

**CS 432**  
**Databases**

**Assignment IV**

**DEPLOYING THE DBMS**

**QUERY CRAFTERS**

**Group Members**

Saurabh Kumar Sah	21110188
Shreya Patel	21110155
Riya Jain	21110178
Twinkle Devda	21110228
Ishika Raj	21110081
Saumya Jaiswal	21110186
Darsh Dalal	21110049
Het Trivedi	21110226

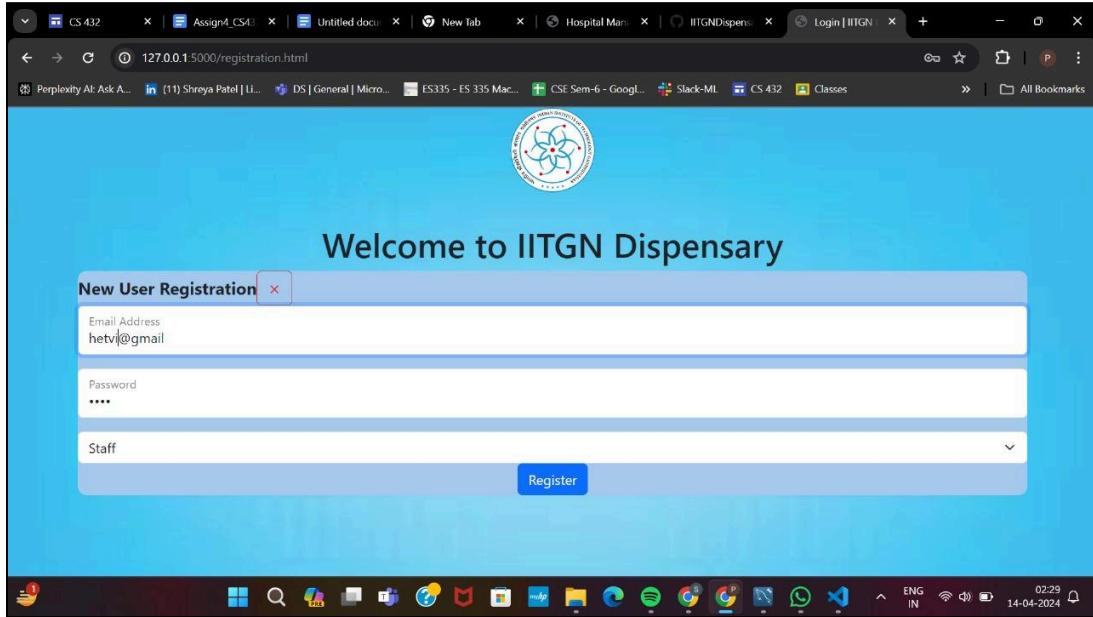
**April 16, 2024**

# Responsibility of G1

## Q1.

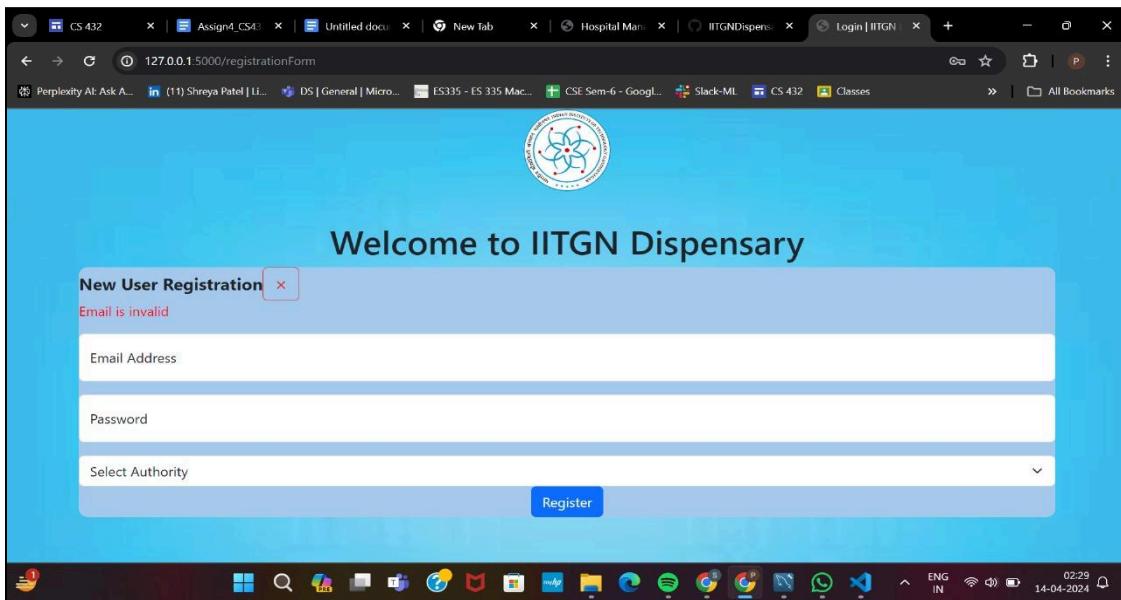
### 1. Changes in the Registration Form

#### a. Before first feedback



Anyone can register with an invalid email address and its entry also gets added to the authentication table. However, allowing registration with invalid email addresses can pose security risks and undermine the integrity of the authentication process.

#### b. After the first feedback



Following updates to the registration page, attempting to register with an invalid email address will prompt the message "**Email is invalid.**" This enhancement aims to improve user experience by providing immediate feedback guiding users towards providing valid email addresses.

## 2. Added the reason and remarks column

### a. Before Feedback

A screenshot of a web browser displaying a "Prescription" page. The URL is 127.0.0.1:5000/emergency. The page has two tabs: "Emergency" and "Prescription". The "Prescription" tab is active. Below the tabs is a table with columns: Actions, Prescription\_ID, Patient\_ID, Date, and Diagnosis. Each row contains an "Edit" button and a "Delete" button. The data in the table is as follows:

Actions	Prescription_ID	Patient_ID	Date	Diagnosis
Edit Delete	P0001	21060	11/1/2023	Fever
Edit Delete	P0002	23024	11/1/2023	Hepatitis
Edit Delete	P0003	19034	11/1/2023	Flu
Edit Delete	P0004	19027	11/1/2023	Chronic kidney disease
Edit Delete	P0005	19011	11/1/2023	Sprained ankle
Edit Delete	P0006	22013	11/1/2023	Sinus infection
Edit Delete	P0007	19055	11/1/2023	Pneumonia
Edit Delete	P0008	23012	11/2/2023	Osteoarthritis

Before the changes, users were not prompted to provide a reason for editing prescriptions, and doctors lacked the option to include additional remarks. By incorporating these features, the system now fosters better transparency, enabling users to provide context for edits and doctors to offer supplementary information, enhancing the overall quality of care.

### b. After Feedback

A screenshot of a web browser displaying a "Prescription" page. The URL is 127.0.0.1:5000/emergency. A modal window titled "127.0.0.1:5000 says" is open, asking "Please provide the reason for editing:" with an empty input field and "OK" and "Cancel" buttons. To the right of the modal is a table with columns: Actions, Prescription\_ID, Patient\_ID, Date, Diagnosis, Reason, and Remarks. The data in the table is as follows:

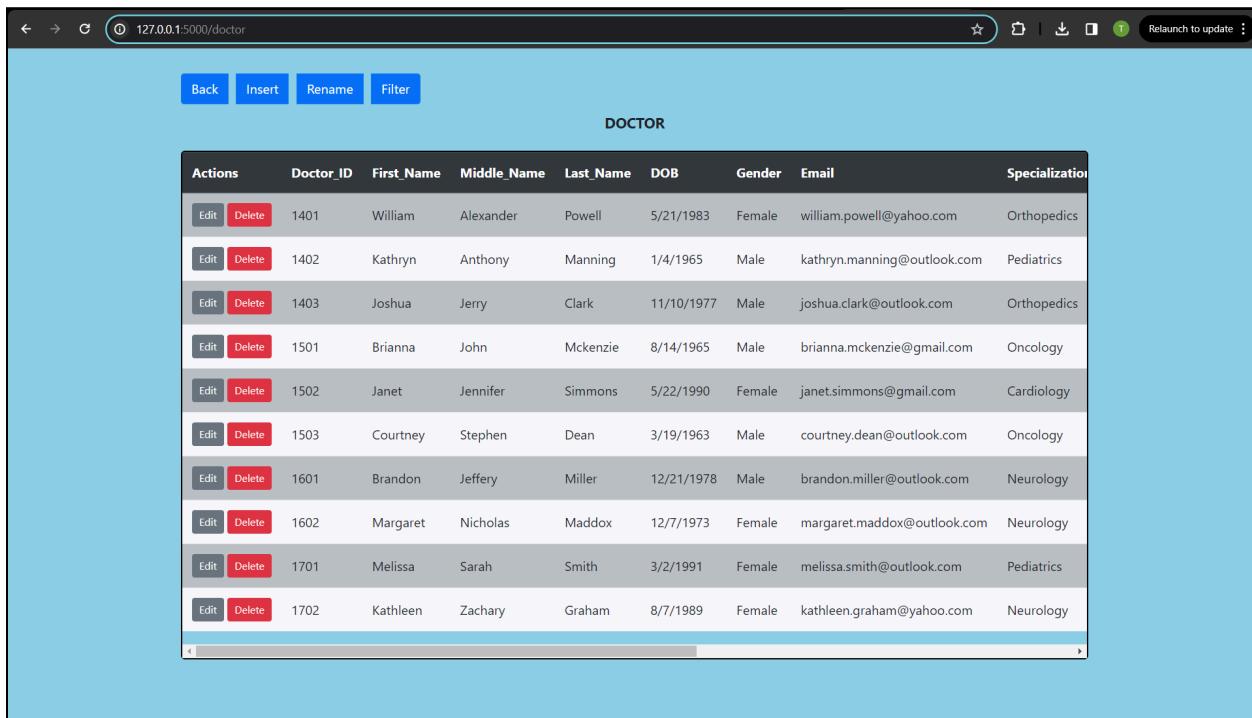
Actions	Prescription_ID	Patient_ID	Date	Diagnosis	Reason	Remarks
Edit Delete	P0001	21060	11/1/2023	Fever	Wrong Info	Drink water
Edit Delete	P0002	23024	11/1/2023	Hepatitis	null	null
Edit Delete	P0003	19034	11/1/2023	Flu	null	null
Edit Delete	P0004	19027	11/1/2023	Chronic kidney disease	null	null
Edit Delete	P0005	19011	11/1/2023	Sprained ankle	null	null
Edit Delete	P0006	22013	11/1/2023	Sinus infection	null	null
Edit Delete	P0007	19055	11/1/2023	Pneumonia	null	null
Edit Delete	P0008	23012	11/2/2023	Osteoarthritis	null	null

Here, upon clicking the edit button, a dialogue box prompts the user to input the reason for the modification. Once submitted, this reason is recorded alongside the edited prescription in the "Reason" column. An extra column titled "Remarks" also allows doctors to provide supplementary information or comments. These enhancements streamline the editing process, facilitate better communication, and ensure all relevant information is captured within the prescription record.

### 3. Staff and Doctor can see the OPD checked by the Doctor

#### a. Before Feedback

In the image below, we can see that there is no such option to track which patients a particular doctor has attended to in the OPD for both the current day and cumulative records.



