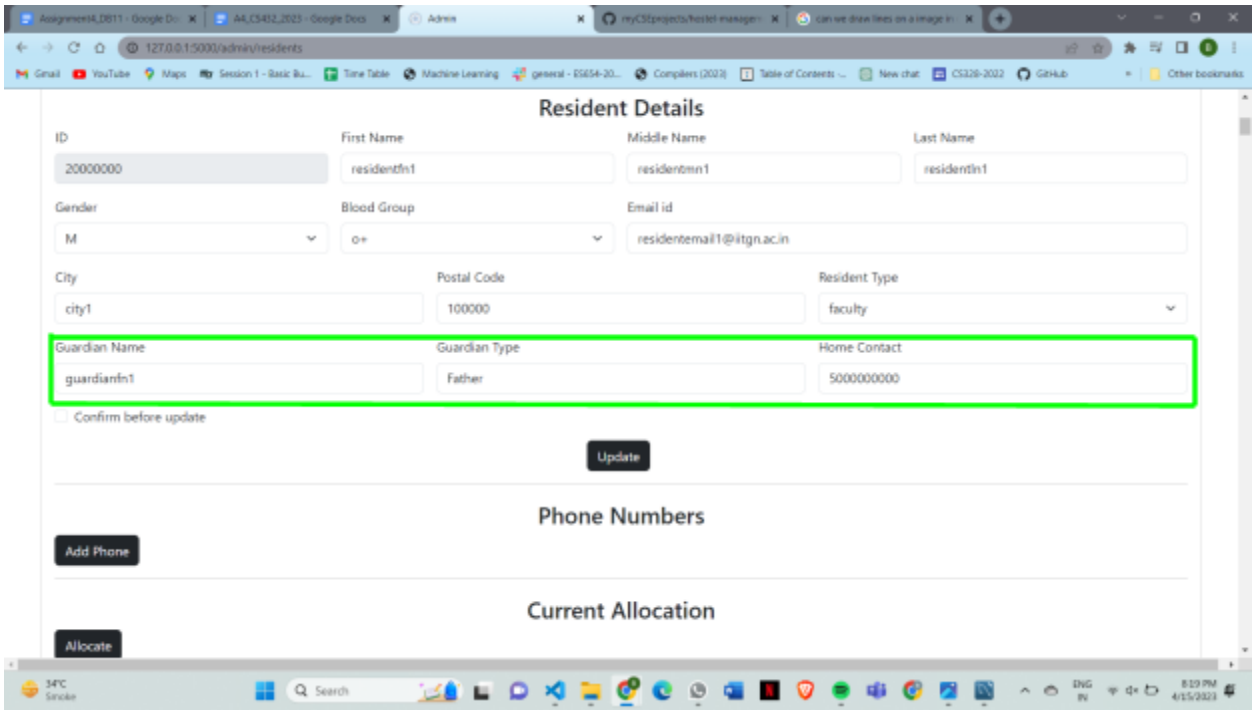
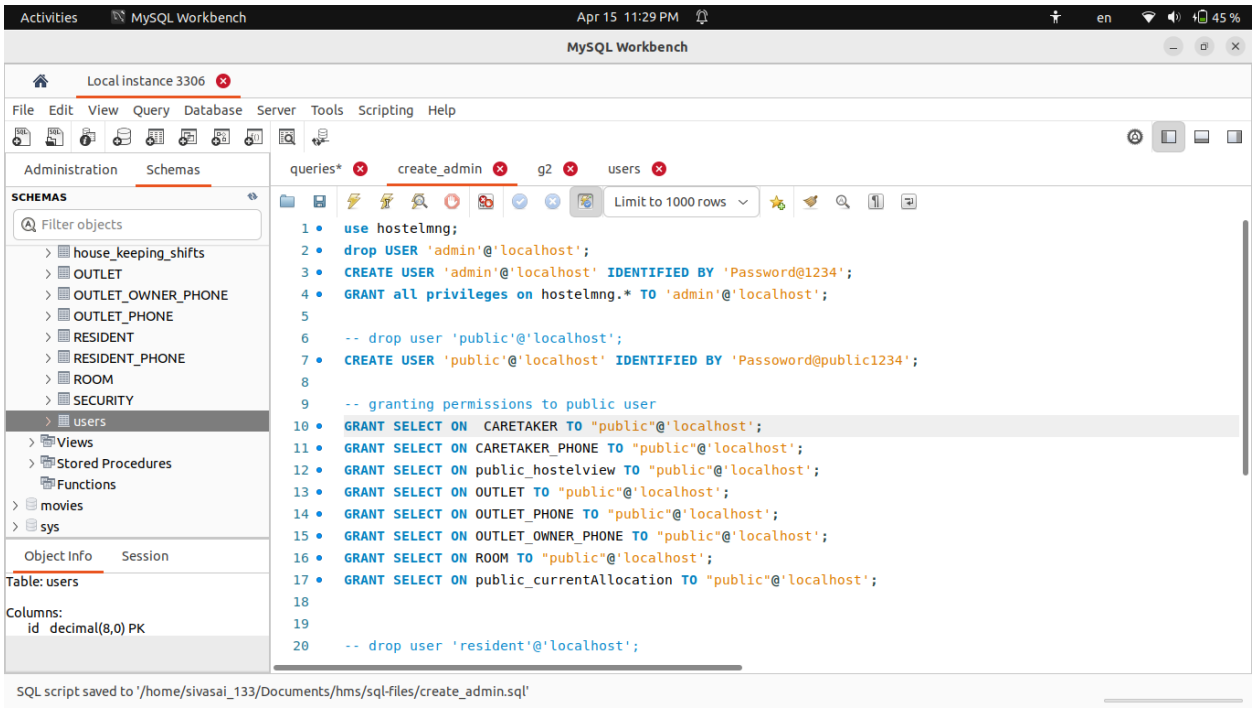
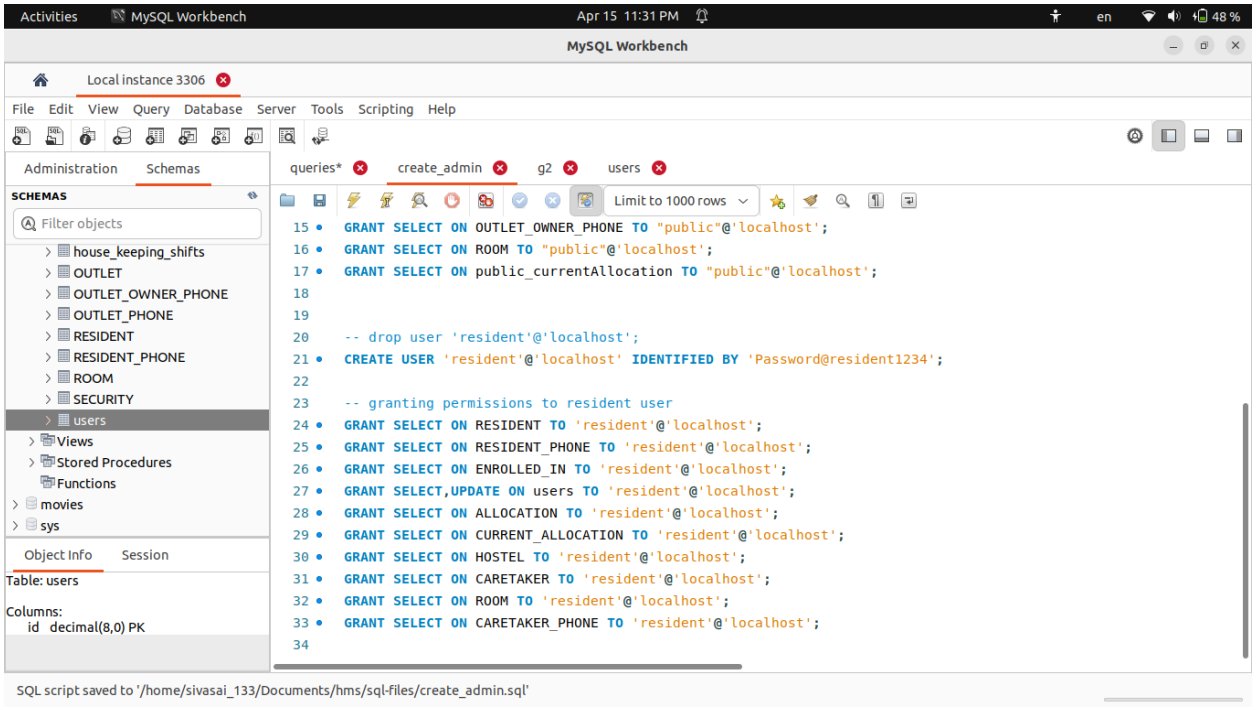


