Geras

Connecting elderly to the care they deserve

TASK (Team 59)

- Samiksha Modi (2019331)
- Kabir Dureja (2019365)
- Aman Priyadarshi (2019294)
- Tushar Mahajan (2019280)
- Kunal Sagar (2019055)

Application/Domain

Senior Citizen welfare

Problem Area

Aged care is a significant issue which concerns all governments. In order to provide assistance and provide quality of life to senior citizens, enormous resources are invested by the government each year. A significant proportion of the investment is used to provide a range of services for senior citizens who stay at home. Since providing quality aged care is an ongoing and complex process, there is a desperate need to develop innovative solutions, which will benefit society at large. Although modern information technology products are changing the lifestyle of younger generations, they have much less impact on old people.

Idea

Senior Citizens play a very important role in our family and society. Their well being is very crucial for a family. In the current age, the families are not able to keep a constant check on the well being and needs of their elder one. The senior citizens are not well versed with technology which could be very useful for them in their daily lives. Our service plans to provide the senior citizens, their families, NGOs and the Government with an application to link all these actors for the betterment of the elderly.

We all are aware that everyone is not tech-savvy, especially the elderly who are often confused with new technology. Keeping this in mind, we designed our application in a way to eliminate the inconvenience caused to the elderly while using the technology.

Description

Our application “Geras” allows senior citizens and their families, NGOs, the government and other volunteers to connect and build a network. The government will be registered as an owner and administrator as they will assign officers and also fund the application while ensuring the smooth working of the services. NGOs will be registered as agencies partnering with the government on our application which will help in providing the services by assigning volunteers. On the other hand, the senior citizens, who are the beneficiary of our application, will be registered as users and the app will allow them to connect and avail the offered services without any hassle. In order to eliminate the concern of the families of the senior citizens, our application also serves as a portal for the families to get updates about the services the senior citizen requests and ensure their well being. Adding to this, our application will be open to any volunteer (who will be registered with the NGOs after the verification process) who wishes to join and help us improve our services by suggesting new innovative ideas.

Stakeholders

1. The Elderly

We presently have the largest population of youth and that is the bedrock of our future. However, we are soon also going to have perhaps the largest population of the elderly and get the label of a Geriatric Nation. But these elderlies have been the backbone of our nation and we must ensure that it remains strong - physically as well as mentally. We the young people and the government must now help them in their sensitive phase, providing them the care and comfort to lead a healthy and dignified life without worries and anxiety.

The elderly can easily connect to services that the government and the NGOs provide through our application.

Queries

- Register and add their details
- Unregister from the app
- Request a service
- Track the service request
- Update their prescription
- Access their prescription history
- List all completed services
- List all pending services
- See service history

- SOS

2. Volunteer

Volunteering holds a very important role in a community as it helps in holding the community together. Hence, our application is not only limited to registered members, but it is also open for volunteers. With their enthusiasm, the volunteers can be a helping hand for the elderly.

The volunteers are assigned the area in which they are willing to and qualified for the task. The volunteers can register with the registered NGOs in the database.

Queries

- Register and add their details
- Register with an NGO
- Unregister from an NGO
- Unregister from the app
- View the work allotted to them
- Report completion of their work
- Update their availability

3. NGOs

A non governmental organization (NGO) is a non profit that functions independently of any government, so as to serve a social or political goal such as humanitarian causes or environment. NGOs are instrumental in the social development of a state, nation or community. They act as a common link between government, families, volunteers and senior citizens. Through our application, the NGOs can effectively organize and execute their data and functions respectively, thereby amplifying their reach.

The NGOs can employ our database in order to fulfill the services required by the elderly by assigning volunteers to them. Besides, they can also request medical check-up for the elderly in the nearest government hospital because the government is also one of the stakeholders. The NGOs can even interact with the families of the elderly and discuss the further measures in order to maximize the well-being of elderly (feedback from families).

Queries

- Register and add their details

- Unregister as NGO
- Receive service request from the government
- Allot available volunteers to the work
- Check the status of the work allotted
- Access transactions with the government

4. Government

The government plays a very important role in connecting the Elderly, NGO and it's officers. The government will serve as a supervisor and administrator as they will assign officers and also fund the application while ensuring the smooth working of the services. They are responsible for keeping a check on all the services.

Our application will help the government officers like the admin of the database to maintain the list of NGO that the Elderly can connect to. And government officers like doctors and nurses to keep regular check-ups on the elderly.

Queries

- Register all the officers and their designation
- Create/Delete/Update Elderly, Healthcare Workers and NGOs partnered with them
- Access all service request
- Assign healthcare workers for service request
- Assign NGOs for service request
- Mark service as completed
- Access transaction history

5. Healthcare Workers

Healthcare workers play a very vital role in our daily lives. They save lives but their importance goes far beyond that. They make a difference by helping patients minimize pain, recover from a disease faster or learn to live with a disabling injury. The profession of a doctor is the most noble and respected one. They do a great service to society.
Through our app, doctors and nurses can keep the track of their elderly patients and employ proper measures to ensure the well being of their patients.

Queries

- Register and add their details
- Access prescription history of an Elderly
- Report completion of their work
- Contact family in case of an emergency
- Access transactions with the government

Families

The importance of family in a senior's life is truly immeasurable. Family is often the one connection that remains constant. Whether the senior citizen lives with or away from their family, the family is always concerned about the well being of their elderly. Our application would be a way that would help the families to have a better connection with their elderly by providing them various services that would help them monitor them.