The screenshot shows a web browser window with the URL 127.0.0.1:5000/doctor. The page title is "DOCTOR". At the top, there are four buttons: Back, Insert, Rename, and Filter. Below the title is a table with the following columns: Actions, Doctor\_ID, First\_Name, Middle\_Name, Last\_Name, DOB, Gender, Email, and Specialization. The table contains ten rows of data:

Actions	Doctor_ID	First_Name	Middle_Name	Last_Name	DOB	Gender	Email	Specialization
Edit Delete	1401	William	Alexander	Powell	5/21/1983	Female	william.powell@yahoo.com	Orthopedics
Edit Delete	1402	Kathryn	Anthony	Manning	1/4/1965	Male	kathryn.manning@outlook.com	Pediatrics
Edit Delete	1403	Joshua	Jerry	Clark	11/10/1977	Male	joshua.clark@outlook.com	Orthopedics
Edit Delete	1501	Brianna	John	Mckenzie	8/14/1965	Male	brianna.mckenzie@gmail.com	Oncology
Edit Delete	1502	Janet	Jennifer	Simmons	5/22/1990	Female	janet.simmons@gmail.com	Cardiology
Edit Delete	1503	Courtney	Stephen	Dean	3/19/1963	Male	courtney.dean@outlook.com	Oncology
Edit Delete	1601	Brandon	Jeffery	Miller	12/21/1978	Male	brandon.miller@outlook.com	Neurology
Edit Delete	1602	Margaret	Nicholas	Maddox	12/7/1973	Female	margaret.maddox@outlook.com	Neurology
Edit Delete	1701	Melissa	Sarah	Smith	3/2/1991	Female	melissa.smith@outlook.com	Pediatrics
Edit Delete	1702	Kathleen	Zachary	Graham	8/7/1989	Female	kathleen.graham@yahoo.com	Neurology

#### b. After Feedback

We've introduced the "**OPD**" option, triggering a dialogue box when clicked. Within this box, users can select whether they wish to view the total patients checked by the doctor either for the current day or cumulatively till the present. Users can access the OPD entries corresponding to the selected doctor upon submission. If there are no entries for that doctor in the OPD, the system will display a message stating "**No OPD data found,**" ensuring clarity and transparency.

in information retrieval.

The screenshot shows a web application titled "DOCTOR" with a table listing seven doctors. Each row contains columns for Actions (Edit, Delete, OPD), Apartment Number, City, DOB, Doctor ID, and Email. The "OPD" button in the first row is highlighted in blue, indicating it is the current action being performed.

Actions	Apartment_Number	City	DOB	Doctor_ID	Email
Edit Delete OPD		Valsad	Sat, 09 Feb 2002 00:00:00 GMT	156	riyaj@
Edit Delete OPD		New York	Fri, 09 Sep 1994 00:00:00 GMT	1401	scvjd@
Edit Delete OPD	954	Hopkinsstad	Mon, 04 Jan 1965 00:00:00 GMT	1402	kathy@
Edit Delete OPD	84898	Port Raymond	Thu, 10 Nov 1977 00:00:00 GMT	1403	joshua@
Edit Delete OPD	4835	Port Spencer	Sat, 14 Aug 1965 00:00:00 GMT	1501	brian@
Edit Delete OPD	5	Brownmouth	Tue, 22 May 1990 00:00:00 GMT	1502	janet@
Edit Delete OPD	9475	Joseburgh	Tue, 19 Mar 1963 00:00:00 GMT	1503	courtney@

A modal window titled "OPD Details" is displayed over the main table. It contains two radio buttons: "Today's OPD" (unchecked) and "Total OPD" (checked). Below the radio buttons is a text input field labeled "Doctor ID:" with the value "156". A large blue button labeled "View OPD" is centered below the input field. In the background, the main table is partially visible, showing rows for Doctors 1401, 1402, 1403, and 1501. A small portion of the table for Doctor 156 is also visible at the bottom right of the modal.

OPD Details

Actions Apartment

Doctor\_ID Email

156 riyaj@...  
1401 scvjd...  
1402 kathy...  
1403 joshua...  
1501 briann...  
janet.s...  
court...  
...  
...

4. Remove view access of some tables to the patient

a. Before Feedback

Update Details

Role

OPD

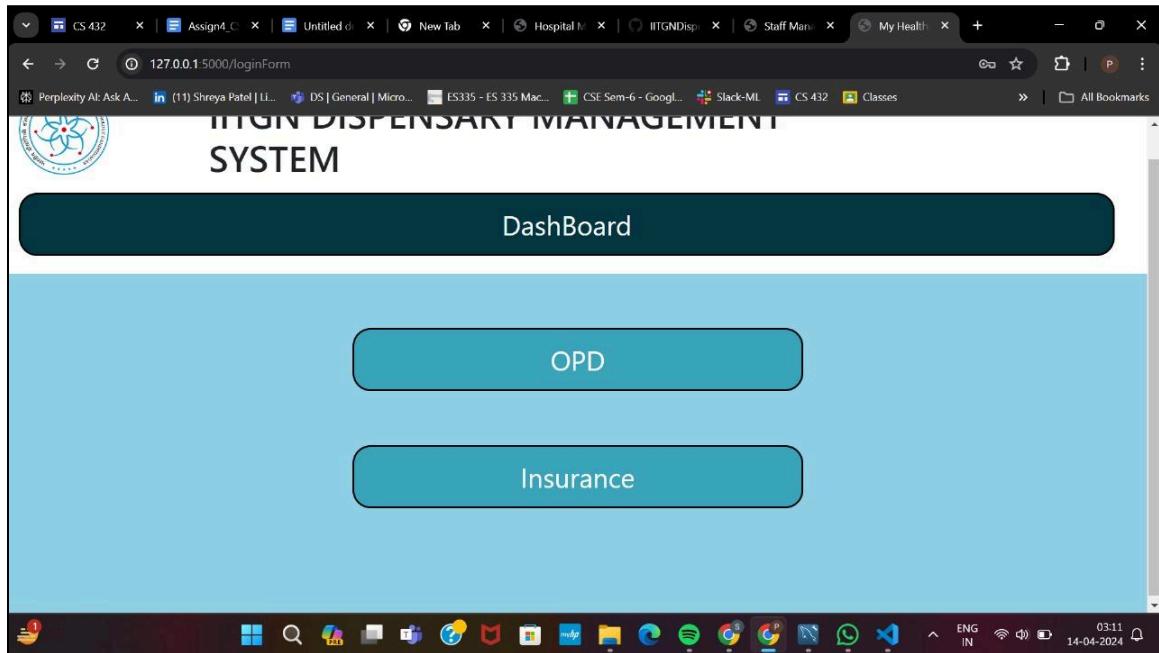
Emergency/Prescription

Medicines & Eq/Purchase Order

Insurance

Initially, patients had view access to all tables, but following discussions with stakeholders, it was determined that patients should only have access to specific tables: OPD, Insurance, and Patient. In these tables, patients can only view entries corresponding to their data, ensuring privacy and confidentiality.

b. After Feedback



The screenshot shows a web browser window for the 'ITGN DISPENSARY MANAGEMENT SYSTEM'. The title bar displays the URL '127.0.0.1:5000/insurance\_p?email=oali@example.com'. The main content area shows a table titled 'Insurance' with the following data:

Insurance_Number	Issue_Date	Expiry_Date	Wallet_Balance	Reimbursement_Status
I0002	2022/05/01	2025/07/15	1800.00	Rejected

The bottom of the screen shows a Windows taskbar with various icons and the date '14-04-2024'.

The screenshot shows a web browser window with a URL of 127.0.0.1:5000/opd\_p?email=oali@example.com. The page title is "Patient". There are two buttons at the top: "Back" and "OPD Patient". Below the title is a table with the following columns: Patient\_ID, First\_Name, Middle\_Name, Last\_Name, DOB, Street\_Number, Street\_Name, Apartment\_Number, and City. A single row of data is displayed: Patient\_ID 19002, First\_Name Brandon, Middle\_Name Maria, Last\_Name Lester, DOB 11/13/2000, Street\_Number 555, Street\_Name Fuller Parkway, Apartment\_Number 15, and City Michaelmout. The browser's address bar shows the URL, and the taskbar at the bottom includes icons for various applications like File Explorer, Edge, and Spotify.

Patients are limited to exclusively viewing the Patient, OPD, and Insurance tables. Moreover, patients can access their entries within these tables, ensuring privacy and confidentiality. This restriction ensures that patients can only access information pertinent to their own healthcare records, aligning with privacy regulations.

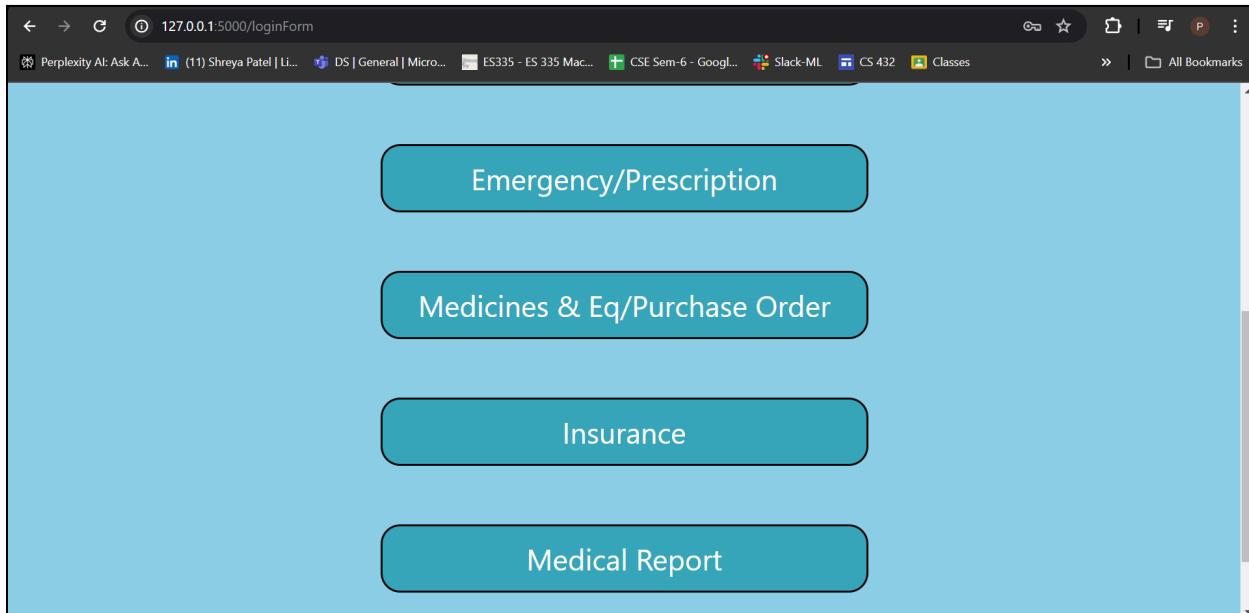
##### 5. Added new table to show the entries for the medical test report

###### a. Before Feedback

The screenshot shows a user interface titled "Update Details". At the top left is a circular logo with a stylized design. Below the title are five teal-colored buttons arranged vertically: "Role", "OPD", "Emergency/Prescription", "Medicines & Eq/Purchase Order", and "Insurance". The background is a light blue color.