Fig.11

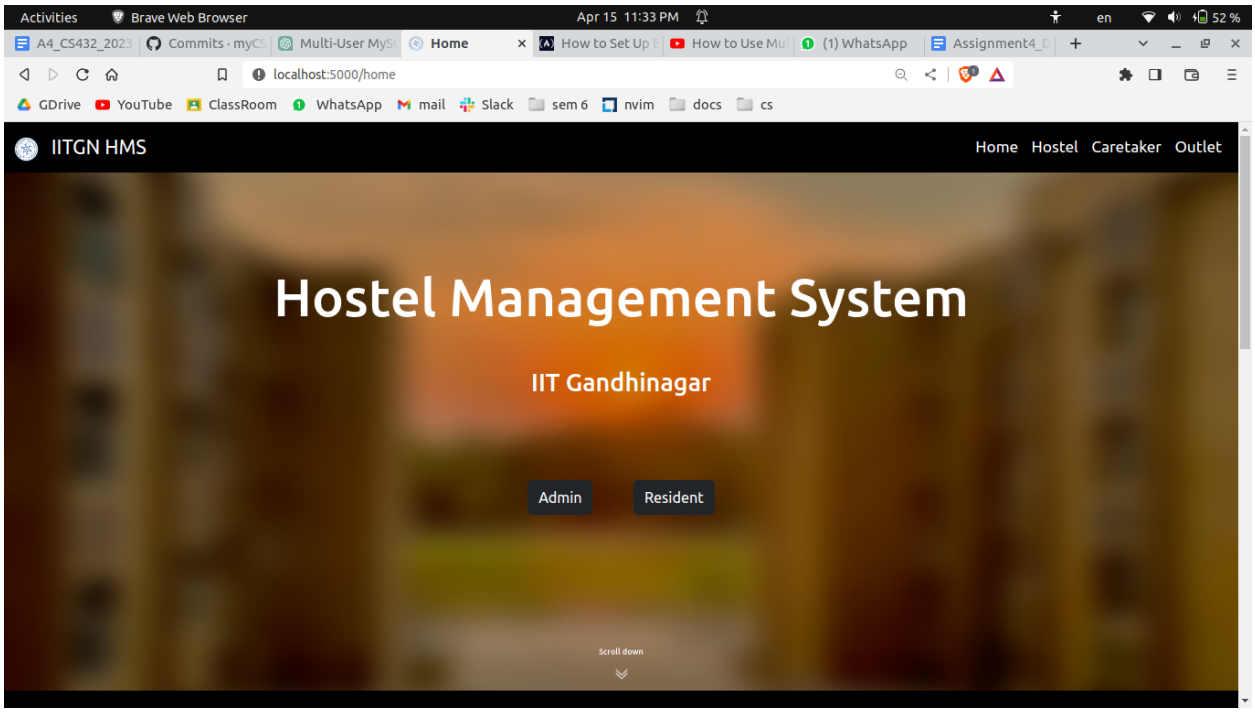
Different Views

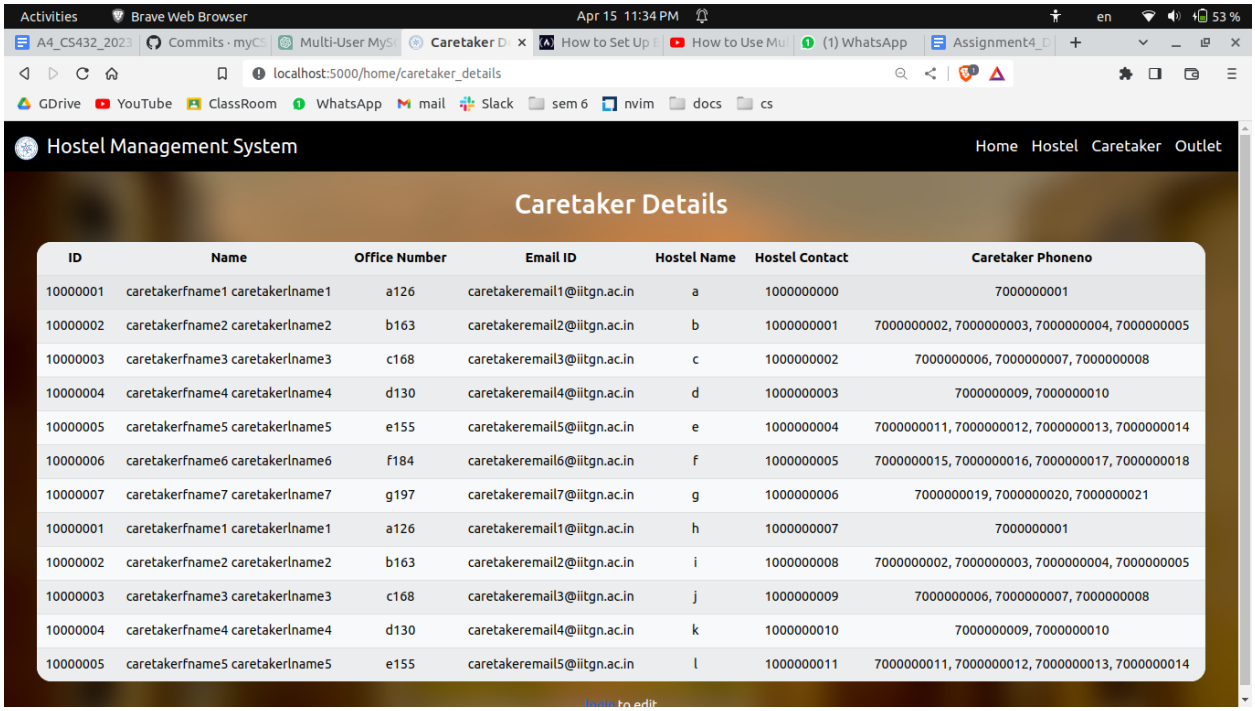
Generating multiple users and views and granting permissions to them.