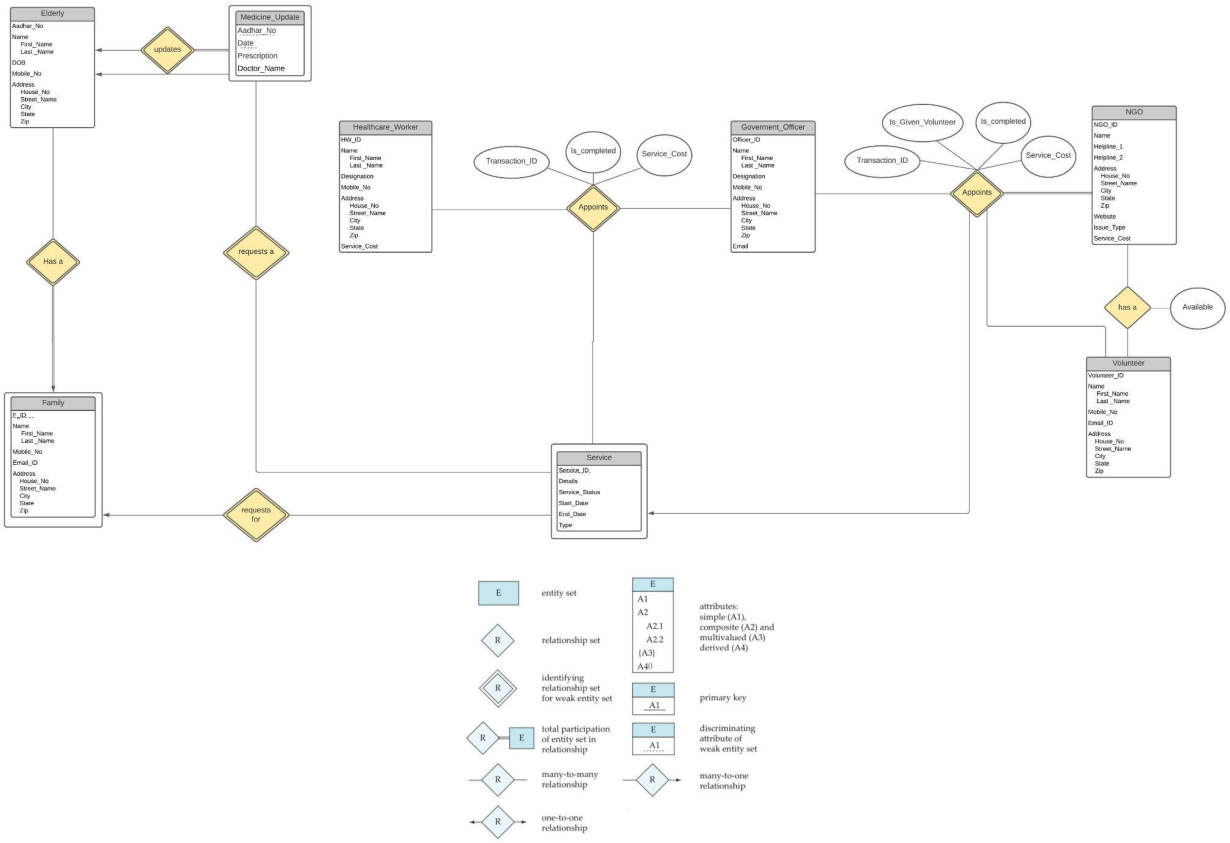
Queries

- Register and add their details
- Get notified when the Elderly requests any service
- View medicine record of elderly
- View healthcare worker currently providing services to the elderly
- Get notified in case of emergencies by the Healthcare Workers
- Request service for/on behalf of the elderly
- Track the service request

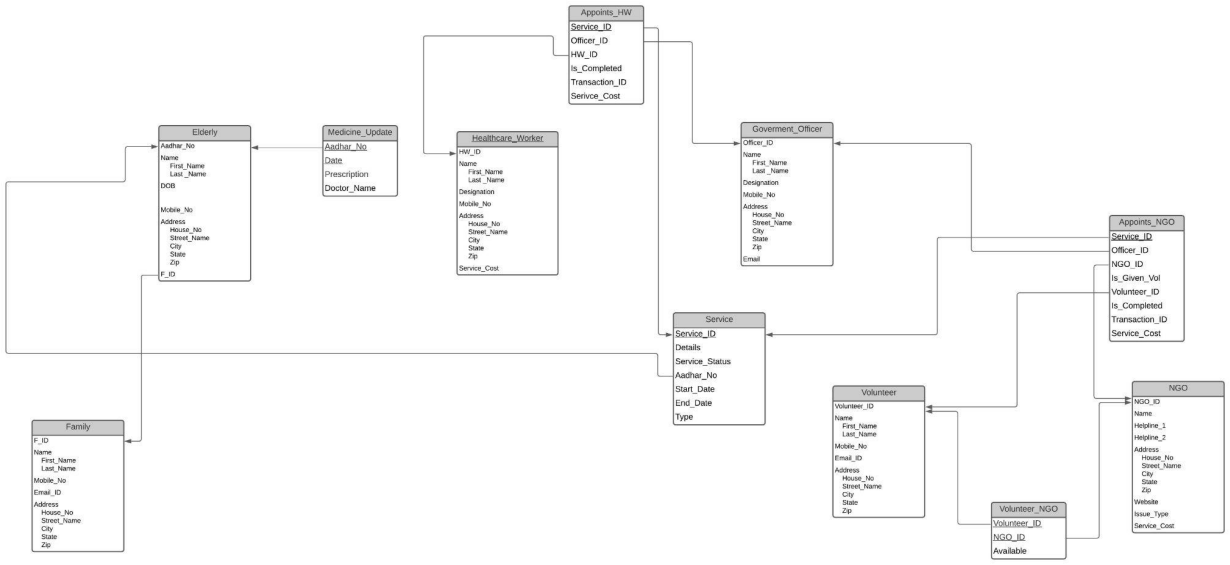
Project Inspiration

https://www.livemint.com/Politics/VbUrZerAH0rYgpFaDl8jsM/CGHS-beneficiaries-aged-80-and-above-to-get-easy-access-to-d.html

E-R Diagram



Schema Diagram



Tables and Attributes

1. **FAMILY:**

```
CREATE TABLE Family (
        F_ID varchar(12) NOT NULL UNIQUE,
        First_Name varchar(40) NOT NULL,
        Last_Name varchar(40) NOT NULL,
        Mobile_No varchar(10) NOT NULL UNIQUE,
        Email_ID varchar(80) NOT NULL UNIQUE,
        House_No varchar(10) NOT NULL,
        Street_Name varchar(50) NOT NULL,
        City varchar(50) NOT NULL,
        State varchar(50) NOT NULL,
        Zip varchar(6) NOT NULL,
        PRIMARY KEY(F_ID)
);
```

2. **ELDERLY:**

```
CREATE TABLE Elderly (
        Aadhar_No varchar(12) NOT NULL UNIQUE,
        First_Name varchar(50) NOT NULL,
        Last_Name varchar(50) NOT NULL,
        DOB date NOT NULL,
        Mobile_No varchar(10) NOT NULL UNIQUE,
        House_No varchar(10) NOT NULL,
        Street_Name varchar(50) NOT NULL,
        City varchar(50) NOT NULL,
        State varchar(50) NOT NULL,
        Zip varchar(6) NOT NULL,
        F_ID varchar(12),
        PRIMARY KEY(Aadhar_No)

);
```

3. **MEDICINE_UPDATE:**

```
CREATE TABLE Medicine_Update (
        Aadhar_No varchar(12) NOT NULL,
        Date date NOT NULL,
        Prescription varchar(400) NOT NULL,
        Doctor_Name varchar(50) NOT NULL,
        PRIMARY KEY(Aadhar_No,Date)
);
```

4. **HEALTHCARE_WORKER:**

```
CREATE TABLE Healthcare_Worker (
        HW_ID varchar(12) NOT NULL,
        First_Name varchar(40) NOT NULL,
        Last_Name varchar(40) NOT NULL,
        Designation varchar(30) NOT NULL,
        Mobile_No varchar(10) NOT NULL UNIQUE,
        House_No varchar(10) NOT NULL,
        Street_Name varchar(50) NOT NULL,
        City varchar(50) NOT NULL,
        State varchar(50) NOT NULL,
        Zip varchar(6) NOT NULL,
        Service_Cost varchar(10) NOT NULL,
        PRIMARY KEY(HW_ID)
);
```