As you can see that initially there is no option to see the information regarding the medical report.

b. After Feedback

A screenshot of a web browser window with a light blue background. The title bar says "127.0.0.1:5000/medical\_report". Below the title bar are browser tabs and icons. The main content area has a header "Medical Report". Underneath is a table with a dark grey header row containing columns: "Action", "Prescription\_ID", "Patient\_ID", "Date", and "Diagnosis". There are seven rows of data, each with an "Edit" button in the "Action" column. The data is as follows:

Action	Prescription_ID	Patient_ID	Date	Diagnosis
Edit	P0001	21060	Wed, 01 Nov 2023 00:00:00 GMT	Fever
Edit	P0002	23024	Wed, 01 Nov 2023 00:00:00 GMT	Hepatitis
Edit	P0003	19034	Wed, 01 Nov 2023 00:00:00 GMT	Flu
Edit	P0004	19027	Wed, 01 Nov 2023 00:00:00 GMT	Chronic kidney disease
Edit	P0004	19027	Wed, 01 Nov 2023 00:00:00 GMT	Chronic kidney disease
Edit	P0005	19011	Wed, 01 Nov 2023 00:00:00 GMT	Sprained ankle
Edit	P0006	22013	Wed, 01 Nov 2023 00:00:00 GMT	Sinus infection

The Medical Report table contains essential information such as patient ID, prescription ID, date of test, diagnosis, test name, and result. Additionally, there is an option to edit entries, facilitating

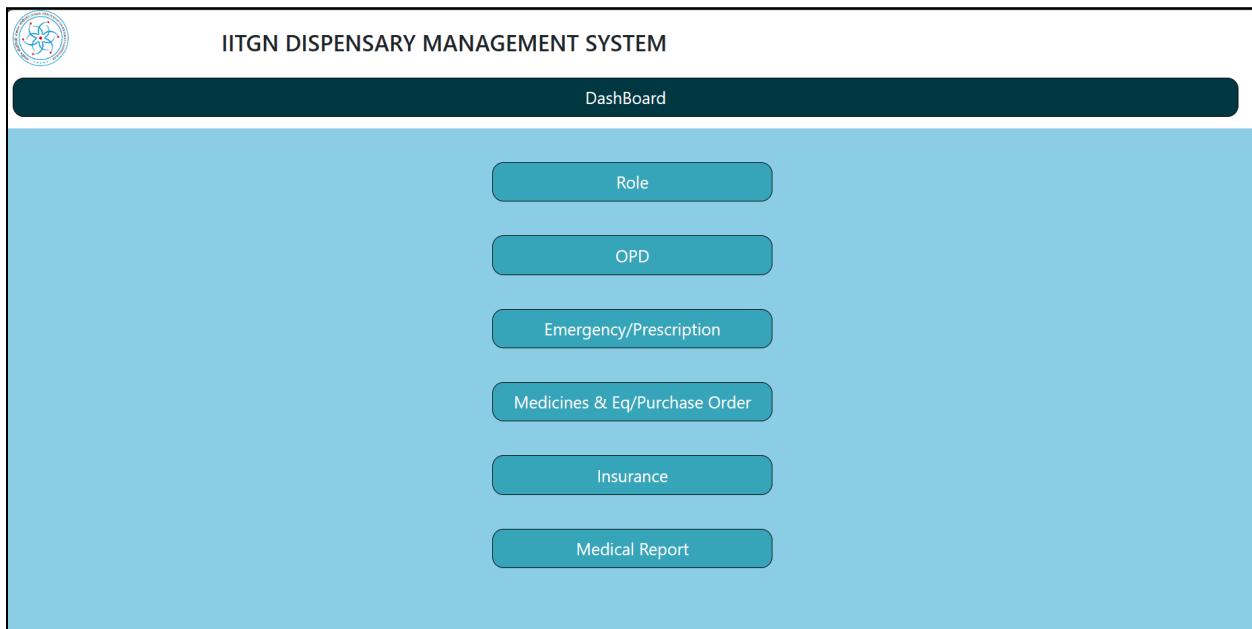
adjustments by staff or doctors. This edit functionality is particularly useful for updating test results from "Pending" to "Positive" or "Negative" based on the outcome, ensuring accurate and up-to-date medical records.

Q2.

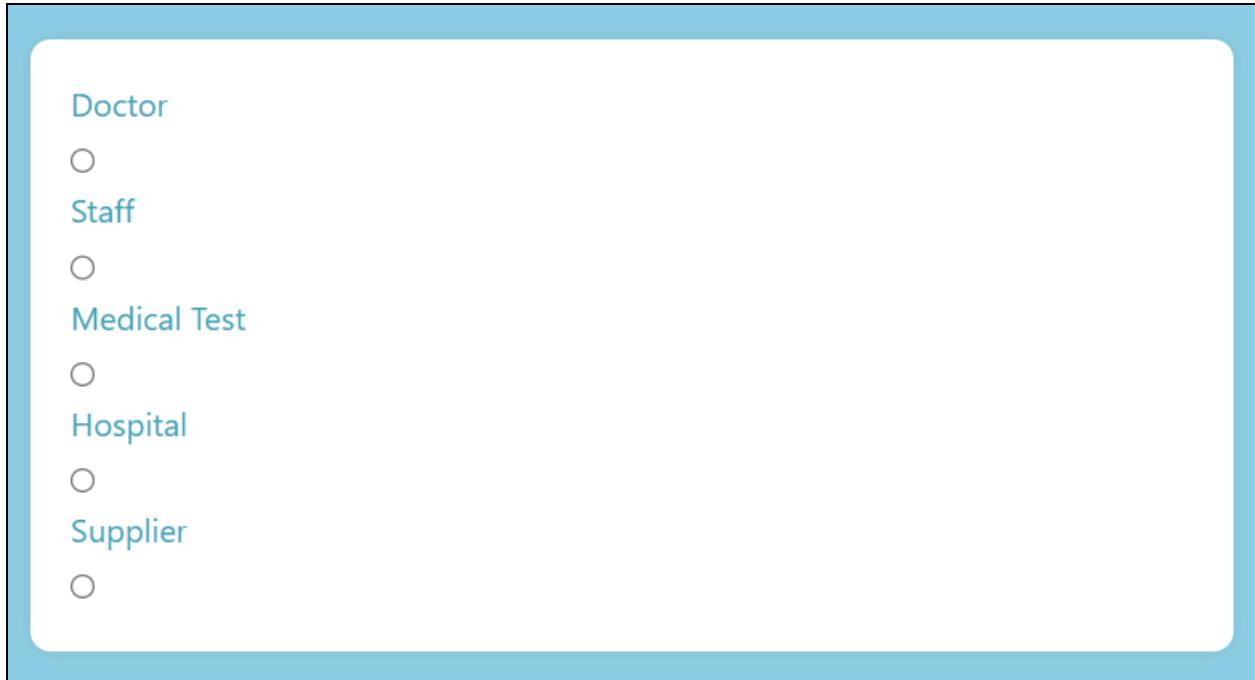
We are dealing total 3 types of the users :

- 1) Staff: Responsible for administrative and operational tasks within the healthcare facility. They might handle patient registrations, appointments, and general administrative duties.
- 2) Doctor: Professionals who diagnose, treat, and provide medical care to patients. They may prescribe medications, order tests, and oversee treatment plans.
- 3) Patient: Individuals seeking medical treatment within the healthcare facility. Patients access services, view their medical records, and participate in their treatment plans.

### 1) Staff



When a staff member logs in, they'll see the page depicted above. They have the freedom to select any of the provided options and perform actions such as viewing, editing, deleting, and inserting data into the corresponding tables within the database. For instance, if they click on the "Role" option, a page will appear with choices like "Doctor," "Staff," "Medical Test," "Hospital," and "Supplier." By selecting any of these tables, they can then view the data contained within it, as illustrated in the image below.



DOCTOR						
Actions	Apartment_Number	City	DOB	Doctor_ID	Email	
Edit Delete OPD	678	West Sandy	Sat, 21 May 1983 00:00:00 GMT	1401	william.powell@yahoo.com	
Edit Delete OPD	954	Hopkinsstad	Mon, 04 Jan 1965 00:00:00 GMT	1402	kathryn.manning@outlook.com	
Edit Delete OPD	84898	Port Raymond	Thu, 10 Nov 1977 00:00:00 GMT	1403	joshua.clark@outlook.com	
Edit Delete OPD	4835	Port Spencer	Sat, 14 Aug 1965 00:00:00 GMT	1501	brianna.mckenzie@gmail.com	
Edit Delete OPD	5	Brownmouth	Tue, 22 May 1990 00:00:00 GMT	1502	janet.simmons@gmail.com	
Edit Delete OPD	9475	Joseburgh	Tue, 19 Mar 1963 00:00:00 GMT	1503	courtney.dean@outlook.com	
Edit Delete OPD	320	East Ashleymouth	Thu, 21 Dec 1978 00:00:00 GMT	1601	brandon.miller@outlook.com	
Edit Delete OPD	10523	Mannfort	Fri, 07 Dec 1973 00:00:00 GMT	1602	margaret.maddox@outlook.com	
Edit Delete OPD	361	Arnoldview	Sat, 02 Mar 1991 00:00:00 GMT	1701	melissa.smith@outlook.com	
Edit Delete OPD	14256	Port Patrick	Mon, 07 Aug 1989 00:00:00 GMT	1702	kathleen.graham@yahoo.com	

Likewise, if the user selects the "OPD" option, they will have access to view and perform various operations on the OPD table. Additionally, they can also choose to perform operations on the patient table. Below is an example interface for the OPD table, where options for editing and deleting entries are visible.

OPD						
Actions	Serial_Number	Date	Time	Patient_ID	Doctor_ID	Case_Type
Edit Delete OPD	1	2024/03/31	1:30:24	21060	1602	New
Edit Delete OPD	1	2023/12/11	17:53:27	20058	1401	Old
Edit Delete OPD	2	2023/12/11	18:26:03	21009	1601	New
Edit Delete OPD	3	2023/12/11	19:18:30	21031	1601	New
Edit Delete OPD	4	2023/12/11	18:23:46	20024	1503	Old
Edit Delete OPD	5	2023/12/11	16:42:19	22014	1401	New
Edit Delete OPD	6	2023/12/11	19:20:17	22055	1403	Old
Edit Delete OPD	7	2023/12/11	16:24:45	19009	1602	New
Edit Delete OPD	8	2023/12/11	16:11:52	23001	1502	New
Edit Delete OPD	1	2023/12/10	15:51:04	22003	1502	Old

**Privilege :** staff members are granted full access privileges across all tables present in the database. This includes the ability to view, edit, and delete entries within any table.

## 2) Doctor

If a user is logged in as a doctor, they will have access to all the tables just like staff members. However, doctors will be restricted from editing or deleting entries in certain tables. For example, as shown in the image below, a doctor can edit and delete entries from the doctor table, but such options are not available for the staff table. This restriction ensures that doctors can manage their own data effectively while maintaining the integrity and security of other tables within the database.