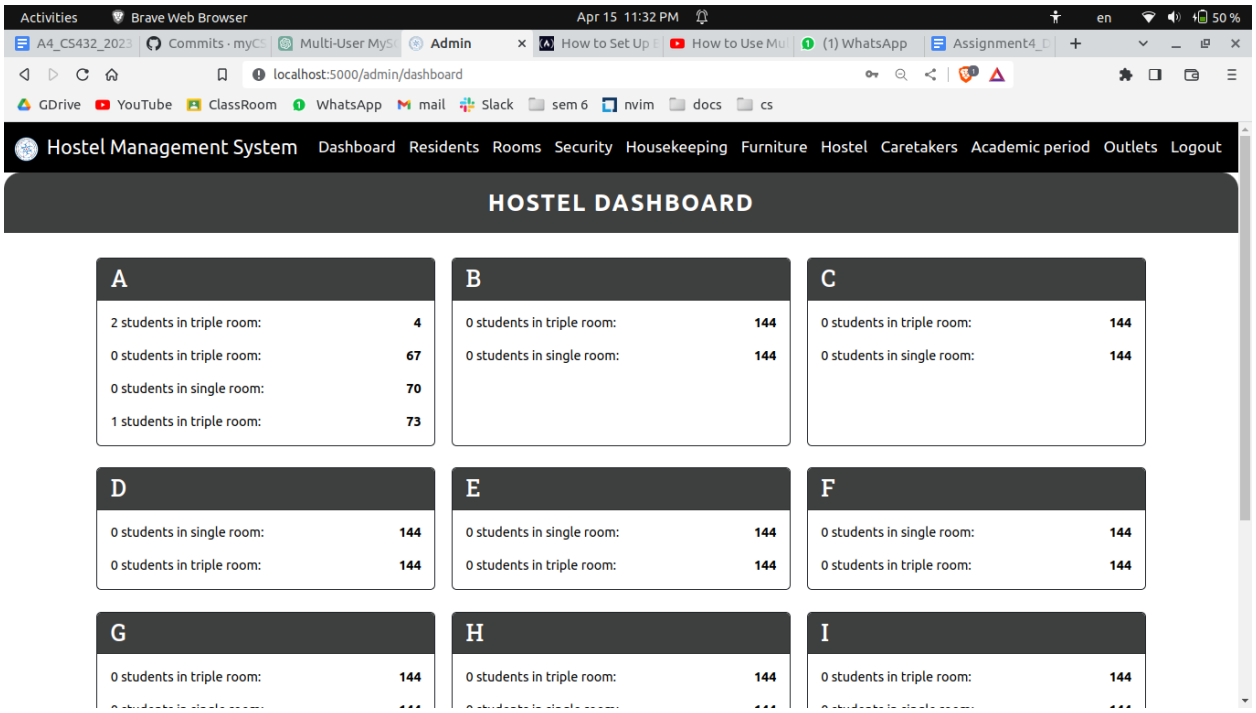
Public view

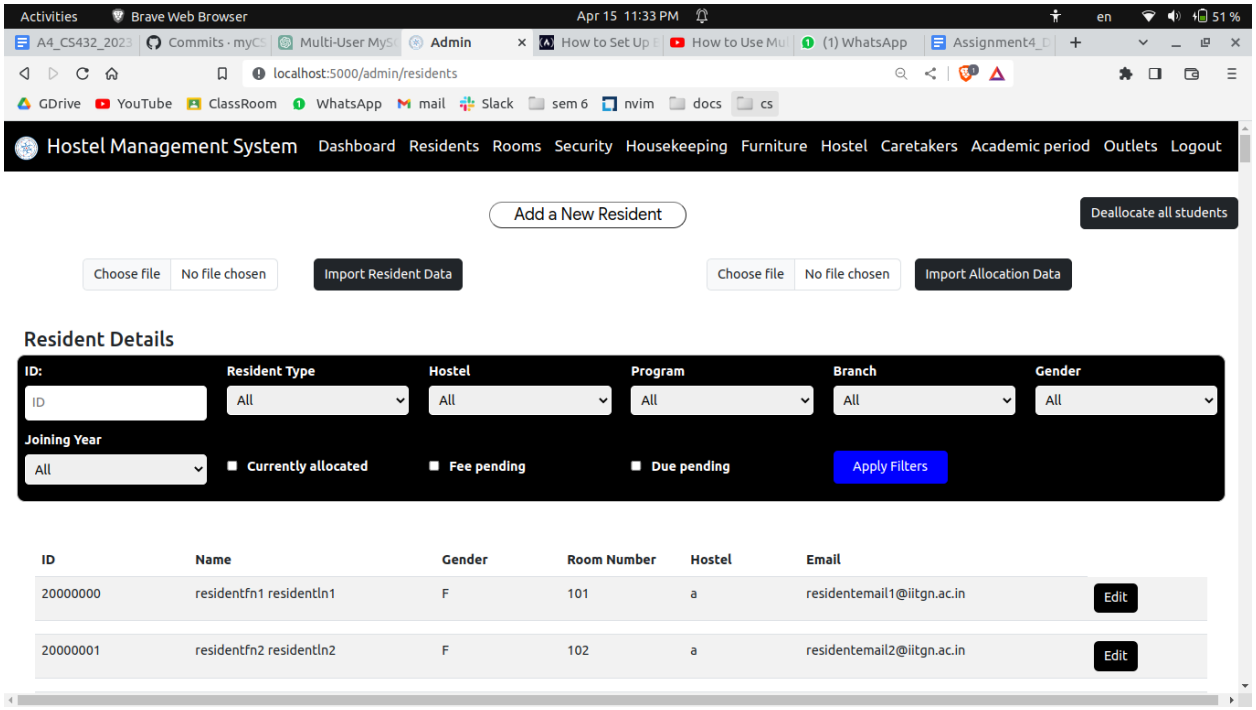




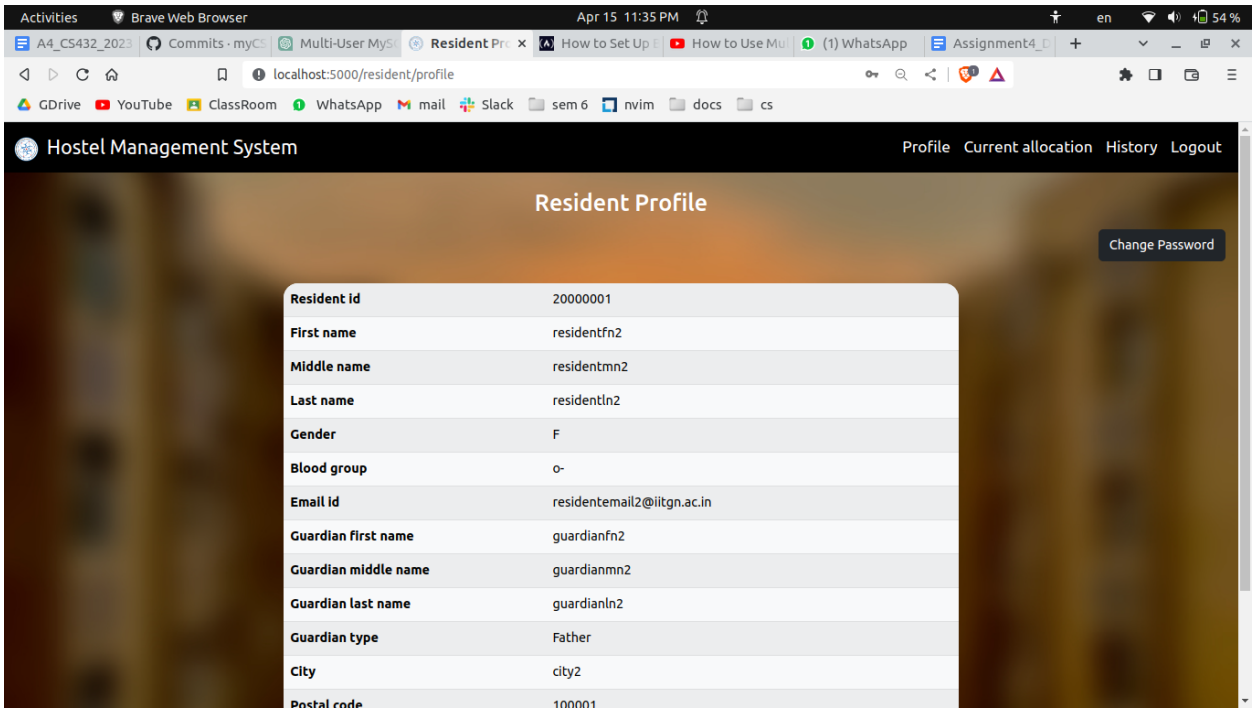
ID	Name	Office Number	Email ID	Hostel Name	Hostel Contact	Caretaker Phoneno
10000001	caretakerfname1 caretakerlname1	a126	caretakeremail1@iitgn.ac.in	a	1000000000	7000000001
10000002	caretakerfname2 caretakerlname2	b163	caretakeremail2@iitgn.ac.in	b	1000000001	7000000002, 7000000003, 7000000004, 7000000005
10000003	caretakerfname3 caretakerlname3	c168	caretakeremail3@iitgn.ac.in	c	1000000002	7000000006, 7000000007, 7000000008
10000004	caretakerfname4 caretakerlname4	d130	caretakeremail4@iitgn.ac.in	d	1000000003	7000000009, 7000000010
10000005	caretakerfname5 caretakerlname5	e155	caretakeremail5@iitgn.ac.in	e	1000000004	7000000011, 7000000012, 7000000013, 7000000014
10000006	caretakerfname6 caretakerlname6	f184	caretakeremail6@iitgn.ac.in	f	1000000005	7000000015, 7000000016, 7000000017, 7000000018
10000007	caretakerfname7 caretakerlname7	g197	caretakeremail7@iitgn.ac.in	g	1000000006	7000000019, 7000000020, 7000000021
10000001	caretakerfname1 caretakerlname1	a126	caretakeremail1@iitgn.ac.in	h	1000000007	7000000001
10000002	caretakerfname2 caretakerlname2	b163	caretakeremail2@iitgn.ac.in	i	1000000008	7000000002, 7000000003, 7000000004, 7000000005
10000003	caretakerfname3 caretakerlname3	c168	caretakeremail3@iitgn.ac.in	j	1000000009	7000000006, 7000000007, 7000000008
10000004	caretakerfname4 caretakerlname4	d130	caretakeremail4@iitgn.ac.in	k	1000000010	7000000009, 7000000010
10000005	caretakerfname5 caretakerlname5	e155	caretakeremail5@iitgn.ac.in	l	1000000011	7000000011, 7000000012, 7000000013, 7000000014

Admin view


Activities
Brave Web Browser
Apr 15 11:32 PM
en
50 %
Admin
localhost:5000/admin/dashboard
GDrive
YouTube
ClassRoom
WhatsApp
mail
Slack
sem 6
nvim
docs
cs
Hostel Management System
Dashboard
Residents
Rooms
Security
Housekeeping
Furniture
Hostel
Caretakers
Academic period
Outlets
Logout
HOSTEL DASHBOARD
A
2 students in triple room: 4
0 students in triple room: 67
0 students in single room: 70
1 students in triple room: 73
B
0 students in triple room: 144
0 students in single room: 144
C
0 students in triple room: 144
0 students in single room: 144
D
0 students in single room: 144
0 students in triple room: 144
E
0 students in single room: 144
0 students in triple room: 144
F
0 students in single room: 144
0 students in triple room: 144
G
0 students in triple room: 144
H
0 students in triple room: 144
I
0 students in triple room: 144


Activities
Brave Web Browser
Apr 15 11:33 PM
en
51 %
Admin
localhost:5000/admin/residents
GDrive
YouTube
ClassRoom
WhatsApp
mail
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
Choose file
No file chosen
Import Resident Data
Choose file
No file chosen
Import Allocation Data
Resident Details
ID:
Resident Type
Hostel
Program
Branch
Gender
All
Joining Year
Currently allocated
Fee pending
Due pending
Apply Filters
ID
Name
Gender
Room Number
Hostel
Email
20000000
residentfn1 residentln1
F
101
a
residentemail1@iitgn.ac.in
Edit
20000001
residentfn2 residentln2
F
102
a
residentemail2@iitgn.ac.in
Edit