5. **GOVERNMENT_OFFICER:**

```
CREATE TABLE Government_Officer (
        Officer_ID varchar(12) NOT NULL,
        First_Name varchar(40) NOT NULL,
        Last_Name varchar(34) NOT NULL,
        Designation varchar(30) NOT NULL,
        Mobile_No varchar(10) NOT NULL UNIQUE,
        House_No varchar(10) NOT NULL,
        Street_Name varchar(50) NOT NULL,
        City varchar(50) NOT NULL,
```

```
        State varchar(50) NOT NULL,
        Zip varchar(6) NOT NULL,
        Email varchar(80) NOT NULL UNIQUE,
        PRIMARY KEY(Officer_ID)
);
```

6. **SERVICE:**

```
CREATE TABLE Service(
        Service_ID varchar(12) NOT NULL,
        Details varchar(400) NOT NULL,
        Service_Status varchar(10) NOT NULL,
        Aadhar_No VARCHAR(12) NOT NULL,
        Start_Date date NOT NULL,
        End_Date date,
        Type varchar(5),
        PRIMARY KEY(Service_ID)
);
```

7. **APPOINTS_HW:**

```
CREATE TABLE Appoints_HW (
        Service_ID varchar(12) NOT NULL UNIQUE,
        Officer_ID varchar(12) NOT NULL,
        HW_ID varchar(12) NOT NULL,
        Is_Completed VARCHAR(10) NOT NULL,
        Transaction_ID varchar(12) UNIQUE,
        Service_Cost varchar(7) NOT NULL,
        PRIMARY KEY(Service_ID)
);
```

8. **APPOINTS_NGO:**

```
CREATE TABLE Appoints_NGO (
        Service_ID varchar(12) NOT NULL,
        Officer_ID varchar(12) NOT NULL,
        NGO_ID varchar(12) NOT NULL,
```

```
    Is_Given_Vol VARCHAR(10) NOT NULL,
    Volunteer_ID varchar(12),
    Is_Completed VARCHAR(10) NOT NULL,
    Transaction_ID varchar(12) UNIQUE,
    Service_Cost varchar(4) NOT NULL,
    PRIMARY KEY(Service_ID)
);
```

9. **VOLUNTEER_NGO:**

```
CREATE TABLE Volunteer_NGO (
    Volunteer_ID varchar(12) NOT NULL,
    NGO_ID varchar(12) NOT NULL,
    Available varchar(10) NOT NULL,
    PRIMARY KEY(Volunteer_ID,NGO_ID)
);
```

10. **VOLUNTEER:**

```
CREATE TABLE Volunteer (
    Volunteer_ID varchar(12) NOT NULL,
    First_Name varchar(40) NOT NULL,
    Last_Name varchar(40) NOT NULL,
    Mobile_No varchar(10) NOT NULL UNIQUE,
    Email_ID varchar(80) NOT NULL UNIQUE,
    House_No varchar(10) NOT NULL,
    Street_Name varchar(50) NOT NULL,
    City varchar(50) NOT NULL,
    State varchar(50) NOT NULL,
    Zip varchar(6) NOT NULL,
    PRIMARY KEY(Volunteer_ID)
);
```

11. NGO:

```
CREATE TABLE NGO (
        NGO_ID varchar(12) NOT NULL,
        Name varchar(30) NOT NULL,
        Helpline_1 varchar(11) NOT NULL UNIQUE,
        Helpline_2 varchar(11) UNIQUE,
        House_No varchar(10) NOT NULL,
        Street_Name varchar(50) NOT NULL,
        City varchar(50) NOT NULL,
        State varchar(50) NOT NULL,
        Zip varchar(6) NOT NULL,
        Website varchar(50) UNIQUE,
        Issue_Type varchar(30) NOT NULL,
        Service_Cost varchar(5) NOT NULL,
        PRIMARY KEY(NGO_ID)
);
```