DOCTOR										
Actions	Apartment_Number	City	DOB	Doctor_ID	Email	Experience	First_Name	Gender	Last_Name	Middle_Name
Edit Delete OPD	678	West Sandy	Sat, 21 May 1983 00:00:00 GMT	1401	william.powell@yahoo.com	17	William	Female	Powell	Alexander
Edit Delete OPD	954	Hopkinsstad	Mon, 04 Jan 1965 00:00:00 GMT	1402	kathryn.manning@outlook.com	31	Kathryn	Male	Manning	Anthony
Edit Delete OPD	84898	Port Raymond	Thu, 10 Nov 1977 00:00:00 GMT	1403	joshua.clark@outlook.com	38	Joshua	Male	Clark	Jerry
Edit Delete OPD	4835	Port Spencer	Sat, 14 Aug 1965 00:00:00 GMT	1501	brianna.mckenzie@gmail.com	14	Brianna	Male	Mckenzie	John
Edit Delete OPD	5	Brownmouth	Tue, 22 May 1990 00:00:00 GMT	1502	janet.simmons@gmail.com	36	Janet	Female	Simmons	Jennifer
Edit Delete OPD	9475	Joseburgh	Tue, 19 Mar 1963 00:00:00 GMT	1503	courtney.dean@outlook.com	29	Courtney	Male	Dean	Stephen
Edit Delete OPD	320	East Ashleymouth	Thu, 21 Dec 1978 00:00:00 GMT	1601	brandon.miller@outlook.com	18	Brandon	Male	Miller	Jeffery
Edit Delete OPD	10523	Mannfort	Fri, 07 Dec 1973 00:00:00 GMT	1602	margaret.maddox@outlook.com	1	Margaret	Female	Maddox	Nicholas
Edit Delete OPD	361	Arnoldview	Sat, 02 Mar 1991 00:00:00 GMT	1701	melissa.smith@outlook.com	38	Melissa	Female	Smith	Sarah
Edit Delete OPD	14256	Port Patrick	Mon, 07 Aug 1989 00:00:00 GMT	1702	kathleen.graham@yahoo.com	1	Kathleen	Female	Graham	Zachary

Staff							
Staff_ID	First_Name	Middle_Name	Last_Name	Join_Date	DOB	Street_Number	City
1	we	are	family	Thu, 18 Apr 2024 00:00:00 GMT	Wed, 17 Apr 2024 00:00:00 GMT	1	London
2001	Alexandra	Ross	Arroyo	Thu, 25 Feb 2010 00:00:00 GMT	Thu, 09 Nov 1989 00:00:00 GMT	312	Baltimore
2002	Candace	Angela	Davidson	Wed, 03 Feb 2021 00:00:00 GMT	Thu, 01 Apr 1999 00:00:00 GMT	765	Seattle
2003	Melissa	Tina	Taylor	Thu, 02 Nov 2023 00:00:00 GMT	Mon, 14 Mar 1977 00:00:00 GMT	622	Los Angeles
2004	Caleb	Sandra	Johnson	Mon, 22 Jun 2009 00:00:00 GMT	Mon, 16 Oct 2000 00:00:00 GMT	63	Chicago
2005	Kayla	Matthew	Good	Mon, 12 May 2014 00:00:00 GMT	Tue, 18 Sep 2001 00:00:00 GMT	378	Houston
2006	Aaron	Ernest	Alexander	Sat, 30 Dec 2023 00:00:00 GMT	Tue, 24 Jun 1986 00:00:00 GMT	825	Phoenix
2007	Nicholas	Michael	Thomas	Sun, 24 Jul 2022 00:00:00 GMT	Sun, 09 Apr 1995 00:00:00 GMT	942	San Antonio
2008	Stephanie	Jeremy	Moore	Fri, 15 Oct 2021 00:00:00 GMT	Fri, 21 Aug 1992 00:00:00 GMT	3	Dallas
2009	Daniel	Sue	Collier	Tue, 26 Jun 2012 00:00:00 GMT	Thu, 22 Dec 1988 00:00:00 GMT	311	Austin
2010	Shaun	Nicole	Watson	Fri, 29 Jul 2011 00:00:00 GMT	Sat, 30 Dec 1933 00:00:00 GMT	383	San Diego
2011	Christopher	Scott	Clark	Tue, 10 Dec 2023 00:00:00 GMT	Tue, 14 Feb 1980 00:00:00 GMT	620	San Jose

**Privilege:** Doctors can view all table data in the databases through different pages. They are granted the following access permissions:

- Edit and Delete Access:
  - Doctor table
  - Prescription table
  - Emergency table
- Edit-Only Access:
  - Medical Report table

### 3) Patient

If the User is logged in as a patient then the website view for the patient is as per shown in the change number 4.

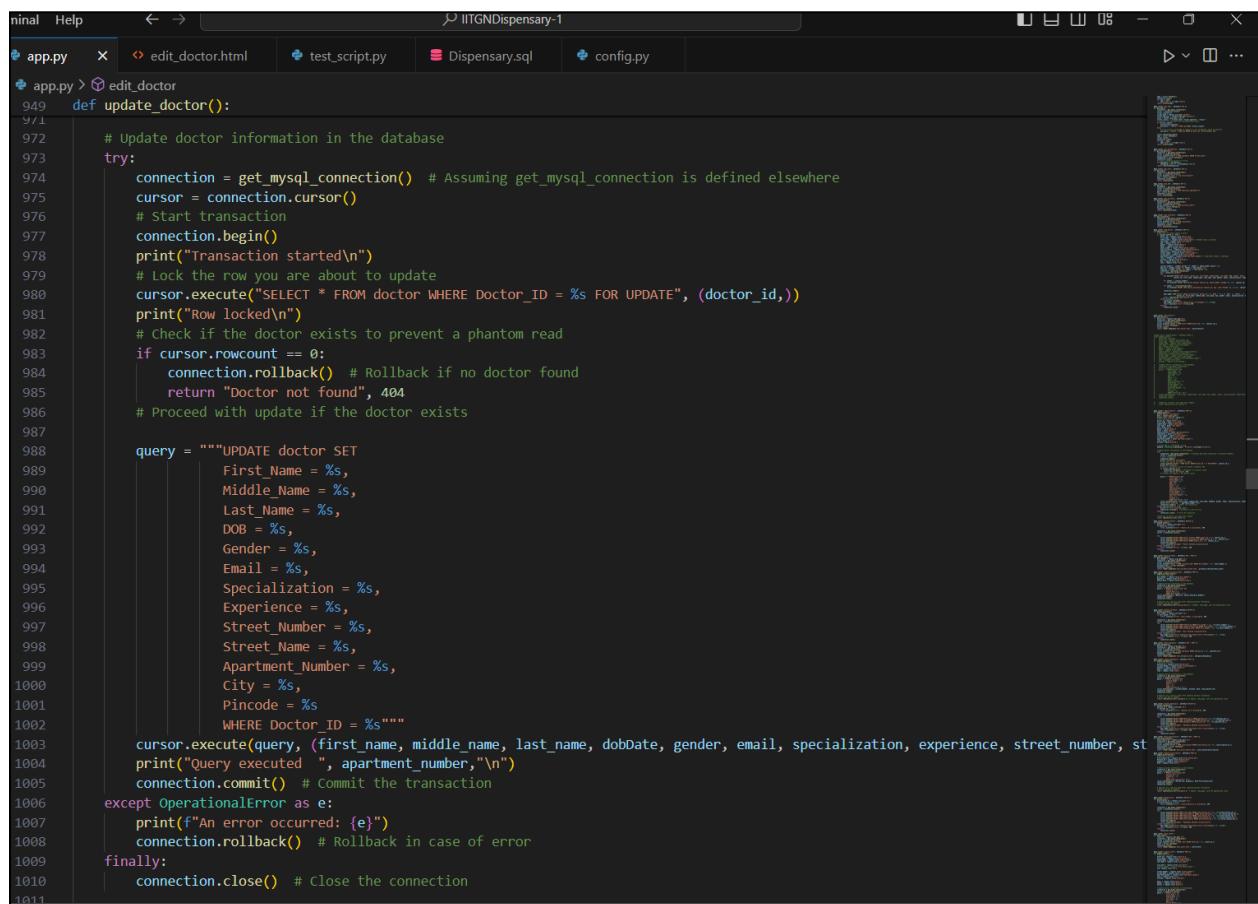
**Privilege:** patients will have restricted access to the following tables: OPD, Patient, Emergency, and Insurance. Specifically, patients will only be able to view their own entries in these tables and will not have access to view entries belonging to other patients. Additionally, patients will not have the ability to edit or delete any entries in these tables.

## Responsibility of G2

### Q1. Implementing Concurrent multi-user access:

Added Row-Level locks in the “update\_doctor” function to control access to individual rows of a table by different transactions. If multiple users try to edit a single doctor entry at the same time, sequential edit requests will be executed. Randomly, one request is chosen, the specific doctor row gets locked till the first request completely gets executed, then next requests in queue get executed. It prevents the data inconsistencies that can occur when multiple transactions attempt to modify the same data simultaneously.

- When a transaction selects a row for updating, the database system places a lock on that specific row.



The screenshot shows a terminal window titled "IITGNDispensary-1". The window contains several tabs: "app.py", "edit\_doctor.html", "test\_script.py", "Dispensary.sql", and "config.py". The "app.py" tab is active and displays the following Python code:

```
1 app.py > edit_doctor
2 def update_doctor():
3     # Update doctor information in the database
4     try:
5         connection = get_mysql_connection() # Assuming get_mysql_connection is defined elsewhere
6         cursor = connection.cursor()
7         # Start transaction
8         connection.begin()
9         print("Transaction started\n")
10        # Lock the row you are about to update
11        cursor.execute("SELECT * FROM doctor WHERE Doctor_ID = %s FOR UPDATE", (doctor_id,))
12        print("Row locked\n")
13        # Check if the doctor exists to prevent a phantom read
14        if cursor.rowcount == 0:
15            connection.rollback() # Rollback if no doctor found
16            return "Doctor not found", 404
17        # Proceed with update if the doctor exists
18
19        query = """UPDATE doctor SET
20                  First_Name = %s,
21                  Middle_Name = %s,
22                  Last_Name = %s,
23                  DOB = %s,
24                  Gender = %s,
25                  Email = %s,
26                  Specialization = %s,
27                  Experience = %s,
28                  Street_Number = %s,
29                  Street_Name = %s,
30                  Apartment_Number = %s,
31                  City = %s,
32                  Pincode = %s
33                  WHERE Doctor_ID = %s"""
34        cursor.execute(query, (first_name, middle_name, last_name, dobDate, gender, email, specialization, experience, street_number, street_name, apartment_number, city, pincode))
35        print("Query executed ", apartment_number, "\n")
36        connection.commit() # Commit the transaction
37    except OperationalError as e:
38        print(f"An error occurred: {e}")
39        connection.rollback() # Rollback in case of error
40    finally:
41        connection.close() # Close the connection
```

- Used test\_script.py to implement concurrent transactions. Here, we are editing a specific row of “Doctor” with multiple edit requests concurrently, here as Data1 and Data2 trying to edit Doctor entry with Doctor\_ID = “1401”.
- Data 1 = Apartment Number → 400
- Data 2 = Apartment Number → 401

```

File Edit Selection View Go Run Terminal Help ↵ → ⌘ ITGNDispensary-1
app.py edit_doctor.html test_script.py Dispensary.sql config.py
test_script.py > ...
1 # test_script.py
2 import requests
3 from concurrent.futures import ThreadPoolExecutor
4
5 def update_doc(doctor_id, data):
6     url = "http://localhost:5000/update_doctor" # Adjust if necessary
7     print("Updating doctor [doctor_id]")
8     headers = {'Content-Type': 'application/json'} # Set the content type to application/json
9     print("Sending request to [url]")
10    response = requests.post(url, json=data, headers=headers) # Use json parameter to send data as JSON
11    print("Response for doctor [doctor_id], completed!\n")
12
13 data1 = {'doctor_id': '1401', 'first_name': 'William', 'middle_name': 'Alexander', 'last_name': 'John', 'dob': '1983-05-21', 'gender': 'Male', 'email': 'william.powell@yahoo.com'}
14 data2 = {'doctor_id': '1401', 'first_name': 'William', 'middle_name': 'Alexander', 'last_name': 'John', 'dob': '1983-05-21', 'gender': 'Male', 'email': 'william.powell@yahoo.com'}
15 data3 = {'doctor_id': '1402', 'first_name': 'Kathryn', 'middle_name': 'Anthony', 'last_name': 'Manning', 'dob': '1965-01-04', 'gender': 'Male', 'email': 'kathryn.manning@outlook.com'}
16 print("Response for doctor ")
17
18 with ThreadPoolExecutor(max_workers=2) as executor:
19     executor.submit(update_doc, 1401, data1)
20     print("executing another!\n")
21     executor.submit(update_doc, 1401, data2)
22     print("executing another!\n")
23     executor.submit(update_doc, 1402, data3)
24