Resident view



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 commiting the changes
    cur.execute(query_string)
    mysql.connection.commit()
    return {"success": True, "reload":True, "message": "Succesfully 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 commiting the changes
    cur.execute(query_string)
    mysql.connection.commit()
    return {"success": True, "reload":True, "message": "Succesfully 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 commiting the changes
            cur.execute(query_string)
            mysql.connection.commit()
            return {"success": True, "reload":True, "message": "Succesfully 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()
```

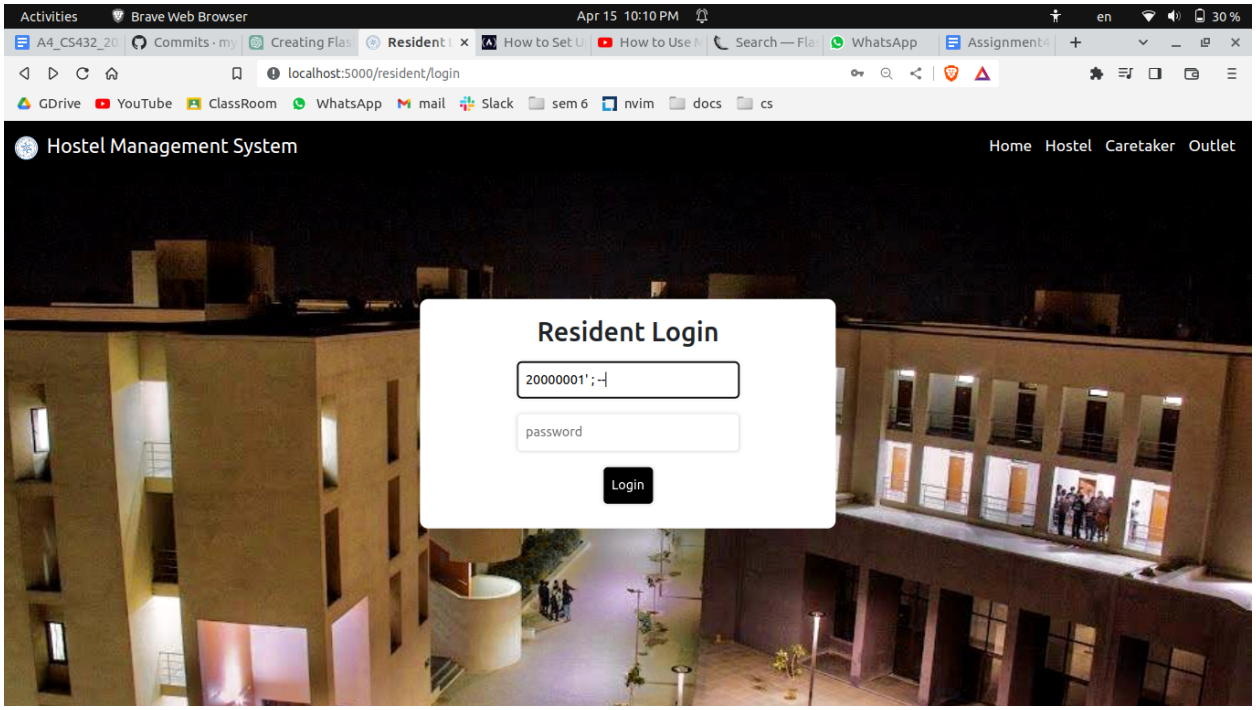
2)