Setting Foreign Key Constraints

1. ALTER TABLE Elderly ADD FOREIGN KEY (F_ID) REFERENCES Family(F_ID);

2. ALTER TABLE Service ADD FOREIGN KEY (Aadhar_No) REFERENCES Elderly (Aadhar_No);

3. ALTER TABLE Appoints_NGO ADD FOREIGN KEY (Service_ID) REFERENCES Service (Service_ID);

4. ALTER TABLE Appoints_NGO ADD FOREIGN KEY (Officer_ID) REFERENCES Government_Officer (Officer_ID);

5. ALTER TABLE Appoints_NGO ADD FOREIGN KEY (NGO_ID) REFERENCES NGO (NGO_ID);

6. ALTER TABLE Appoints_NGO ADD FOREIGN KEY (Volunteer_ID) REFERENCES Volunteer (Volunteer_ID);

7. ALTER TABLE Volunteer_NGO ADD FOREIGN KEY (Volunteer_ID) REFERENCES Volunteer (Volunteer_ID);

8. ALTER TABLE Volunteer_NGO ADD FOREIGN KEY (NGO_ID) REFERENCES NGO (NGO_ID);

9. ALTER TABLE Appoints_HW ADD FOREIGN KEY (Service_ID) REFERENCES Service (Service_ID);

10. ALTER TABLE Appoints_HW ADD FOREIGN KEY (Officer_ID) REFERENCES Government_Officer (Officer_ID);

11. ALTER TABLE Appoints_HW ADD FOREIGN KEY (HW_ID) REFERENCES Healthcare_Worker (HW_ID);

12. ALTER TABLE Medicine_Update ADD FOREIGN KEY (Aadhar_No) REFERENCES Elderly (Aadhar_No);

Tables Included in the Database

```
mysql> show tables;
+--------------------+
| Tables_in_geras    |
+--------------------+
| appoints_hw        |
| appoints_ngo       |
| elderly            |
| family             |
| government_officer |
| healthcare_worker  |
| medicine_update    |
| ngo                |
| service            |
| volunteer          |
| volunteer_ngo      |
+--------------------+
11 rows in set (0.00 sec)
```

```
mysql> show columns from elderly;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| Aadhar_No   | varchar(12) | NO   | PRI | NULL    |       |
| First_Name  | varchar(50) | NO   |     | NULL    |       |
| Last_Name   | varchar(50) | NO   |     | NULL    |       |
| DOB         | date        | NO   |     | NULL    |       |
| Mobile_No   | varchar(10) | NO   | UNI | NULL    |       |
| House_No    | varchar(10) | NO   |     | NULL    |       |
| Street_Name | varchar(50) | NO   |     | NULL    |       |
| City        | varchar(50) | NO   |     | NULL    |       |
| State       | varchar(50) | NO   |     | NULL    |       |
| Zip         | varchar(6)  | NO   |     | NULL    |       |
| F_ID        | varchar(12) | YES  | MUL | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
11 rows in set (0.00 sec)

mysql> show columns from medicine_update;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| Aadhar_No    | varchar(12)  | NO   | PRI | NULL    |       |
| Date         | date         | NO   | PRI | NULL    |       |
| Prescription | varchar(400) | NO   |     | NULL    |       |
| Doctor_Name  | varchar(50)  | NO   |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> show columns from family;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| F_ID        | varchar(12) | NO   | PRI | NULL    |       |
| First_Name  | varchar(40) | NO   |     | NULL    |       |
| Last_Name   | varchar(40) | NO   |     | NULL    |       |
| Mobile_No   | varchar(10) | NO   | UNI | NULL    |       |
| Email_ID    | varchar(80) | NO   | UNI | NULL    |       |
| House_No    | varchar(10) | NO   |     | NULL    |       |
| Street_Name | varchar(50) | NO   |     | NULL    |       |
| City        | varchar(50) | NO   |     | NULL    |       |
| State       | varchar(50) | NO   |     | NULL    |       |
| Zip         | varchar(6)  | NO   |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
10 rows in set (0.00 sec)
```

```
mysql> show columns from healthcare_worker;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| HW_ID        | varchar(12) | NO   | PRI | NULL    |       |
| First_Name   | varchar(40) | NO   |     | NULL    |       |
| Last_Name    | varchar(40) | NO   |     | NULL    |       |
| Designation  | varchar(30) | NO   |     | NULL    |       |
| Mobile_No    | varchar(10) | NO   | UNI | NULL    |       |
| House_No     | varchar(10) | NO   |     | NULL    |       |
| Street_Name  | varchar(50) | NO   |     | NULL    |       |
| City         | varchar(50) | NO   |     | NULL    |       |
| State        | varchar(50) | NO   |     | NULL    |       |
| Zip          | varchar(6)  | NO   |     | NULL    |       |
| Service_Cost | varchar(10) | NO   |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
11 rows in set (0.00 sec)

mysql> show columns from government_officer;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| Officer_ID  | varchar(12) | NO   | PRI | NULL    |       |
| First_Name  | varchar(40) | NO   |     | NULL    |       |
| Last_Name   | varchar(34) | NO   |     | NULL    |       |
| Designation | varchar(30) | NO   |     | NULL    |       |
| Mobile_No   | varchar(10) | NO   | UNI | NULL    |       |
| House_No    | varchar(10) | NO   |     | NULL    |       |
| Street_Name | varchar(50) | NO   |     | NULL    |       |
| City        | varchar(50) | NO   |     | NULL    |       |
| State       | varchar(50) | NO   |     | NULL    |       |
| Zip         | varchar(6)  | NO   |     | NULL    |       |
| Email       | varchar(80) | NO   | UNI | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
11 rows in set (0.00 sec)
```

```
mysql> show columns from service;
+----------------+--------------+------+-----+---------+-------+
| Field          | Type         | Null | Key | Default | Extra |
+----------------+--------------+------+-----+---------+-------+
| Service_ID     | varchar(12)  | NO   | PRI | NULL    |       |
| Details        | varchar(400) | NO   |     | NULL    |       |
| Service_Status | varchar(10)  | NO   |     | NULL    |       |
| Aadhar_No      | varchar(12)  | NO   |     | NULL    |       |
| Start_Date     | date         | NO   |     | NULL    |       |
| End_Date       | date         | YES  |     | NULL    |       |
| Type           | varchar(5)   | YES  |     | NULL    |       |
+----------------+--------------+------+-----+---------+-------+
7 rows in set (0.00 sec)

mysql> show columns from appoints_hw;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| Service_ID     | varchar(12) | NO   | PRI | NULL    |       |
| Officer_ID     | varchar(12) | NO   | MUL | NULL    |       |
| HW_ID          | varchar(12) | NO   | MUL | NULL    |       |
| Is_Completed   | varchar(10) | NO   |     | NULL    |       |
| Transaction_ID | varchar(12) | YES  | UNI | NULL    |       |
| Service_Cost   | varchar(7)  | NO   |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
6 rows in set (0.00 sec)

mysql> show columns from appoints_ngo;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| Service_ID     | varchar(12) | NO   | PRI | NULL    |       |
| Officer_ID     | varchar(12) | NO   | MUL | NULL    |       |
| NGO_ID         | varchar(12) | NO   | MUL | NULL    |       |
| Is_Given_Vol   | varchar(10) | NO   |     | NULL    |       |
| Volunteer_ID   | varchar(12) | YES  | MUL | NULL    |       |
| Is_Completed   | varchar(10) | NO   |     | NULL    |       |
| Transaction_ID | varchar(12) | YES  | UNI | NULL    |       |
| Service_Cost   | varchar(4)  | NO   |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

```
mysql> show columns from volunteer_ngo;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| Volunteer_ID | varchar(12) | NO   | PRI | NULL    |       |
| NGO_ID       | varchar(12) | NO   | PRI | NULL    |       |
| Available    | varchar(10) | NO   |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> show columns from volunteer;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| Volunteer_ID | varchar(12) | NO   | PRI | NULL    |       |
| First_Name   | varchar(40) | NO   |     | NULL    |       |
| Last_Name    | varchar(40) | NO   |     | NULL    |       |
| Mobile_No    | varchar(10) | NO   | UNI | NULL    |       |
| Email_ID     | varchar(80) | NO   | UNI | NULL    |       |
| House_No     | varchar(10) | NO   |     | NULL    |       |
| Street_Name  | varchar(50) | NO   |     | NULL    |       |
| City         | varchar(50) | NO   |     | NULL    |       |
| State        | varchar(50) | NO   |     | NULL    |       |
| Zip          | varchar(6)  | NO   |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
10 rows in set (0.00 sec)