```

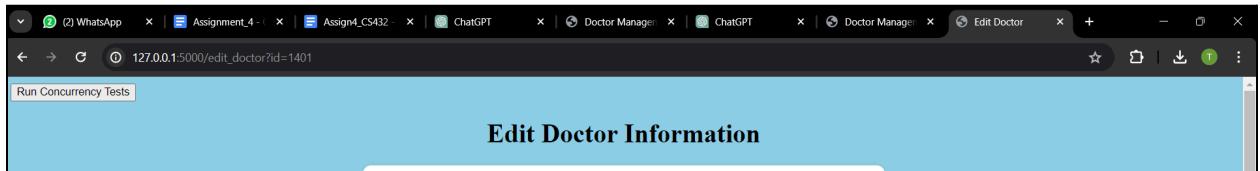
Ln 13, Col 304 Spaces: 4 UTF-8 CRLF ↴ MagicPython 3.12.1 64-bit

32°C Smoke

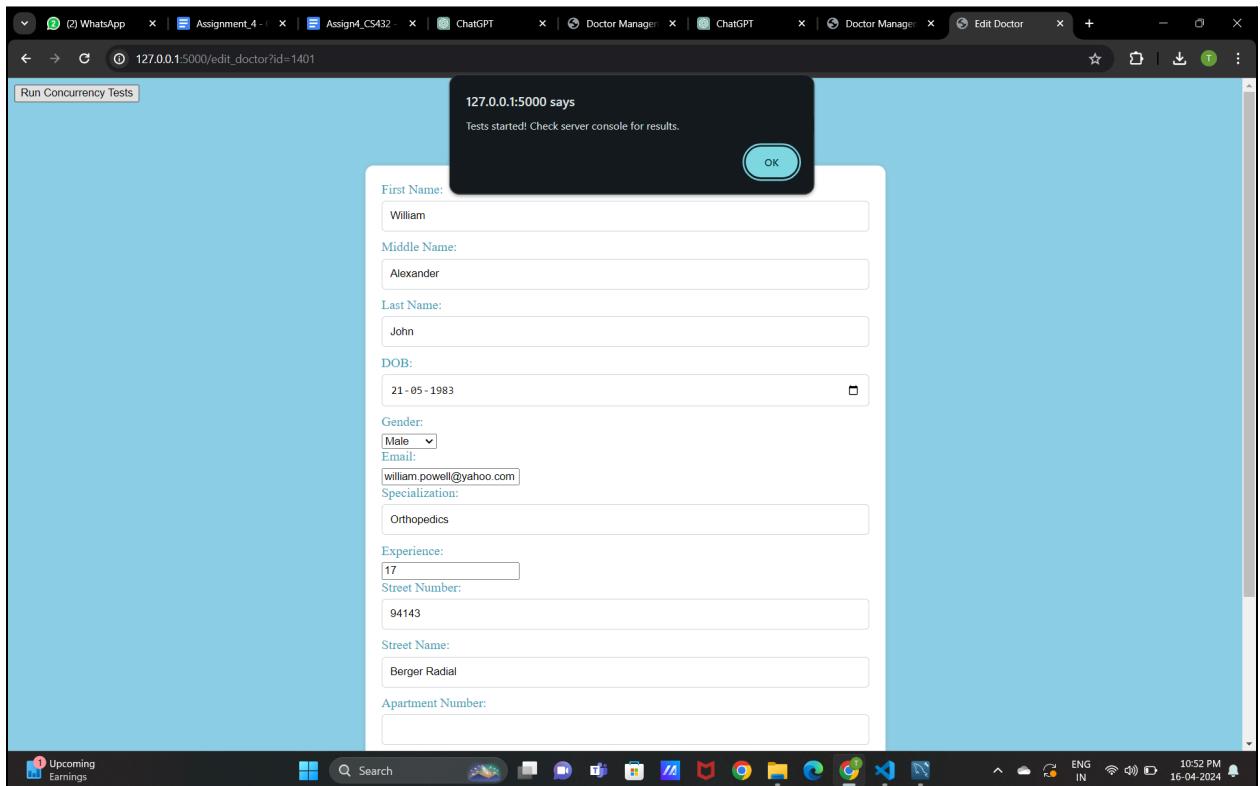
- Before concurrent editing, for Doctor\_ID = 1401 , Apartment Number = 896 & Doctor\_ID = 1402, Apartment Number = 954.

DOCTOR					
Actions	Apartment_Number	City	DOB	Doctor_ID	Email
Edit Delete OPD		Valsad	Sat, 09 Feb 2002 00:00:00 GMT	156	riyaj@gmail.com
Edit Delete OPD	896	Sandy	Sat, 21 May 1983 00:00:00 GMT	1401	william.powell@yahoo.com
Edit Delete OPD	954	Hopkinsstad	Mon, 04 Jan 1965 00:00:00 GMT	1402	kathryn.manning@outlook.com
Edit Delete OPD	84898	Port Raymond	Thu, 10 Nov 1977 00:00:00 GMT	1403	joshua.clark@outlook.com
Edit Delete OPD	4835	Port Spencer	Sat, 14 Aug 1965 00:00:00 GMT	1501	brianna.mckenzie@gmail.com
Edit Delete OPD	5	Brownmouth	Tue, 22 May 1990 00:00:00 GMT	1502	janet.simmons@gmail.com
Edit Delete OPD	9475	Joseburgh	Tue, 19 Mar 1963 00:00:00 GMT	1503	courtney.dean@outlook.com
Edit Delete OPD	320	East Ashleymouth	Thu, 21 Dec 1978 00:00:00 GMT	1601	brandon.miller@outlook.com
Edit Delete OPD	10523	Mannfort	Fri, 07 Dec 1973 00:00:00 GMT	1602	margaret.maddox@outlook.com
Edit Delete OPD	361	Arnoldview	Sat, 02 Mar 1991 00:00:00 GMT	1701	melissa.smith@outlook.com
Edit Delete OPD	14256	Port Patrick	Mon, 07 Aug 1989 00:00:00 GMT	1702	kathleen.graham@yahoo.com

- Running “Run Concurrency Tests”



- Test Started Confirmation



After editing, for Doctor\_ID = “1401”, Apartment Number gets updated to “400” and Doctor\_ID = “1402”, Apartment Number gets updated to “400”

- Updated Doctor Table entries for Doctor\_ID = 1401 & 1402

The screenshot shows a web application interface for managing doctor information. At the top, there are buttons for Back, Insert, Rename, and Filter. Below this is a table titled "DOCTOR" with the following data:

Actions	Apartment_Number	City	DOB	Doctor_ID	Email
Edit Delete OPD		Valsad	Sat, 09 Feb 2002 00:00:00 GMT	156	riyaj@gmail.com
Edit Delete OPD	400	Sandy	Sat, 21 May 1983 00:00:00 GMT	1401	william.powell@yahoo.com
Edit Delete OPD	400	Hopkinsstad	Mon, 04 Jan 1965 00:00:00 GMT	1402	kathryn.manning@outlook.com
Edit Delete OPD	84898	Port Raymond	Thu, 10 Nov 1977 00:00:00 GMT	1403	joshua.clark@outlook.com
Edit Delete OPD	4835	Port Spencer	Sat, 14 Aug 1965 00:00:00 GMT	1501	brianna.mckenzie@gmail.com
Edit Delete OPD	5	Brownmouth	Tue, 22 May 1990 00:00:00 GMT	1502	janet.simmons@gmail.com
Edit Delete OPD	9475	Joseburgh	Tue, 19 Mar 1963 00:00:00 GMT	1503	courtney.dean@outlook.com
Edit Delete OPD	320	East Ashleymouth	Thu, 21 Dec 1978 00:00:00 GMT	1601	brandon.miller@outlook.com
Edit Delete OPD	10523	Mannfort	Fri, 07 Dec 1973 00:00:00 GMT	1602	margaret.maddox@outlook.com
Edit Delete OPD	361	Arnoldview	Sat, 02 Mar 1991 00:00:00 GMT	1701	melissa.smith@outlook.com
Edit Delete OPD	14256	Port Patrick	Mon, 07 Aug 1989 00:00:00 GMT	1702	kathleen.graham@yahoo.com

- Output from terminal on executing the concurrent requests.

For Doctor\_ID = 1401, both data1 and data2 requests are being sent concurrently, but are being executed sequentially.

Here, in the output shown,

First Query (Data2) which has Apartment Number = 401 gets executed, while being executed, the row gets locked so that further queries can't be executed in the midway.