```
try:
    #Locking table Current_allocation
    cur.execute("LOCK TABLES CURRENT_ALLOCATION WRITE")
    # Executing the query and commiting the changes
    cur.execute(query_string)
    mysql.connection.commit()
    return {"success": True, "reload":True, "message": "Succesfully 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()
```

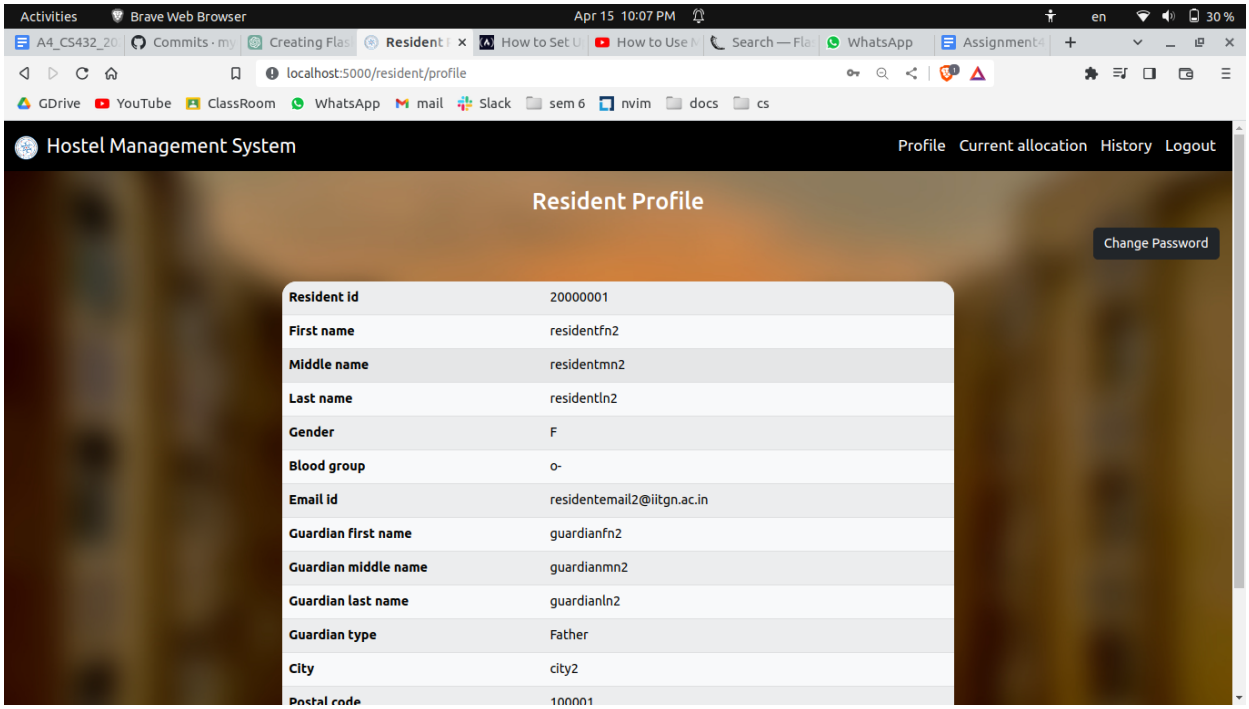
Locks have been implemented at all UPDATE, INSERT, DELETE, and other SQL queries to support concurrency.

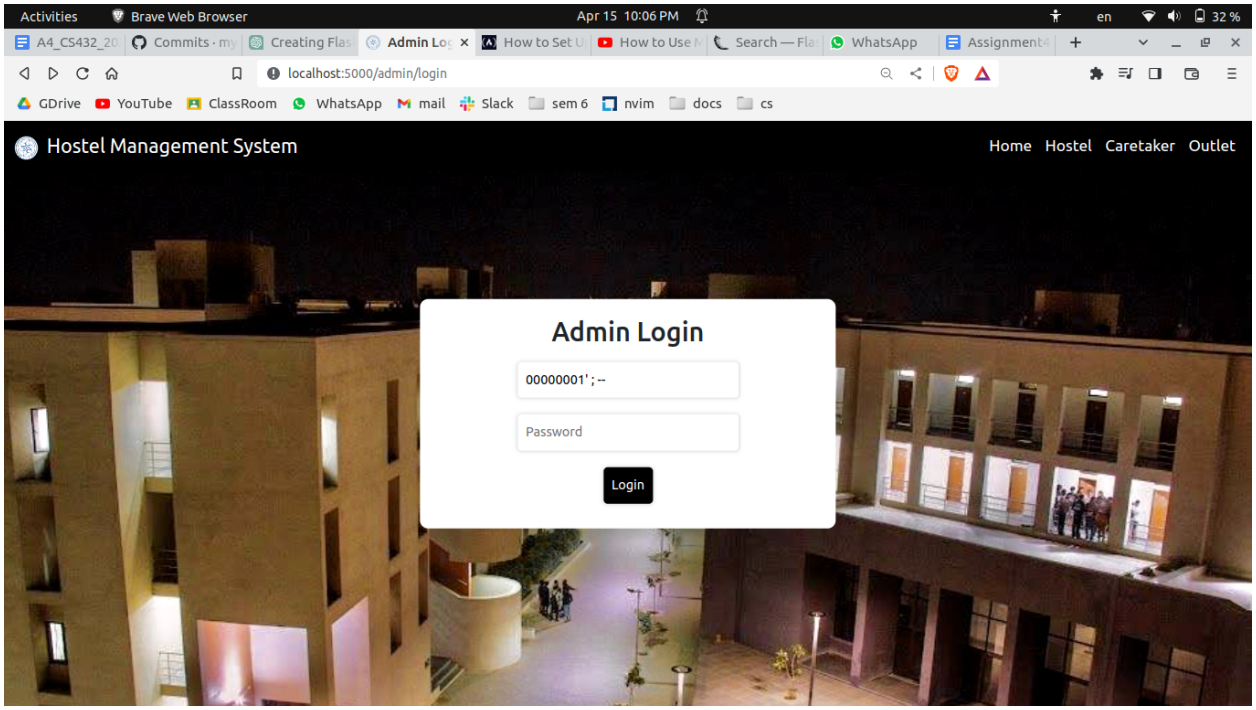
RESPONSIBILITY OF G1&G2:

SQL Injection:

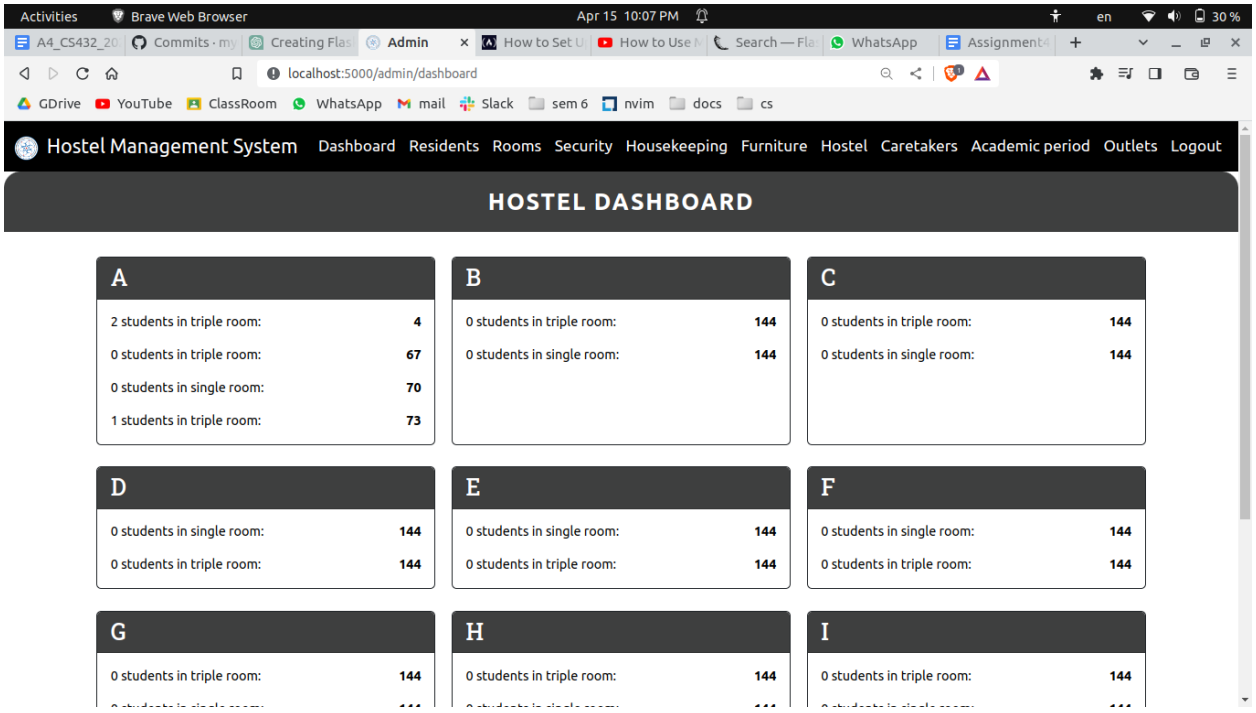


Logging in the resident using SQL injection. Injection used: 20000001’ ; --

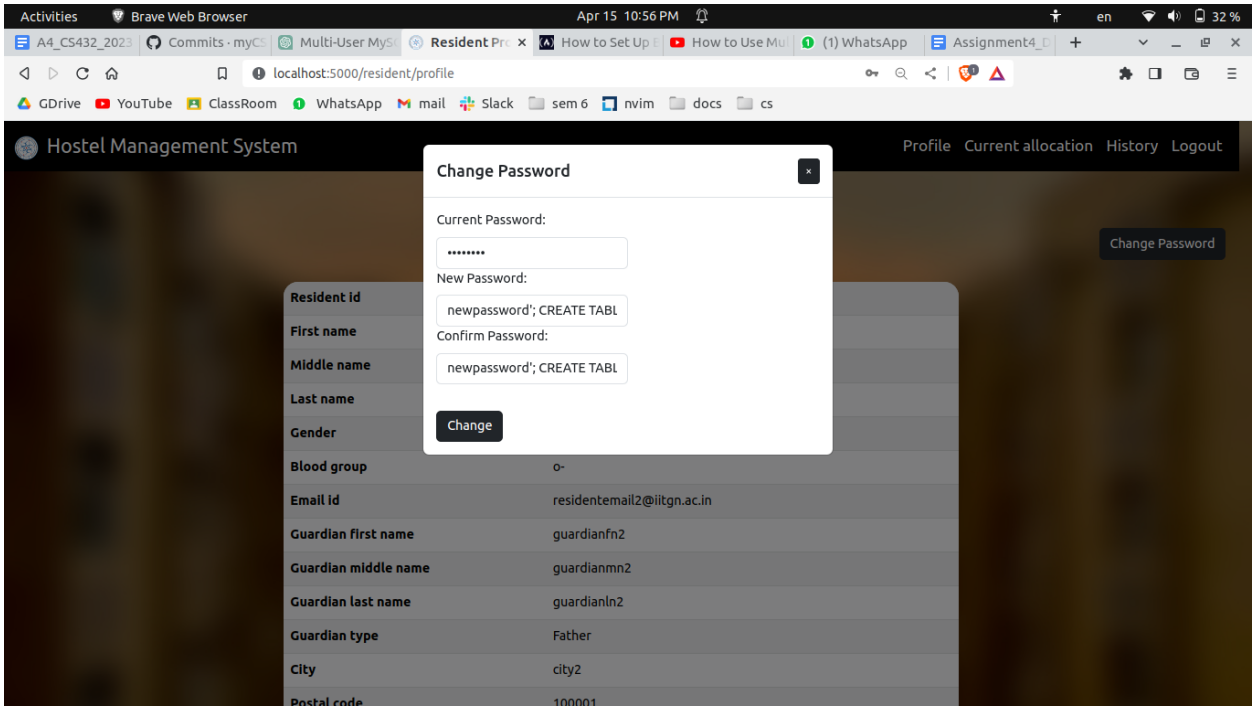




Logging in the admin using sql injection
Injection used: 00000001'; - -

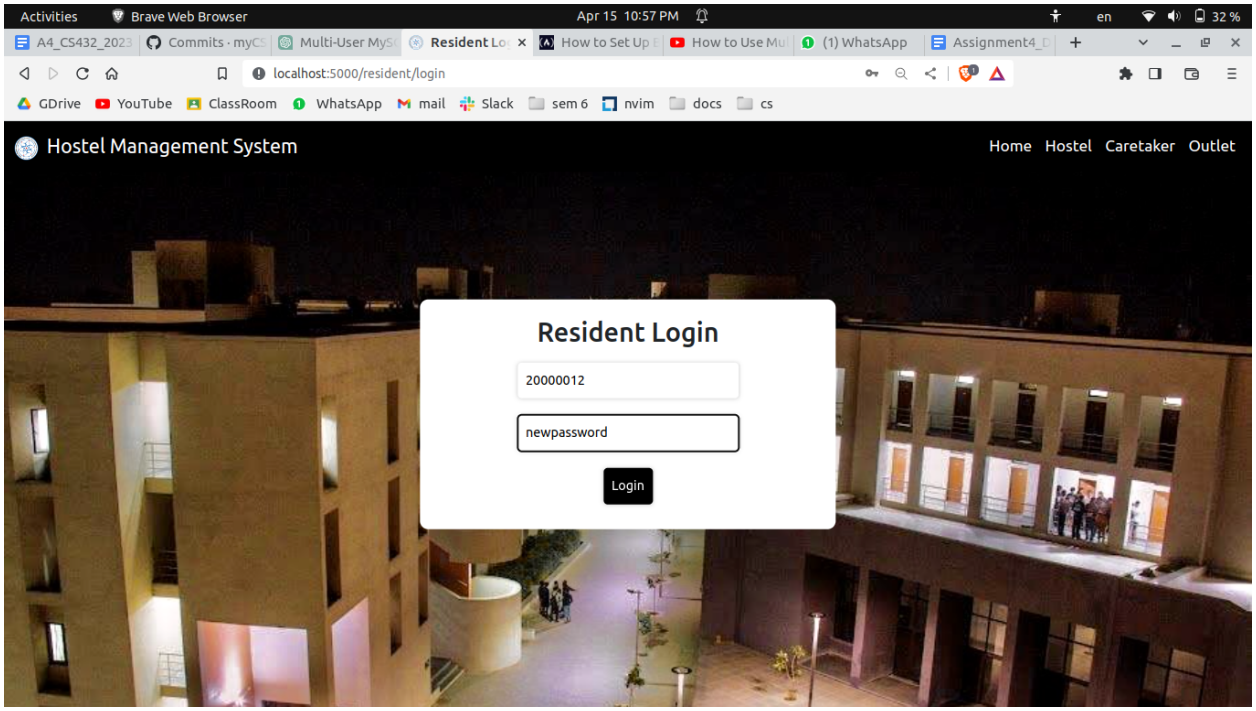


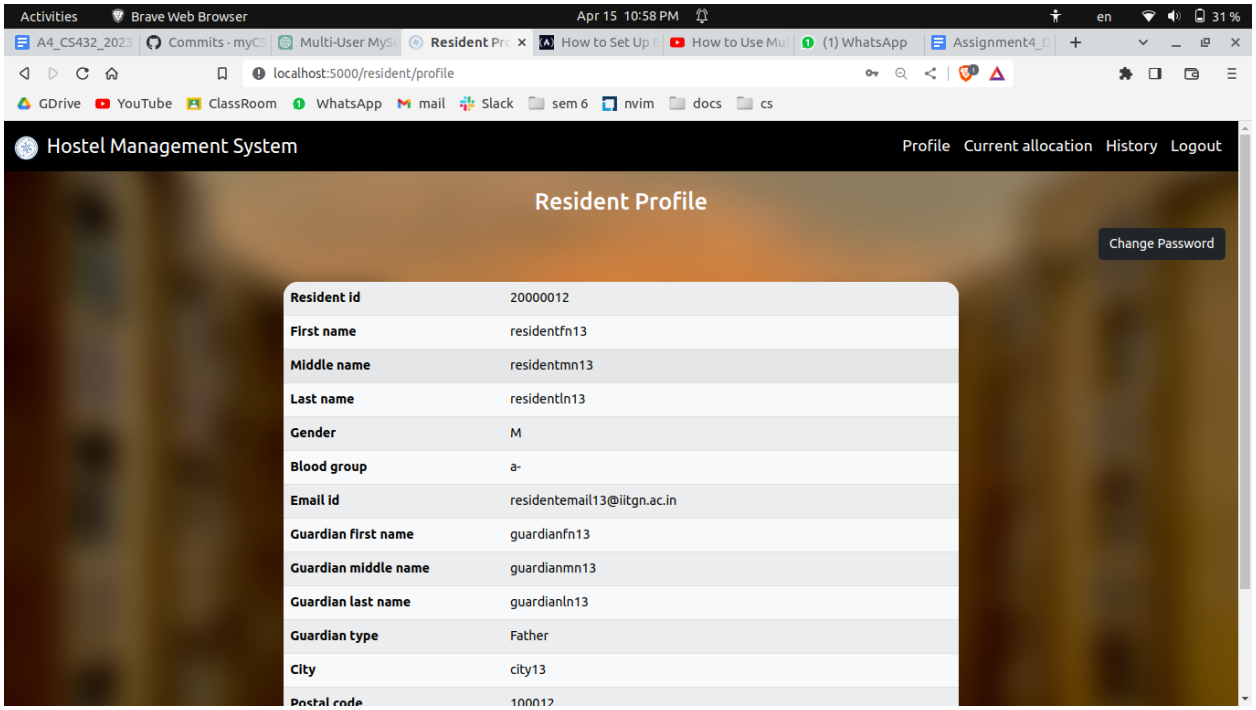