mysql> show columns from ngo;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| NGO_ID       | varchar(12) | NO   | PRI | NULL    |       |
| Name         | varchar(30) | NO   |     | NULL    |       |
| Helpline_1   | varchar(11) | NO   | UNI | NULL    |       |
| Helpline_2   | varchar(11) | YES  | UNI | NULL    |       |
| House_No     | varchar(10) | NO   |     | NULL    |       |
| Street_Name  | varchar(50) | NO   |     | NULL    |       |
| City         | varchar(50) | NO   |     | NULL    |       |
| State        | varchar(50) | NO   |     | NULL    |       |
| Zip          | varchar(6)  | NO   |     | NULL    |       |
| Website      | varchar(50) | YES  | UNI | NULL    |       |
| Issue_Type   | varchar(30) | NO   |     | NULL    |       |
| Service_Cost | varchar(5)  | NO   |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
12 rows in set (0.00 sec)
```

Indexing Commands

1. Create Index trans_appoints_NGO
   ON Appoints_NGO(Service_Cost, Transaction_ID);

2. Create Index trans_appoints_HW
   ON Appoints_HW(Service_Cost, Transaction_ID);

3. Create Index officer_appoints_HW
   ON Appoints_HW(HW_ID, Officer_ID);

4. Create Index available_ngos_with_service_cost
   ON NGO(NGO_ID, Service_Cost, Status);

5. Create Index track_elderly_service
   ON Elderly(Aadhar_No, Service_ID);

6. Create Index elderly_aadhar
   ON Elderly(Aadhar_No);

7. Create Index completed_elderly_service
   ON Elderly(Aadhar_No, Service_Status);

8. Create Index services_info
   ON Services(Service_ID, Service_Status, Type);

9. Create Index active_volunteers
   ON Volunteer(Volunteer_ID, Status);

10. Create Index active_HW
    ON Volunteer(HW_ID, Status);

Relational Algebra Queries

1. **Track service request of an elderly**
   Let SER1 be the service request of elderly with aadhar no 123456:

   Select * from SERVICE where aadhar_no='123456' and Service_ID = 'SER1';

2. **Display prescription history of an elderly**
   Let elderly aadhar be 123456:
   $\sigma_{Aadhar\_No\ =\ '123456'}$ (Medicine_Update)

   Select * from Medicine_Update where Aadhar_No="123456"

3. **List all completed services of an elderly**
   Let eldelry aadhar be 123456:
   $\sigma_{Aadhar\_No\ =\ '123456'\ \wedge\ Service\_Status\ =\ 'Completed'}$ (Service)

   Select * from Service where Aadhar_No="123456" and Service_Status="Completed"

4. **List all pending services of an elderly**
   Let eldelry aadhar be 123456:
   $\sigma_{Aadhar\_NO\ =\ '123456'\ \wedge\ Service\_Status\ =\ 'Inprogress"}$ (Service) $\cup$ $\sigma_{Aadhar\_No\ =\ '123456'\ \wedge\ Service\_Status\ =\ "Requested"}$ (Service)

   Select * from Service where Aadhar_No="123456" and (Service_Status = "Inprogress" or Service_Status="Requested")

5. **List service history of an elderly**
   Let elderly aadhar be 123456:

   Select * from Service where Aadhar_No="123456"

6. **List all services of an elderly which were completed on the same day it was requested**
   $\prod_{Service\_ID,Details}(\sigma_{Start\_Date\ =\ End\_Date\ \wedge\ Service\_Status\ =\ 'Completed'}(Service))$

   Select Service_ID, Details from Service where Service_Status="Completed" and Start_Date=End_Date

7. **List all HW workers appointed by a govt officer**
   Let officer id be 'OFF2'

$\prod_{HW\_ID}(\sigma_{Officer\_ID = 'OFF2'}(Appoints\_HW))$

Select HW_ID from Appoints_HW where Officer_ID="OFF2"

8. **<u>Display all service details assigned to a HW Worker</u>**
   Let HW id be 'HW2'
   $\sigma_{HW\_ID = 'HW2'}(Appoints\_HW \bowtie Service)$

   Select * from Appoints natural join Service where HW_ID="HW2"

9. **<u>Display details of all volunteers in an NGO</u>**
   Let NGO id be 'NGO1'
   $\sigma_{NGO\_ID = 'NGO1'}(Volunteer \bowtie Volunteer\_NGO)$

   Select * from Volunteer natural join. Volunteer_NGO where NGO_ID="NGO1"

10. **<u>Display all active NGOs</u>**
    $\prod_{NGO\_ID}(\sigma_{Status='active'}(NGO))$

    Select NGO_ID from NGO where Status="active"

11. **<u>List pending services of a HW worker</u>**
    Let HW id be 'HW2'
    $\sigma_{HW\_ID = 'HW2' \wedge Is\_completed='no'}(Appoints\_HW)$

12. **<u>List all the transactions of a HW with the Government</u>**
    Let HW id be 'HW2'
    $\prod_{Transaction\_ID,Service\_Cost}(\sigma_{HW\_ID = 'HW2'}(Appoints\_HW))$

    Select Transaction _ID, Service_Cost from Appoints_HW where HW_ID=2

13. **<u>Display the list of available NGOs and their service cost</u>**
    $\prod_{NGO\_ID,Service\_Cost}(\sigma_{Status='active'}(NGO))$

    Select NGO_ID, Service_Cost from NGO where Status="active"

Embedded SQL Query (Advanced)

```
import mysql.connector
from tabulate import tabulate

mydb = mysql.connector.connect(
    host='localhost',
    user='root',
    password='',
    database='GERAS'
)

sql_cursor = mydb.cursor()

def pretty_print_results(results, column_names_list):
    print(tabulate(results, headers=column_names_list,
tablefmt='fancy_grid'))

def get_number_of_ngo_grouped_by_issue_types(sql_cursor):
    query = '''SELECT Issue_Type,
                    COUNT(NGO_ID) as Number_Of_NGO
                from NGO
                GROUP BY Issue_Type;'''

    try:
        sql_cursor.execute(query)
        results = sql_cursor.fetchall()
    except Exception as e:
        print(e)
        return

    return list(results), sql_cursor.column_names

def get_number_of_volunteers_per_ngo(sql_cursor):
    query = '''SELECT NGO.NGO_ID,
                    NGO.Name,
                    COUNT(DISTINCT(Volunteer_NGO.Volunteer_ID)) as
Number_Of_Volunteers
                FROM NGO
                    INNER JOIN Volunteer_NGO ON Volunteer_NGO.NGO_ID =
NGO.NGO_ID
                GROUP BY NGO.NGO_ID
```

```
                ORDER BY Number_Of_Volunteers DESC;'''

    try:
        sql_cursor.execute(query)
        results = sql_cursor.fetchall()
    except Exception as e:
        print(e)
        return

    return list(results), sql_cursor.column_names

def get_number_of_completed_and_pending_services(sql_cursor, hw_id):
    query =  '''SELECT HW_ID, CASE
                        WHEN Is_Completed = 1 THEN 'Completed'
                        ELSE 'Pending'
                    END as Status,
                    COUNT(DISTINCT(Service_ID)) as Number_Of_Services
                FROM Appoints_HW
                WHERE HW_ID = %s
                GROUP BY CASE
                        WHEN Is_Completed = 1 THEN 'Completed'
                        ELSE 'Pending'
                    END;'''

    params = [hw_id]

    try:
        sql_cursor.execute(query, params)
        results = sql_cursor.fetchall()
    except Exception as e:
        print(e)
        return

    return list(results), sql_cursor.column_names

def get_total_cost_per_healthworker(sql_cursor):
    query =  '''SELECT Appoints_HW.HW_ID,
                    CONCAT(
                        Healthcare_Worker.First_Name,
                        ' ',
                        Healthcare_Worker.Last_Name
                    ),
                    SUM(CAST(Appoints_HW.Service_Cost AS UNSIGNED)) as
```