After transaction is successful, another query for Data 1 gets executed and that remains final.

```

Terminal Help ← → IITGDispensary-1
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS python + ⌂ … ×

Running the script...
127.0.0.1 - - [16/Apr/2024 22:57:29] "POST /run_tests HTTP/1.1" 200 -

Response for doctor
Updating doctor 1401
executing another!

Sending request to http://localhost:5000/update_doctor
Updating doctor 1401
Sending request to http://localhost:5000/update_doctor
executing another!

Request received

Data received: {"doctor_id": "1401", "first_name": "William", "middle_name": "Alexander", "last_name": "John", "dob": "1983-05-21", "gender": "Male", "email": "william.powell@yahoo.com", "specialization": "Orthopedics", "experience": "17", "street_number": "94143", "street_name": "Berger Radial", "apartment_number": "400", "city": "Sandy", "pincode": "549310"}

Request received

Data received: {"doctor_id": "1401", "first_name": "William", "middle_name": "Alexander", "last_name": "John", "dob": "1983-05-21", "gender": "Male", "email": "william.powell@yahoo.com", "specialization": "Orthopedics", "experience": "17", "street_number": "94143", "street_name": "Berger Radial", "apartment_number": "401", "city": "Sandy", "pincode": "549310"}

Transaction started
Transaction started
Row locked
Query executed 401
Row locked
127.0.0.1 - - [16/Apr/2024 22:57:32] "POST /update_doctor HTTP/1.1" 302 -
Query executed 400
127.0.0.1 - - [16/Apr/2024 22:57:32] "POST /update_doctor HTTP/1.1" 302 -
127.0.0.1 - - [16/Apr/2024 22:57:34] "GET /doctor HTTP/1.1" 200 -
127.0.0.1 - - [16/Apr/2024 22:57:34] "GET /doctor HTTP/1.1" 200 -
Response for doctor 1401,completed!
Updating doctor 1402
Sending request to http://localhost:5000/update_doctor
Response for doctor 1401,completed!

Request received

Data received: {"doctor_id": "1402", "first_name": "Kathryn", "middle_name": "Anthony", "last_name": "Manning", "dob": "1965-01-04", "gender": "Male", "email": "kathy.manning@outlook.com", "specialization": "Pediatrics", "experience": "31", "street_number": "97637", "street_name": "David Row", "apartment_number": "400", "city": "Hopkinsstad", "pincode": "434762"}

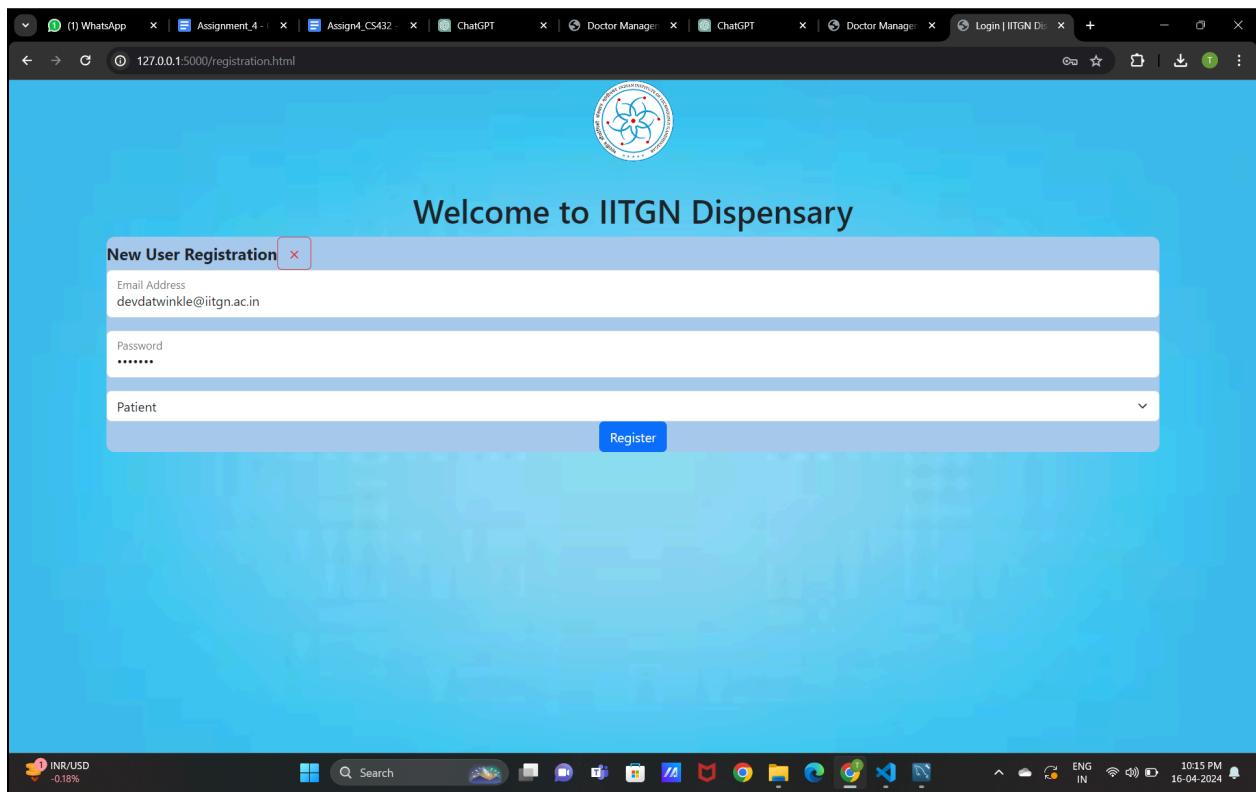
Transaction started
Row locked
Query executed 400
127.0.0.1 - - [16/Apr/2024 22:57:36] "POST /update_doctor HTTP/1.1" 302 -
127.0.0.1 - - [16/Apr/2024 22:57:38] "GET /doctor HTTP/1.1" 200 -
Response for doctor 1402,completed!
127.0.0.1 - - [16/Apr/2024 22:57:58] "GET /edit_doctor?id=1401 HTTP/1.1" 200 -
127.0.0.1 - - [16/Apr/2024 22:58:02] "GET /doctor HTTP/1.1" 200 -
127.0.0.1 - - [16/Apr/2024 22:58:02] "GET /get_doctor HTTP/1.1" 200 -

```

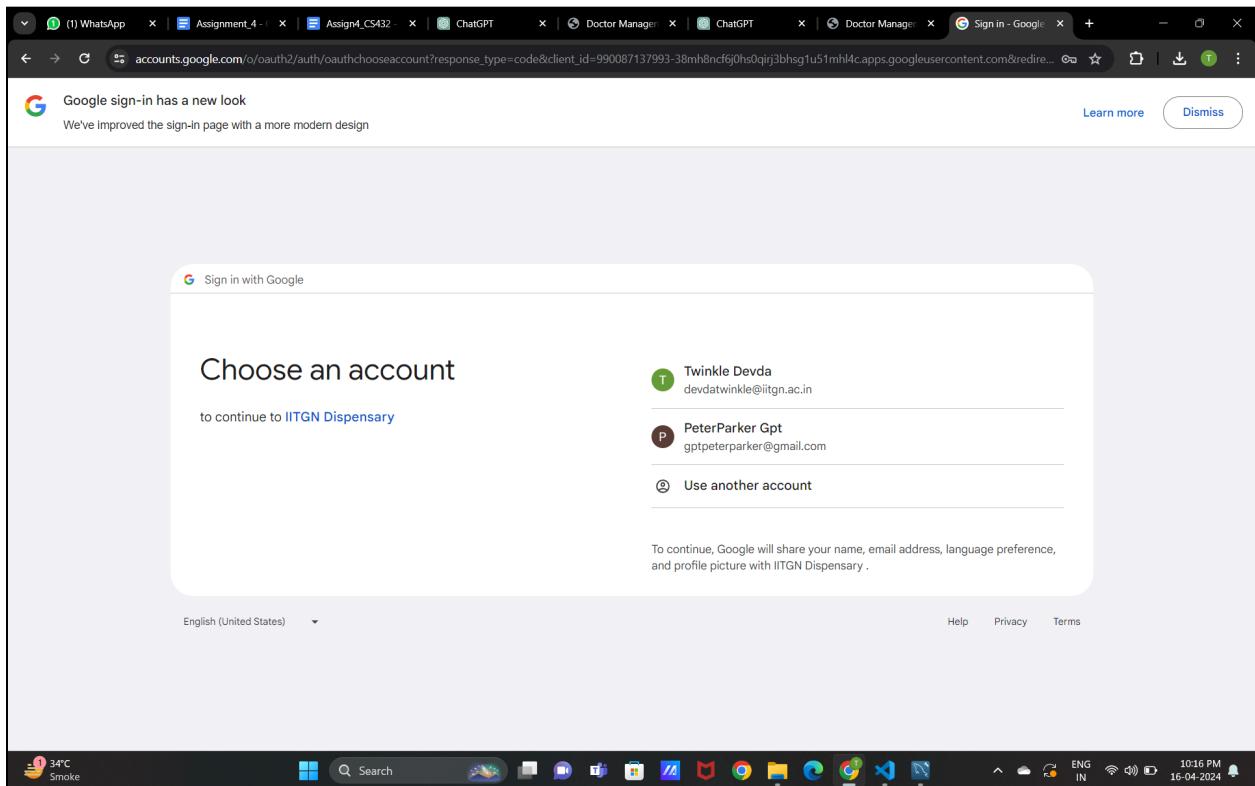
**Q2. Already done above (all the changes after the feedback has been compiled altogether by both the groups)**

**Q3. Google Authentication for Login & Register**  
(Only IITGN User can Login and Register in the Webapp)

- Creating a new user with email id “[devdatwinkle@iitgn.ac.in](mailto:devdatwinkle@iitgn.ac.in)” which is a valid IITGN user.



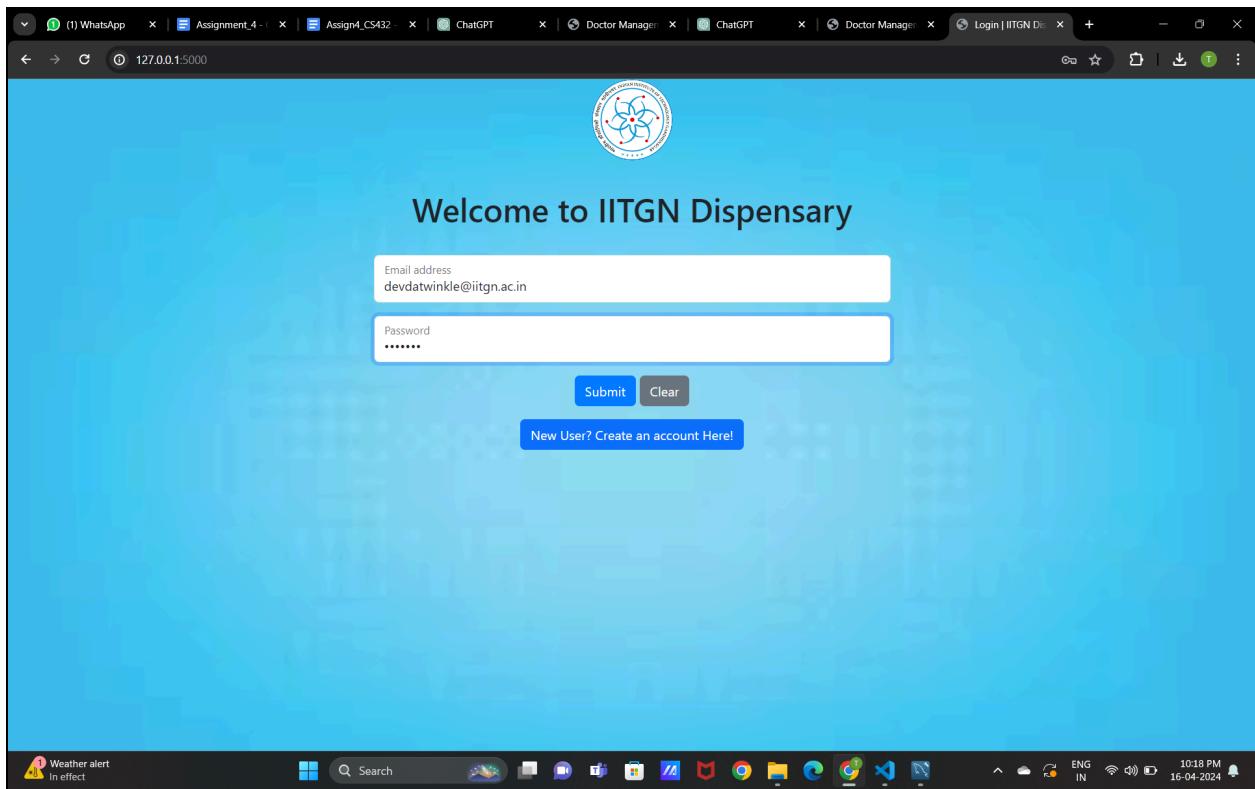
- Checking if the given email id is Valid Google email Id or not!