After logging in with correct credentials, you can change passwords of all users using the below sql injection script

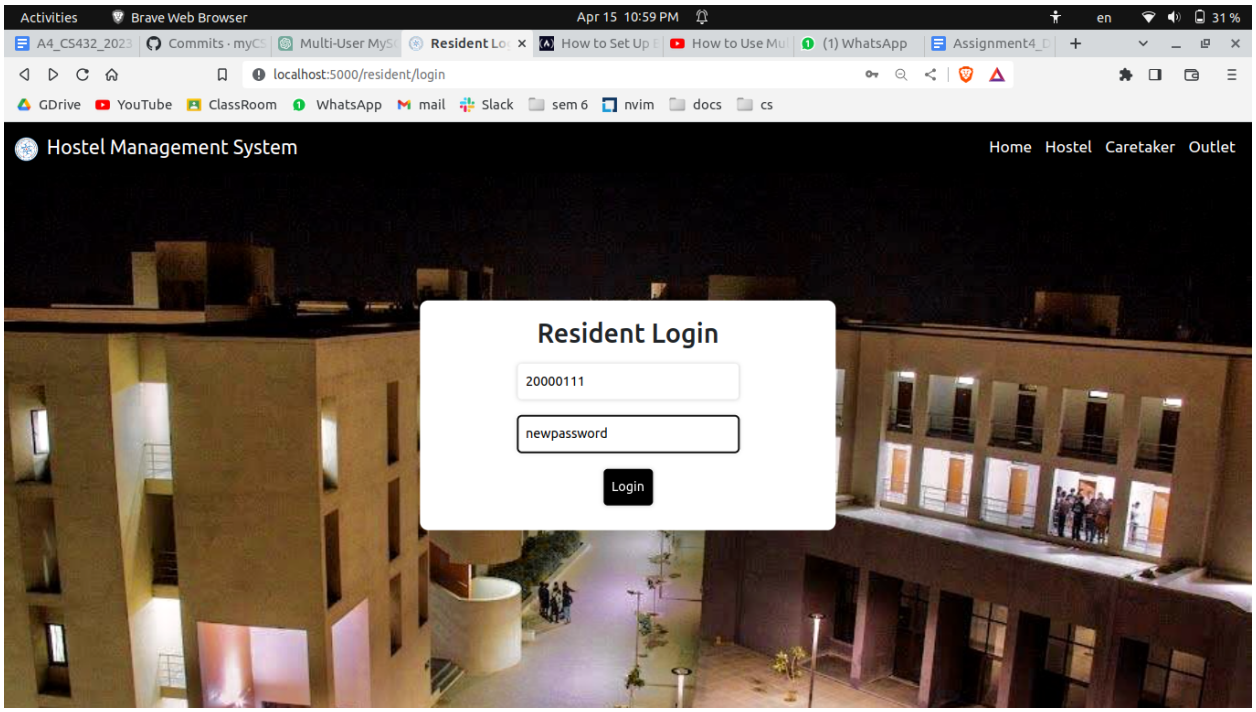


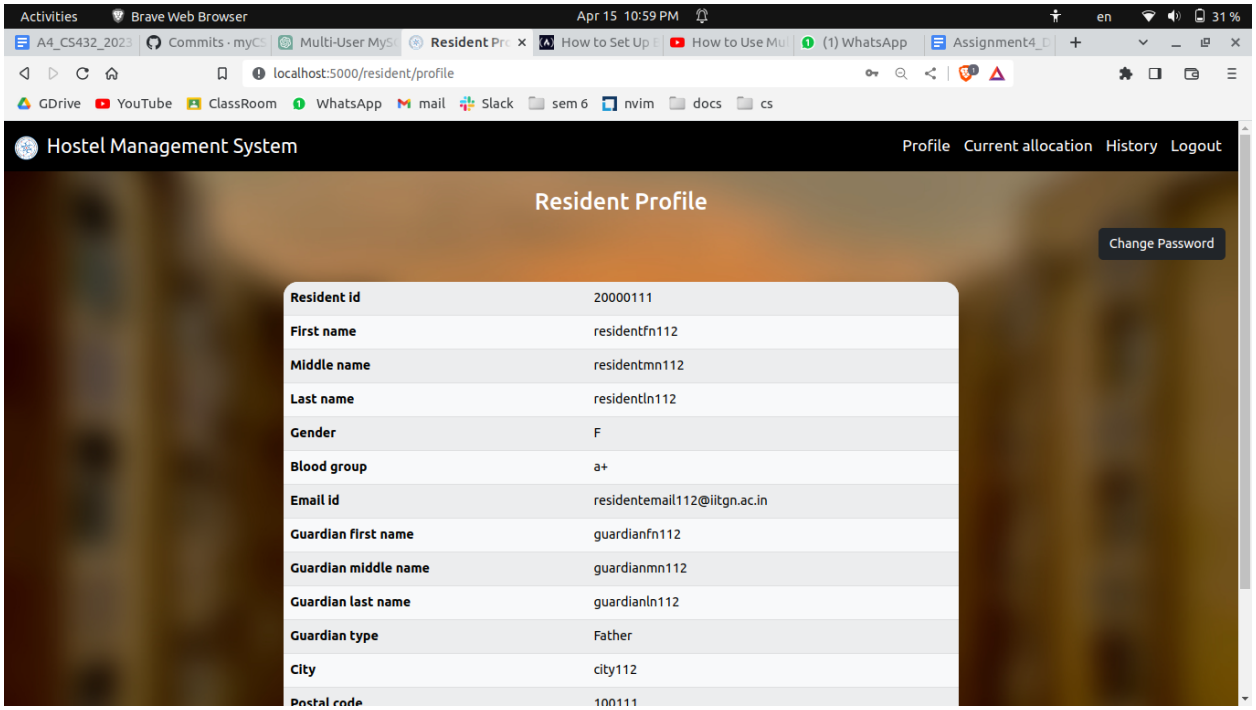
Injection used: newpassword'; CREATE TABLE buffer; SELECT * FROM users JOIN buffer ON '1' = '1

Changing passwords of all users




Activities
Brave Web Browser
Apr 15 10:58 PM
en
31 %
localhost:5000/resident/profile
GDrive
YouTube
ClassRoom
WhatsApp
mail
Slack
sem 6
nvim
docs
cs
Hostel Management System
Profile
Current allocation
History
Logout
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

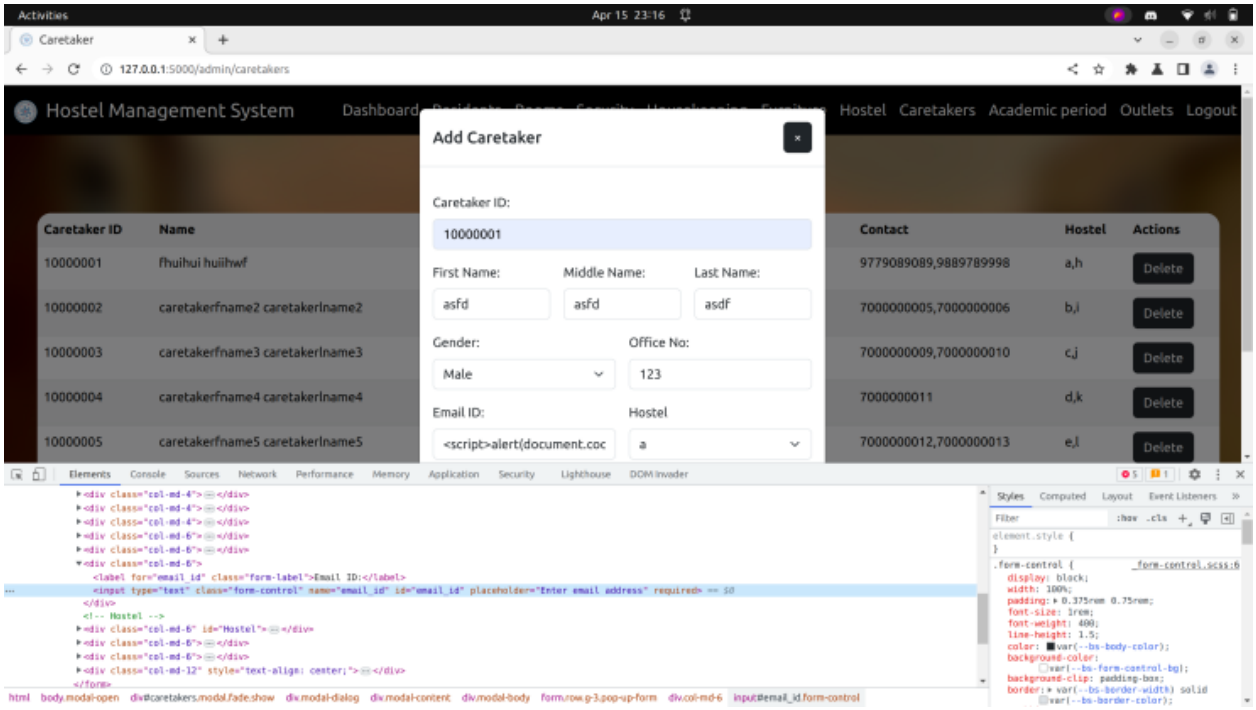

Activities
Brave Web Browser
Apr 15 10:59 PM
en
31 %
localhost:5000/resident/login
GDrive
YouTube
ClassRoom
WhatsApp
mail
Slack
sem 6
nvim
docs
cs
Hostel Management System
Home
Hostel
Caretaker
Outlet
Resident Login
20000111
newpassword
Login

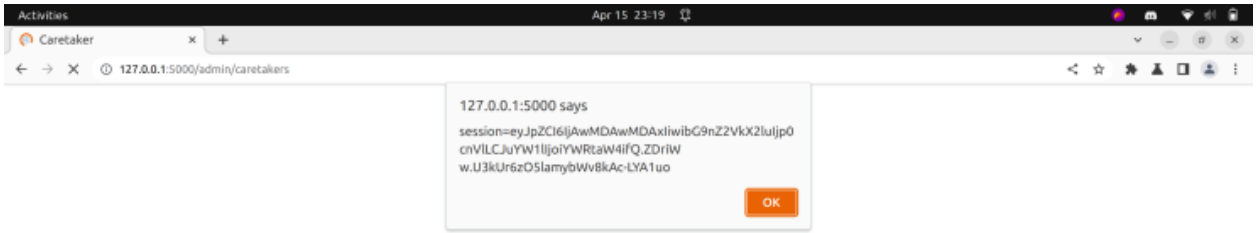


Defense:

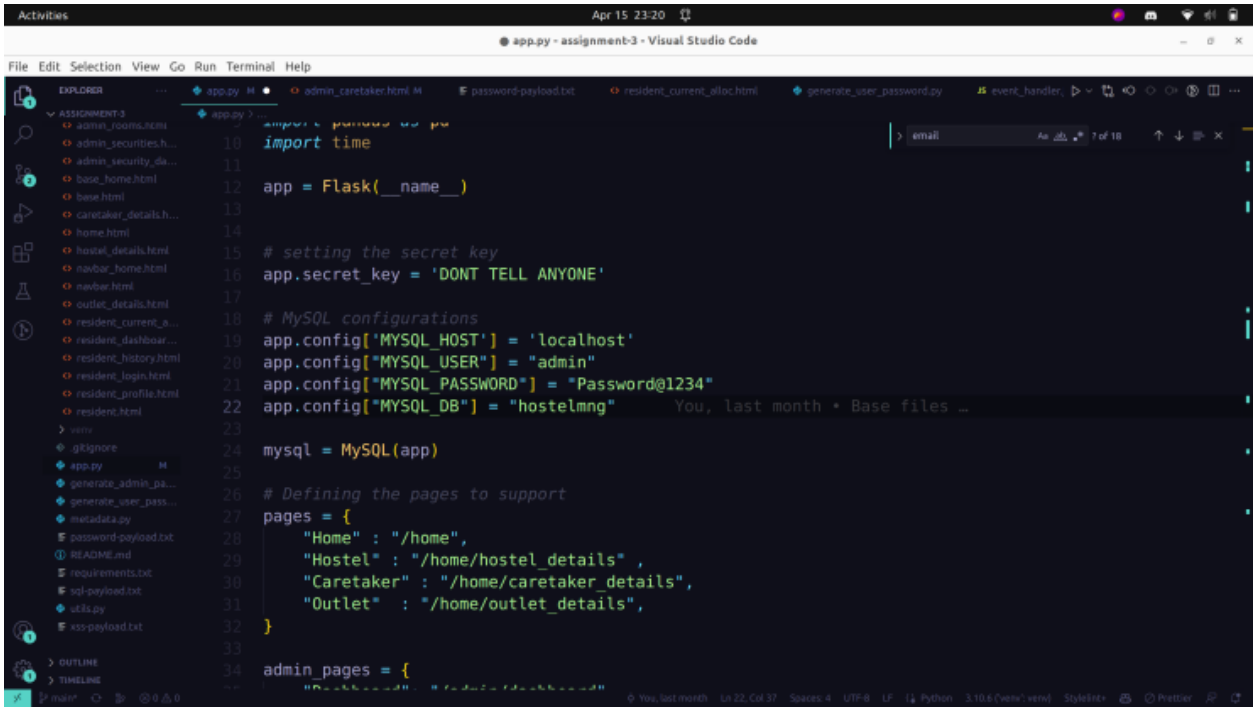