```
Total_Cost
                FROM Appoints_HW
                    INNER JOIN Healthcare_Worker ON Appoints_HW.HW_ID =
Healthcare_Worker.HW_ID
                GROUP BY Appoints_HW.HW_ID;'''


    try:
        sql_cursor.execute(query)
        results = sql_cursor.fetchall()
    except Exception as e:
        print(e)
        return

    column_names = list(sql_cursor.column_names)
    column_names[1] = 'Full Name'
    return list(results), column_names

def get_ngo_details_with_cost_in_a_range(sql_cursor, ngo_issue_type,
filter_cost_range_start, filter_cost_range_end, ngo_activity_status):
    query =  '''SELECT *
                FROM NGO
                WHERE Issue_Type = %s
                    AND Service_Cost BETWEEN %s AND %s
                    AND C_Status = %s
                ORDER BY Service_Cost;'''

    params = [ngo_issue_type, filter_cost_range_start,
filter_cost_range_end, ngo_activity_status]

    try:
        sql_cursor.execute(query, params)
        results = list(sql_cursor.fetchall())
    except Exception as e:
        print(e)
        return

    for i in range(len(results)):
        results[i] = list(results[i])
        row_start = results[i][:4]
        address = ', '.join(results[i][4:9])
        row_end = results[i][9:]
        row_start.append(address)
```

```
        row_start.extend(row_end)
        results[i] = row_start

    column_names = list(sql_cursor.column_names)
    column_names_start = column_names[:4]
    column_names_start.append('address')
    column_names_start.extend(column_names[9:])


    return results, column_names_start

def get_top_servicing_ngos(sql_cursor, num_rows_to_return):
    query =  '''SELECT NGO.*,
                    Appoints_NGO.IS_Completed,
                    COUNT(DISTINCT(Appoints_NGO.Service_ID)) AS
Number_Of_Services
                FROM NGO
                    INNER JOIN Appoints_NGO ON NGO.NGO_ID =
Appoints_NGO.NGO_ID
                WHERE Appoints_NGO.Is_Completed = 1
                GROUP BY NGO.NGO_ID
                ORDER BY Number_Of_Services DESC
                LIMIT %s;'''

    params = [num_rows_to_return]

    try:
        sql_cursor.execute(query, params)
        results = list(sql_cursor.fetchall())
    except Exception as e:
        print(e)
        return

    for i in range(len(results)):
        results[i] = list(results[i])
        row_start = results[i][:4]
        address = ', '.join(results[i][4:9])
        row_end = results[i][9:]
        row_start.append(address)
        row_start.extend(row_end)
        results[i] = row_start

    column_names = list(sql_cursor.column_names)
```

```
    column_names_start = column_names[:4]
    column_names_start.append('address')
    column_names_start.extend(column_names[9:])

    return list(results), column_names_start

pretty_print_results(*get_number_of_ngo_grouped_by_issue_types(sql_cursor))
pretty_print_results(*get_number_of_volunteers_per_ngo(sql_cursor))
pretty_print_results(*get_number_of_completed_and_pending_services(sql_curs
or, 'HW40'))
pretty_print_results(*get_total_cost_per_healthworker(sql_cursor))
pretty_print_results(*get_ngo_details_with_cost_in_a_range(sql_cursor,
'Food', 100, 500, 'active'))
pretty_print_results(*get_top_servicing_ngos(sql_cursor, 10))

mydb.close()
```

Elderly Embedded SQL

```
#ELDERLY
def
register_an_elderly(sql_cursor,aadhar_no,first_name,last_name,dob,mobile,ho
use_no,street_name,city,state,zip,f_id):
    query =  ''' INSERT INTO
Elderly(aadhar_no,first_name,last_name,dob,mobile_no,house_no,street_name,c
ity,state,pincode,f_id,status) VALUES
(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,'Active');'''

    params =
[aadhar_no,first_name,last_name,dob,mobile,house_no,street_name,city,state,
zip,f_id]

    try:
        sql_cursor.execute(query, params)
        mydb.commit()
        print("Welcome to our app Geras. Thank you for registering.")
    except Exception as e:
        print(e)
        return


def request_service_by_elderly(sql_cursor,service_ID, details,
service_status, aadhar_no,start_date):
    query =  ''' INSERT INTO Service(service_ID, details, service_status,
aadhar_no, start_date) VALUES (%s,%s,%s,%s,%s);'''

    params = [service_ID, details, service_status,aadhar_no,start_date]

    try:
        sql_cursor.execute(query, params)
        mydb.commit()
        print("Your service request has been recorded")
    except Exception as e:
        print(e)
        return


def track_service_request_of_an_elderly(sql_cursor,aadhar_no, ser_ID):
    query =  '''SELECT * FROM SERVICE WHERE aadhar_no=%s and Service_ID =
```

```
%s;'''

    params = [aadhar_no,ser_ID]

    try:
        sql_cursor.execute(query, params)
        results = sql_cursor.fetchall()
    except Exception as e:
        print(e)
        return

    return list(results), sql_cursor.column_names

def get_presecription_history_of_an_elderly(sql_cursor,aadhar_no):
    query =  ''' SELECT * FROM Medicine_Update where aadhar_no=%s;'''
    params = [aadhar_no]

    try:
        sql_cursor.execute(query, params)
        results = sql_cursor.fetchall()
    except Exception as e:
        print(e)
        return

    return list(results), sql_cursor.column_names


def update_prescription_of_an_elderly(sql_cursor, aadhar_no, date,
prescription,doctor_name):
    query =  ''' INSERT INTO
Medicine_Update(Aadhar_No,Date,Prescription,Doctor_Name) VALUES
(%s,%s,%s,%s);'''

    params = [aadhar_no, date, prescription, doctor_name]

    try:
        sql_cursor.execute(query, params)
        mydb.commit()
        print("Updated prescription of"+ aadhar_no)
    except Exception as e:
        print(e)
        return
```

```
def get_service_history_of_an_elderly(sql_cursor,aadhar_no):
    query =  ''' SELECT * FROM Service where aadhar_no=%s;'''
    params = [aadhar_no]

    try:
        sql_cursor.execute(query, params)
        results = sql_cursor.fetchall()
    except Exception as e:
        print(e)
        return

    return list(results), sql_cursor.column_names



def get_all_completed_services_of_an_elderly(sql_cursor,aadhar_no):
    query =  ''' SELECT * FROM Service where aadhar_no=%s and
service_status='completed';'''
    params = [aadhar_no]

    try:
        sql_cursor.execute(query, params)
        results = sql_cursor.fetchall()
    except Exception as e:
        print(e)
        return

    return list(results), sql_cursor.column_names

def get_all_pending_services_of_an_elderly(sql_cursor,aadhar_no):
    query =  ''' SELECT * FROM Service where aadhar_no=%s and
(service_status='inprogress' or service_status='requested');'''
    params = [aadhar_no]

    try:
        sql_cursor.execute(query, params)
        results = sql_cursor.fetchall()
    except Exception as e:
        print(e)
        return

    return list(results), sql_cursor.column_names
```

```
def dergister_an_elderly(sql_cursor, aadhar_no):
    query =  ''' Update Elderly SET status='inactive' where aadhar_no=%s'''
    params = [aadhar_no]

    try:
        sql_cursor.execute(query, params)
        mydb.commit()
        print("We're sorry to see you go.")
    except Exception as e:
        print(e)
        return