- New user gets added in Authorisation Table

Result Grid		
Email	Password	redirect_to
123@gmail	1234	index_d
abc@gmail.com	1234	index_d
devdatwinkle@iitgn.ac.in	Twinkle	index_p
doc@gmail.com	12345	index_d
hatt@gmail.com	12345	index
jg@gmail.com	1234	index_p
melindacontreras@example.org	1234	index_p
oall@example.com	1234	index_p
pat@gmail.com	12345	index_p
trial@gmail.com	1234	index_p
try@gmail.com	0987	index
xyz@gmail.com	1234	index
NULL	NULL	NULL

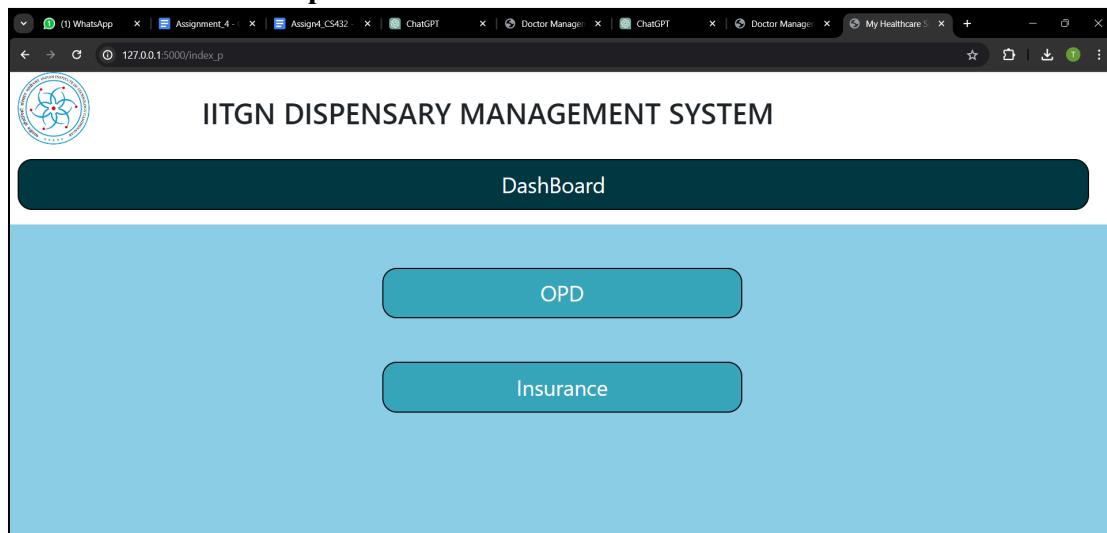
- Logging In in the newly created account



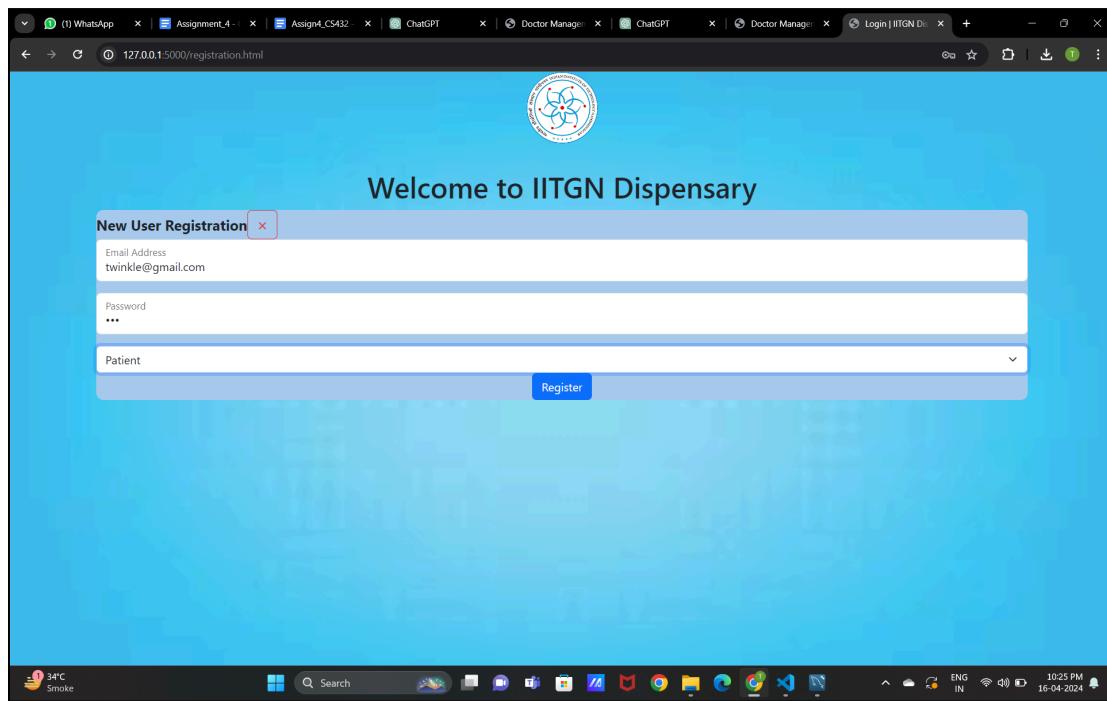
- Authority = ‘Patient’ as the user is Patient

A screenshot of a web browser showing the Patient Dashboard of the IITGN Dispensary Management System. The dashboard features a large image of a red brick building on a hillside overlooking a river. At the top, there is a navigation bar with links for "Back", "Home", "Team", "Contact Us", and "LogOut". The main content area is titled "Welcome to the Patient Dashboard". It contains a welcome message: "Welcome to the IITGN Dispensary System. This system is designed to manage the dispensary operations efficiently. You can use this system to manage patient records, inventory, appointments, and more." To the right of this message is a section titled "Our Team" which lists the names and IDs of the healthcare professionals: Saurabh Kumar Sah - 21110188, Shreya Patel - 21110155, Het Trivedi - 21110226, Riya Jain - 21110178, Twinkle Devda - 21110228, Ishika Raj - 21110081, Saumya Jaiswal - 21110186, and Darsh Dalal - 21110049. The browser's address bar shows the URL "127.0.0.1:5000/pat\_main". The system tray at the bottom of the screen shows the date and time as "16-04-2024 10:19 PM" and a weather icon indicating "34°C Smoke".

- Patient has access to patient dashboard

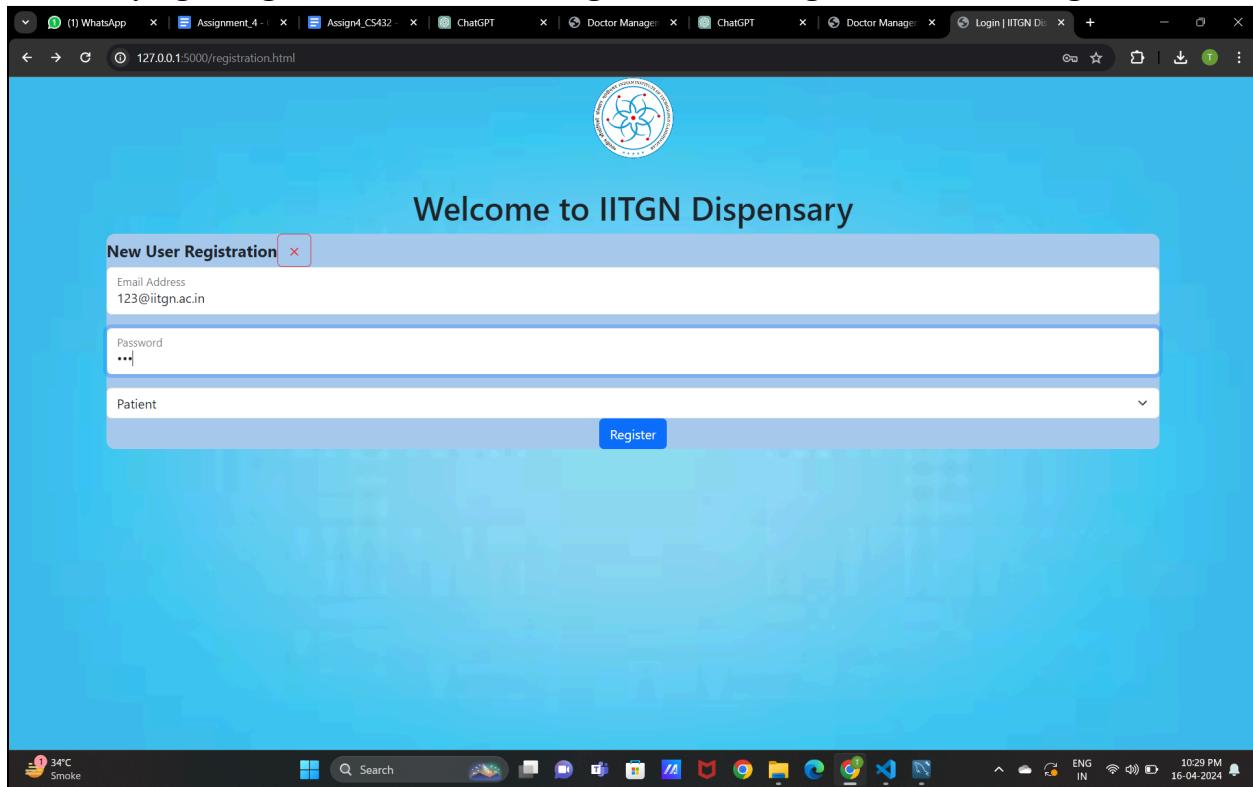


- Trying to Register with a false Email-id (twinkle@gmail.com)

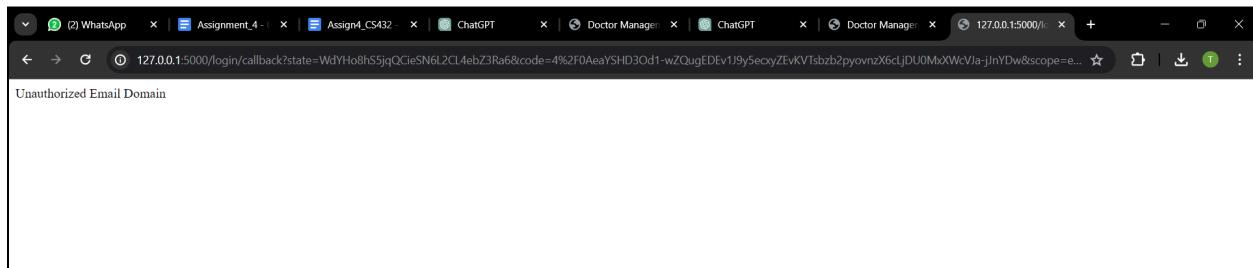


- Error because this email-id is not supported, only IITGN email id is accepted.

- Trying to register with a non existing email id having domain name “[iitgn.ac.in](#)”



- Since “[123@iitgn.ac.in](#)” is a non valid email-id, Google authorisation fails and we get error.



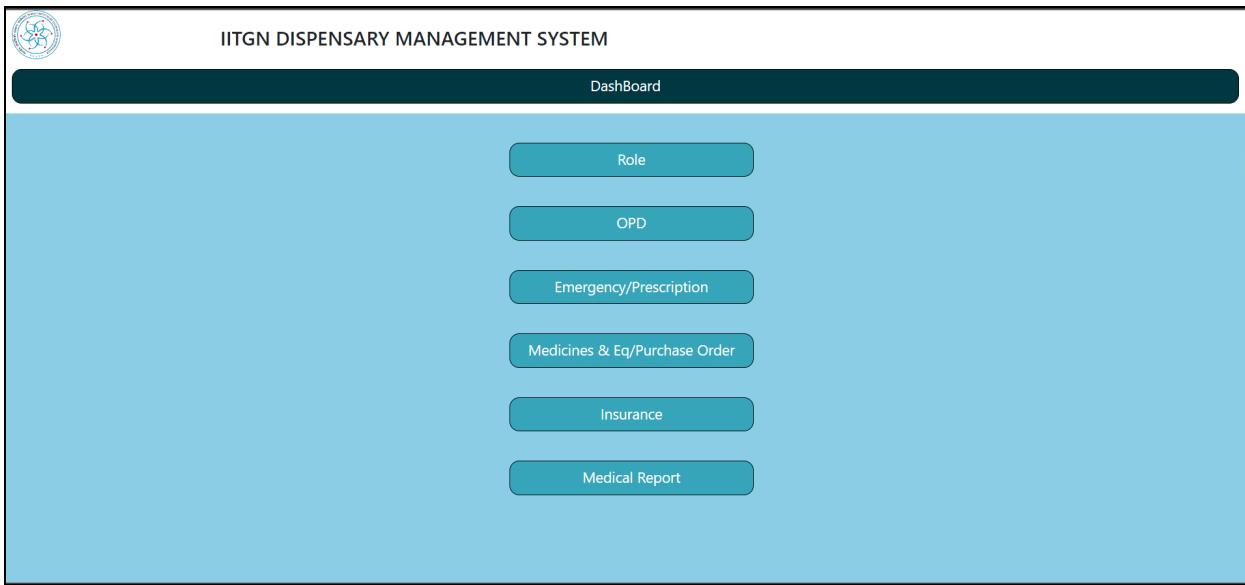
## Responsibility of G1 & G2

Q1.