We used hash generator to generate hash values for the passwords and these generated hash passwords are stored in the database.

XSS Attack:

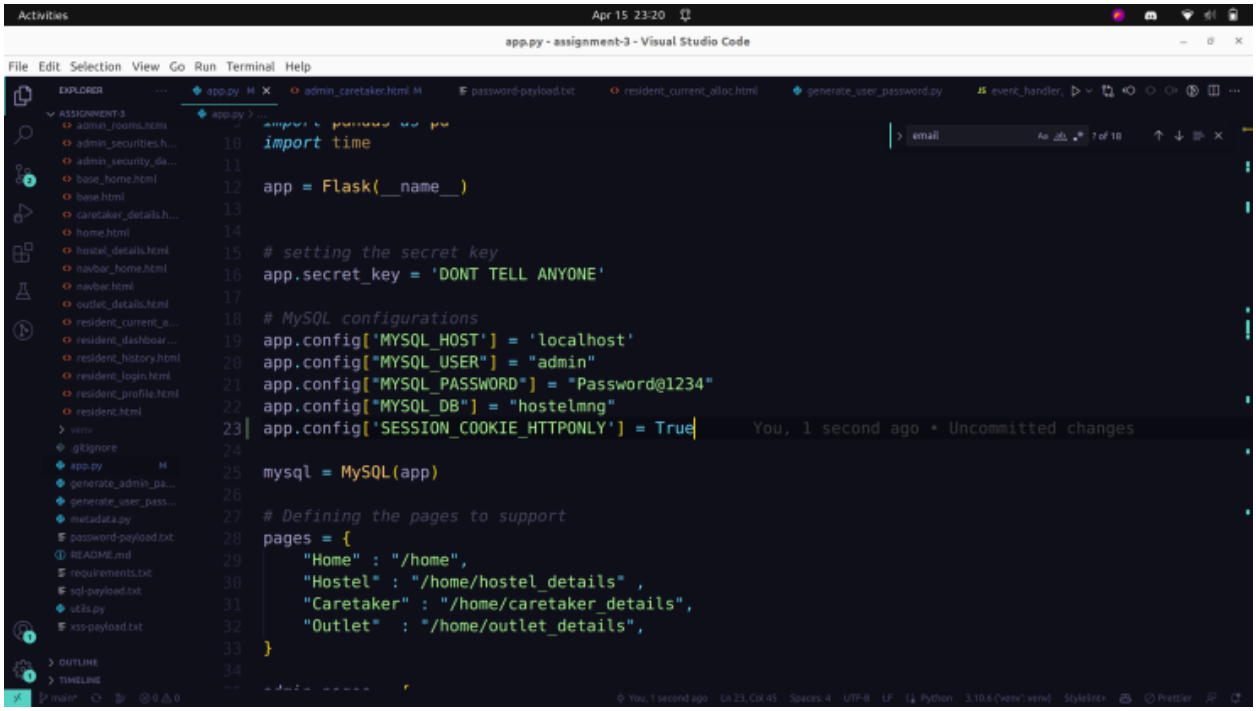

Activities
Apr 15 23:16
Caretaker
127.0.0.1:5000/admin/caretakers
Hostel Management System
Dashboard
Hostel
Caretakers
Academic period
Outlets
Logout
Add Caretaker
Caretaker ID:
10000001
First Name:
Middle Name:
Last Name:
asfd
asfd
asdf
Gender:
Office No:
Male
123
Email ID:
Hostel
<script>alert(document.coc
a
Caretaker ID
Name
Contact
Hostel
Actions
10000001
fhuihui huiihwf
9779089089,9889789998
a,h
Delete
10000002
caretakerfname2 caretakerlname2
7000000005,7000000006
b,i
Delete
10000003
caretakerfname3 caretakerlname3
7000000009,7000000010
c,j
Delete
10000004
caretakerfname4 caretakerlname4
7000000011
d,k
Delete
10000005
caretakerfname5 caretakerlname5
7000000012,7000000013
e,l
Delete
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
DOM Invader
Styles
Computed
Layout