#Elderly Queries
register_an_elderly(sql_cursor,'123456789102','samiksha','modi','19470816',
'9876543210','H1','Kadam Road','Delhi','Delhi','110007','FAM1')
request_service_by_elderly(sql_cursor,'SER51','I would like to request
someone to help me with my garden cleaning','Requested',
'237867112793','20200815')
pretty_print_results(*get_service_history_of_an_elderly(sql_cursor,'2378671
12793'))
pretty_print_results(*track_service_request_of_an_elderly(sql_cursor,'23786
7112793','SER51'))
pretty_print_results(*get_presecription_history_of_an_elderly(sql_cursor,'2
37867112793'))
update_prescription_of_an_elderly(sql_cursor,'237867112793','20200808','Cet
aphin','RK Aggarwal')
pretty_print_results(*get_presecription_history_of_an_elderly(sql_cursor,'2
37867112793'))
pretty_print_results(*get_service_history_of_an_elderly(sql_cursor,'4734887
49586'))
pretty_print_results(*get_all_completed_services_of_an_elderly(sql_cursor,'
473488749586'))
pretty_print_results(*get_all_pending_services_of_an_elderly(sql_cursor,'34
6907249932'))
dergister_an_elderly(sql_cursor,'123456789102')
```

Volunteer Embedded Queries

```
#VOLUNTEER QUERIES
def register_volunteer(sql_cursor,First_Name, Last_Name, Mobile_No,
Email_ID, House_No, Street_Name, City, State, Zip):
    query = '''INSERT INTO
Volunteer(First_Name,Last_Name,Mobile_No,Email_ID,House_No,Street_Name,City
,State,Zip) VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s);'''
    params =
[First_Name,Last_Name,Mobile_No,Email_ID,House_No,Street_Name,City,State,Zi
p]
    try:
        sql_cursor.execute(query,params)
        mydb.commit()
        print("Welcome to our app Geras. Thank you for registering.")
    except Exception as e:
        print(e)
        return

def reg_volunteer_with_ngo(sql_cursor,Volunteer_ID, NGO_ID):
    query = '''INSERT INTO Volunteer_NGO(Volunteer_ID, NGO_ID) VALUES(%s,
%s);'''
    params = [Volunteer_ID, NGO_ID]
    try:
        sql_cursor.execute(query,params)
        mydb.commit()
        print("You have been successfully registered")
    except Exception as e:
        print(e)
        return

def unregister_volunteer(sql_cursor, Volunteer_ID, NGO_ID):
    query = '''UPDATE Volunteer_NGO SET Available = 'No' WHERE Volunteer_ID
= %s AND NGO_ID = %s;'''
    params = [Volunteer_ID,NGO_ID]
    try:
        sql_cursor.execute(query,params)
        mydb.commit()
        print("You are no longer a volunteer for NGO with ID",NGO_ID)
    except Exception as e:
        print(e)
        return
```

```
def view_volunteer_work(sql_cursor, Volunteer_ID):
    query = '''SELECT a.Service_ID, s.Details, s.Requested_For,
s.Start_Date, s.End_Date
    FROM Appoints_NGO AS a INNER JOIN Service AS s
    WHERE a.Volunteer_ID = %s
    AND a.Is_Completed = 0;'''
    params = [Volunteer_ID]
    try:
        sql_cursor.execute(query,params)
        results = sql_cursor.fetchall()
    except Exception as e:
        print(e)
        return
    return list(results), sql_cursor.column_names

def report_completion_by_volunteer(sql_cursor,Volunteer_ID, Service_ID,
End_Date):
    query = '''UPDATE Appoints_NGO SET Is_Completed = 1 WHERE Volunteer_ID
= %s;'''
    params = [Volunteer_ID]
    try:
        sql_cursor.execute(query,params)
        mydb.commit()
        print("Service with ID "+Service_ID+" has been marked as
completed")
    except Exception as e:
        print(e)
        return

def update_volunteer_availability(sql_cursor,Volunteer_ID):
    query = '''UPDATE Volunteer_NGO SET Available = 0 WHERE Volunteer_ID =
%s;'''
    params = [Volunteer_ID]
    try:
        sql_cursor.execute(query,params)
        mydb.commit()
        print("Your status have been successfully updated")
    except Exception as e:
        print(e)
        return
```

Family Embedded Queries

```
#Family queries

# 1. register details

def
register_as_family(sql_cursor,F_ID,First_name,Last_name,Mobile_No,Email_ID,
House_No,Street_Name,City,State,Pincode,C_Status):
    query =  ''' INSERT INTO
Family(F_ID,First_name,Last_name,Mobile_No,Email_ID,House_No,Street_Name,Ci
ty,State,Pincode,C_Status) VALUES
(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s,'Active');'''
    params =
[F_ID,First_name,Last_name,Mobile_No,Email_ID,House_No,Street_Name,City,Sta
te,Pincode,C_Status]
    try:
        sql_cursor.execute(query, params)
        mydb.commit()
        print("Welcome to our app Geras. Thank you for registering as
family.")
    except Exception as e:
        print(e)
        return


# 2. View current healthcare workers associated with elderly

def healthcare_worker_details(sql_cursor,F_ID):
    query =  '''SELECT * FROM Healthcare_Worker,Appoints_HW,Service,Elderly
WHERE Healthcare_Worker.HW_ID = Appoints_HW.HW_ID AND
Appoints_HW.Service_ID = Service.Service_ID AND (Service.Service_Status <>
'Completed' OR Service.Service_Status <> 'Cancelled') AND Service.Aadhar_No
= Elderly.Aadhar_No AND Elderly.F_ID = %s;'''
    params = [F_ID]
    try:
        sql_cursor.execute(query, params)
        results = sql_cursor.fetchall()
    except Exception as e:
        print(e)
        return
    return list(results), sql_cursor.column_names
```

```
# 3. View Medicine record

def view_medicine_record(sql_cursor, F_ID):
    query =  ''' SELECT Prescription,Doctor_Name FROM
Medicine_Update,Elderly WHERE Medicine_Update.Aadhar_No = Elderly.Aadhar_no
AND Elderly.F_ID = %s ;'''
    params = [F_ID]
    try:
        sql_cursor.execute(query, params)
        results = sql_cursor.fetchall()
    except Exception as e:
        print(e)
        return
    return list(results), sql_cursor.column_names


# 4. Request Service for Elderly

def
request_service_for_elderly(sql_cursor,F_ID,service_ID,details,service_stat
us,start_date):
    query =  ''' INSERT INTO Service(service_ID, details, service_status,
aadhar_no AS SELECT aadhar_no FROM Elderly WHERE F_ID = Elderly.F_ID ,
start_date) VALUES (%s,%s,%s,%s,%s) ;'''
    params = [F_ID,Service_ID,details,service_status,start_date]
    try:
        sql_cursor.execute(query, params)
        mydb.commit()
        print("Service requested successfully for elderly.")
    except Exception as e:
        print(e)
        return