- **SQL Injection:**

- An attacker can use a technique known as SQL Injection to insert malicious SQL code into a web application's query parameters or input fields.
- Generally, SQL Injection vulnerabilities arise when web applications create SQL queries using user-supplied data without conducting adequate validation or sanitisation.





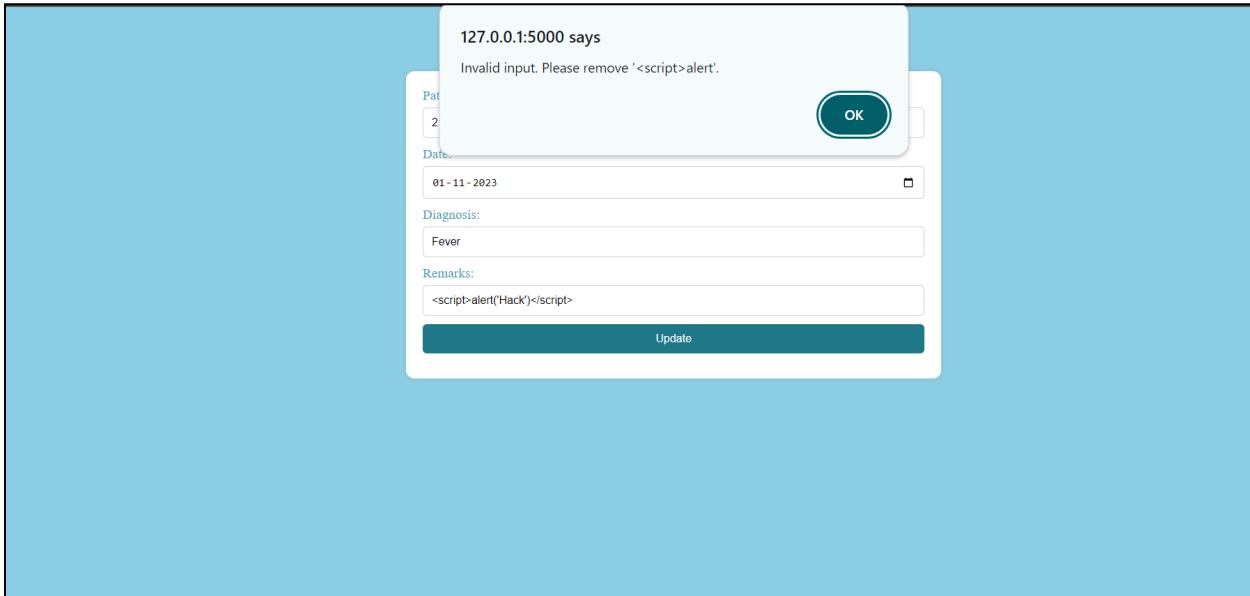
- **XSS:**

- Cross-site scripting (XSS) enables attackers to insert malicious scripts into web pages that other users are viewing.
- In the context of other users' browsers, the injected scripts have the ability to run arbitrary code, possibly stealing cookies, session tokens, or other private data, or acting as the user.
- Three types of XSS vulnerabilities can be distinguished: DOM-based XSS (where the vulnerability is present in client-side JavaScript code), stored XSS (where the injected script is stored on the server and executed when viewed by other users), and reflected XSS (where the injected script is reflected off a web server).

The screenshot shows a modal window titled "Edit Prescription Information". Inside the modal, there are four input fields: "Patient ID:" with value "21060", "Date:" with value "01-11-2023", "Diagnosis:" with value "Fever", and "Remarks:" with value "<script>alert('Hack')</script>". Below these fields is a teal-colored "Update" button.

- Without a defense, entering will cause a dialog box to appear and may cause the application to crash.

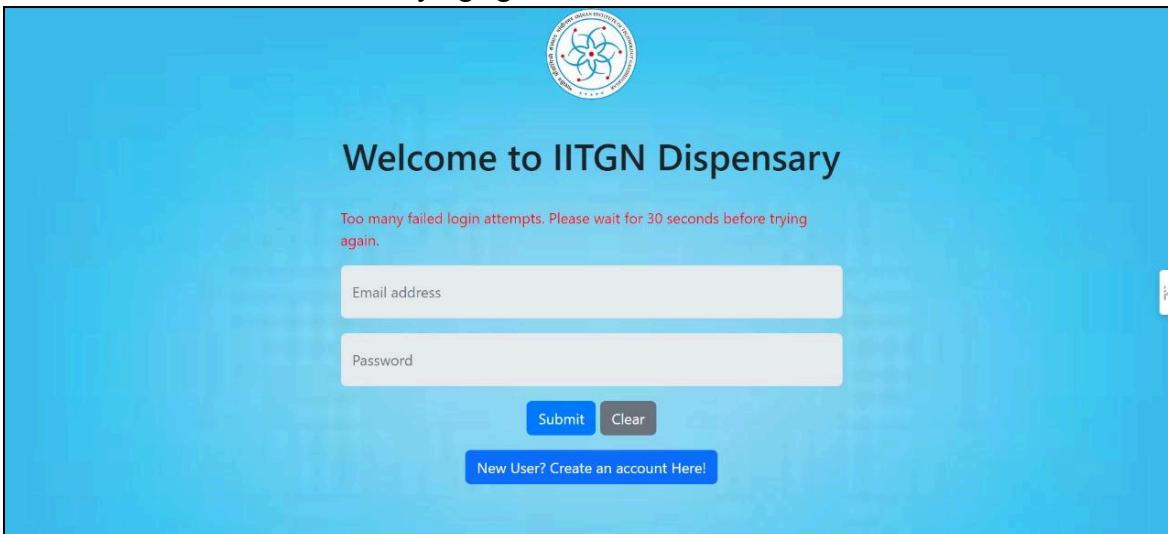
- We have included a dialog box that displays "invalid input. Please remove <script>alert" as a defense, preventing the user from inserting any dangerous content.



## BONUS ATTACK - BY G1 + G2

- **Brute Force:**

- A Brute Force Attack is a kind of cybersecurity attack in which the attacker deliberately tries every possible combination of characters, numbers, and symbols until they find the one that works, in an attempt to guess the target's encryption keys, login credentials, or other sensitive information.
- We have prevented this attack by limiting the user for 30 seconds after 3 login attempts and displaying the message “ Too many login attempts. Please wait for 30 seconds before trying again.”



## Q2.

Example 1) For gender, a drop-down box has been added so that values out of only these 3 can be added.

The screenshot shows a mobile application interface. On the left, there is a large blue vertical bar. To its right, the form fields are displayed. The first field is labeled "Gender:" in light blue text. Below it is a dropdown menu with four options: "Male", "Male" (which is highlighted in blue), "Female", and "Other". To the right of the dropdown is a field labeled "Phone Number:" in light blue text, which is currently empty. The overall background of the screen is white.

Example-2)

The screenshot shows a mobile application interface with several input fields. From top to bottom: 1. A field labeled "Street Name:" containing "ninin". 2. A field labeled "Apartment" containing "67". 3. A field labeled "City:" containing "nin". 4. A field labeled "Pincode:" containing "12345". 5. A field labeled "Day". A modal dialog box is overlaid on the screen, containing an error message: "(1366, "Incorrect integer value: 'asdas' for column 'Doctor\_ID' at row 1")". The modal has a close button ("X") in the top-left corner.

The primary key(Doctor\_ID), which should be an integer gave an error upon adding a non-integer value.

Example-3)

A screenshot of a mobile application interface. At the top, there is a header with the text "nin". Below it, there are two input fields: "Pincode:" containing "12345" and "Day" containing "1". A modal dialog box is displayed in the center, featuring a close button (X) at the top left. The main message in the dialog is: "(3819, "Check constraint 'doctor\_chk\_1' is violated.")".

For a pin code not of 6 digits, the constraint error appeared.

Example-4 )

A screenshot of a mobile application interface. At the top, there is a header with the text "Doctor ID:". Below it, there are three input fields: "First Name:" containing "nnn", "Middle Name:" containing "ihi", and "Doctor ID:" containing "1401". A modal dialog box is displayed in the center, featuring a close button (X) at the top left. The main message in the dialog is: "(1062, "Duplicate entry '1401' for key 'doctor.PRIMARY'")".

As we can see if a duplicate entry for the primary key (doctor\_id) is added then error message would appear.

Example-5)

The screenshot shows a form with a light blue header. On the left, there is a light blue vertical bar. To its right, the text "Email:" is displayed above a text input field containing "2121". A red error message box is overlaid on the form, containing an orange exclamation mark icon and the text "Please include an '@' in the email address. '2121' is missing an '@'." Below the input fields, there is another text input field containing "1234567890".

As we can see if @ is not included in the email, an error appears.

Example-6)

The screenshot shows a form with several input fields. The first field, "Phone Number:", contains "123456789". The second field, "Specialization:", contains "bnbjbj". The third field, "Experience:", contains "2". A red error message box is overlaid on the form, containing a red X icon and the text "(3819, "Check constraint 'doctor\_contact\_chk\_1' is violated.")".

For a mobile number not of 10 digits, the constraint error appeared.

### Contribution:

Name	Group	Contribution
Ishika Raj	G2	Multiple roles access, update databases without overlapping edits. Secure login and authorisation by Google, restricted to IITGN users, Documenting additional security attacks, Document SQL Injection, XSS attacks, and defenses, Documentation
Twinkle Devda	G2	Multiple roles access, update databases without overlapping edits. Secure login and authorisation by Google, restricted to IITGN users, Documenting additional security attacks, Document SQL Injection, XSS attacks, and defenses, Documentation
Saumya Jaiswal	G2	Multiple roles access, update databases without overlapping edits. Secure login and authorisation by Google, restricted to IITGN users, Documenting additional security attacks, Document SQL Injection, XSS attacks, and defenses.
Darsh Dalal	G2	Multiple roles access, update databases without overlapping edits. Secure login and authorisation by Google, restricted to IITGN users, Documenting additional security attacks, Document SQL Injection, XSS attacks, and defenses, ER Diagram Validation
Shreya Patel	G1	Document changes before and after stakeholder feedback, Documenting additional security

		attacks, Document SQL Injection, XSS attacks, and defenses. Showcase database access levels for different user classes, Documentation
Riya Jain	G1	Document changes before and after stakeholder feedback, Documenting additional security attacks, Document SQL Injection, XSS attacks, and defenses. Showcase database access levels for different user classes, Documentation
Trivedi HetKumar	G1	Document changes before and after stakeholder feedback, Documenting additional security attacks, Document SQL Injection, XSS attacks, and defenses. Showcase database access levels for different user classes, Secure login via Google, restricted to IITGN users, Documentation
Saurabh Kumar Sah	G1	Document changes before and after stakeholder feedback, Documenting additional security attacks, Document SQL Injection, XSS attacks, and defenses. Showcase database access levels for different user classes, Documentation