Event Listeners


Activities
Apr 15 23:19
Caretaker
127.0.0.1:5000/admin/caretakers
127.0.0.1:5000 says
session=eyJpZCI6IjAwMDAwMDAxIiwibG9nZ2VkX2luIjp0
cnVlLCJuYW1lIjoiYWRtaW4ifQ.ZDriW
w.U3kUr6zO5lamybWv8kAc-LYA1uo
OK

Before



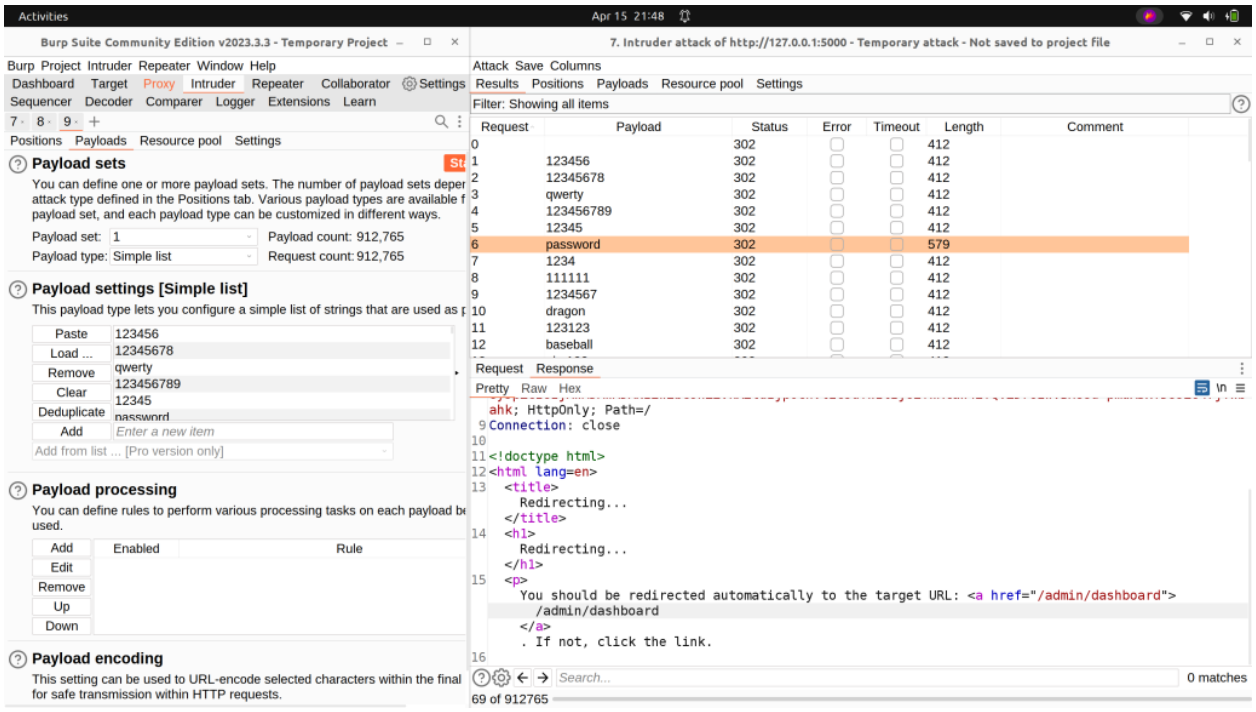
After



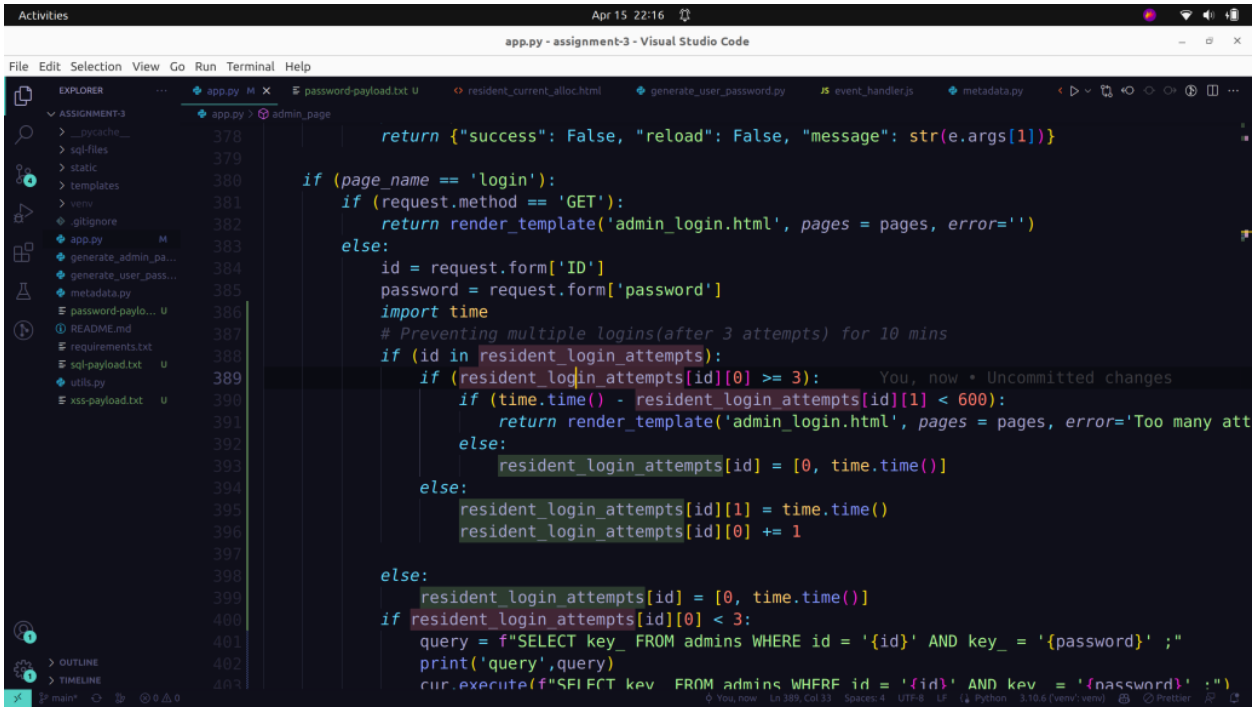
Dictionary attack:

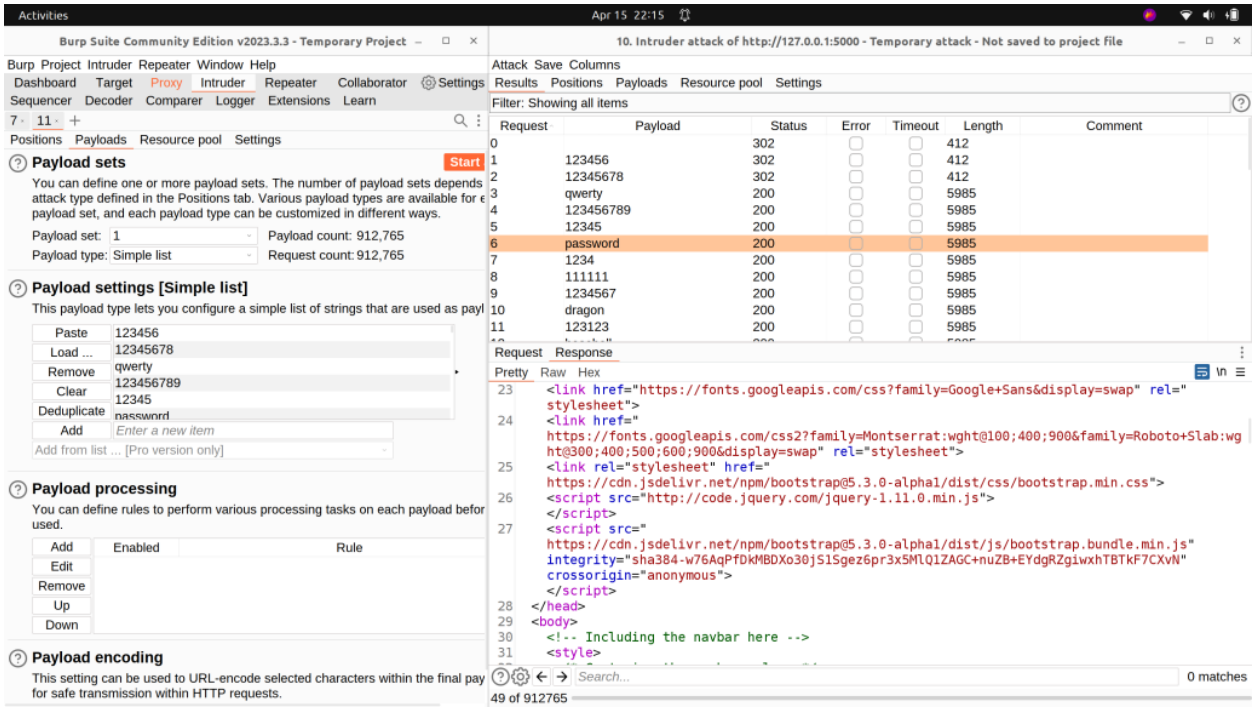
A. Attack : The unresolved code for the dictionary attack

```
Help
resident_current_alloc.html  generate_user_password.py  event_handler.js  metadata.py  admin_login.html  resident
admin_page

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'):
            # Fethcing all the hostel names
            query_string = "select distinct hostel_name from HOSTEL;"

You, 4 weeks ago  Ln 391, Col 29  Spaces: 4  UTF-8  LF  Python  3.10.6 ('venv': venv)
```



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





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