# 5. Track Service Request

def track_elderly_service_request(sql_cursor,F_ID):
    query =  ''' SELECT * FROM Service,Elderly WHERE
Services.aadhar_no=Elderly.aadhar_no AND Eldery.F_ID=%s;'''
    params = [F_ID]
    try:
        sql_cursor.execute(query, params)
```

```
        results = sql_cursor.fetchall()
    except Exception as e:
        print(e)
        return
    return list(results), sql_cursor.column_names
```

Healthcare Worker Embedded Queries

```
# 2. Access prescription history of an Elderly

def presciption_elderly(sql_cursor,HW_ID):
    query =  '''SELECT * FROM Medicine_Update,Elderly,Appoints_HW,Service
WHERE Medicine_Update.Aadhar_No = Elderly.Adhar_No AND Elderly.Aadhar_No =
Service.Aadhar_No AND Service.Service_ID = Appoints_HW.Service_ID AND
Appoints_HW.HW_ID = %s;'''
    params = [HW_ID]
    try:
        sql_cursor.execute(query, params)
        results = sql_cursor.fetchall()
    except Exception as e:
        print(e)
        return
    return list(results), sql_cursor.column_names

# 3. Mark service as Completed

def report_completion_by_volunteer(Service_ID):
    query = '''UPDATE Appoints_HW SET Is_Completed = 'Yes' WHERE Service_ID
= %s;'''
    params = [Service_ID]
    try:
        sql_cursor.execute(query,params)
        mydb.commit()
        print("Service with ID "+Service_ID+" has been marked as
completed")
    except Exception as e:
        print(e)
        return


# 4. Contact Elderly's Family in case of emergency

def contact_family(Service_ID):
    query =  '''SELECT * FROM Family,Elderly,Service WHERE Family.F_ID =
Elderly.F_ID AND Elderly.Aadhar_No = Service.Aadhar_No AND
Service.Service_ID = %s;'''
    params = [Service_ID]
```

```
    try:
        sql_cursor.execute(query, params)
        results = sql_cursor.fetchall()
    except Exception as e:
        print(e)
        return
    return list(results), sql_cursor.column_names
```

SOS Feature :

```
import requests
from googleplaces import GooglePlaces, types, lang
import json

r= requests.get('https://www.geojs.io')

# print(r)

ip_request=requests.get('https://get.geojs.io/v1/ip.json')

# print(ip_request)

ip_address=ip_request.json()['ip']

# ip_address

url='https://get.geojs.io/v1/ip/geo/' + ip_address + '.json'

geo_request=requests.get(url)

latitude=geo_request.json()['latitude']

longitude=geo_request.json()['longitude']

lat=float(latitude)
lng=float(longitude)

#add api key here
API_KEY = 'AIzaSyBt6D3IPoGk5uHGEIPxGM-DavezZveO2CY'

google_places = GooglePlaces(API_KEY)

query_result = google_places.nearby_search(
        lat_lng ={'lat': lat, 'lng': lng},
        radius = 5000,
        # types =[types.TYPE_HOSPITAL] or
        # [types.TYPE_CAFE] or [type.TYPE_BAR]
        # or [type.TYPE_CASINO])
        types =[types.TYPE_HOSPITAL])
```

```
# If any attributions related
# with search results print them
if query_result.has_attributions:
    print (query_result.html_attributions)


# Iterate over the search results
for place in query_result.places:
    # print(type(place))
    # place.get_details()
    print (place.name)
    print("Latitude", place.geo_location['lat'])
    print("Longitude", place.geo_location['lng'])
    print()
```

Contribution

Assigned on	Task	Assigned to	Done by
20 Jan 21	Decide on a project idea	All	All
26 Jan 21	Identify and write the roles and queries for each Stakeholder		
	The Elderly	Samiksha	Samiksha
	Volunteer	Kabir	Kabir
	NGO	Tushar	Tushar
	Government	Kunal	Samiksha
	Families	Aman	Aman
27 Jan 21	Write Problem	Samiksha	Samiksha
	Write Idea	Aman	Aman
	Write Description	Kabir	Kabir
30 Jan 21	Decide on Project Title, App name	All	All
31 Jan 21	Write the data entities, its attributes and relation		
	The Elderly	Samiksha	Samiksha
	Volunteer	Kabir	Kabir
	NGO	Tushar	Tushar
	Government	Kunal	Kunal
	Families	Aman	Aman
4 Feb 21	Add Doctor /Nurses as Stakeholder	Tushar	Tushar
13 Feb 21	Defining entities and relationship between them	Samiksha, Kabir, Aman	Samiksha, Kabir, Aman
15 Feb 21	E-R Diagram	Samiksha, Kabir, Aman, Tushar	Samiksha, Kabir, Aman, Tushar

19 Feb 21	Schema	Tushar, Aman, Kunal, Samiksha	Tushar, Aman, Kunal, Samiksha
20 Feb 21	Create SQL Tables, Foreign Key Constraints	Kunal	Kunal
21 Feb 21	Generate data	Samiksha, Aman, Kunal, Tushar	Samiksha, Aman, Kunal, Tushar
23 Mar	Finalise table data	Samiksha	Samiksha
	Update Database	Kunal	Kunal
	Write at least 10 queries involving various relational algebraic operations supporting the application features involving database access and manipulation.	Samiksha, Kabir	Samiksha, Kabir
	Identify the attribute(s) to create Index tables required for your queries.	Kunal	
	Write at least 4 embedded SQL queries (PL/SQL), advanced aggregation functions, etc supporting your application features	Tushar	Tushar
2 April	Write the embedded SQL queries for each stakeholder		
	Elderly	Samiksha	Samiksha
	Volunteer	Kabir	Kabir
	Government	Kunal	Government
	NGO	Tushar	Tushar
	Family	Aman	Aman
	Healthcare Worker	Aman	Aman
14 April	SOS feature	Aman	Aman
17 April	Write python program for the app	All	All

Meetings

Meeting on	Notes	Absent
23 Jan 2021	Worked on Wedding Planner	
24 Jan 2021	Worked on Pharmacy, FoodShare, Project Geras	
25 Jan 2021	TA meeting, voted on a project - Project Geras	
31 Jan 2021	Assigned tasks for data entities, attributes, type, relation between entities	Kunal
2 Feb 21	Worked the feedback Sir gave into our project	Kunal, Tushar
13 Feb 21	Defining entities and relationship between them	Kunal
15 Feb 21	E-R diagram	Kunal
16 Feb 21	E-R diagram multiplicity	Kunal
19 Feb 21	Assign tasks for mid sem deadline	
23 Mar 21	Assign tasks for queries and finalising database	Aman, Tushar, Kabir
2 April	Assign work regarding embedded sql queries for features	Aman
14 April	Assign leftover over work and catch up on everyone's progress	
17 April	Merge individually done work	