Databases – 2020-21

Group: AL

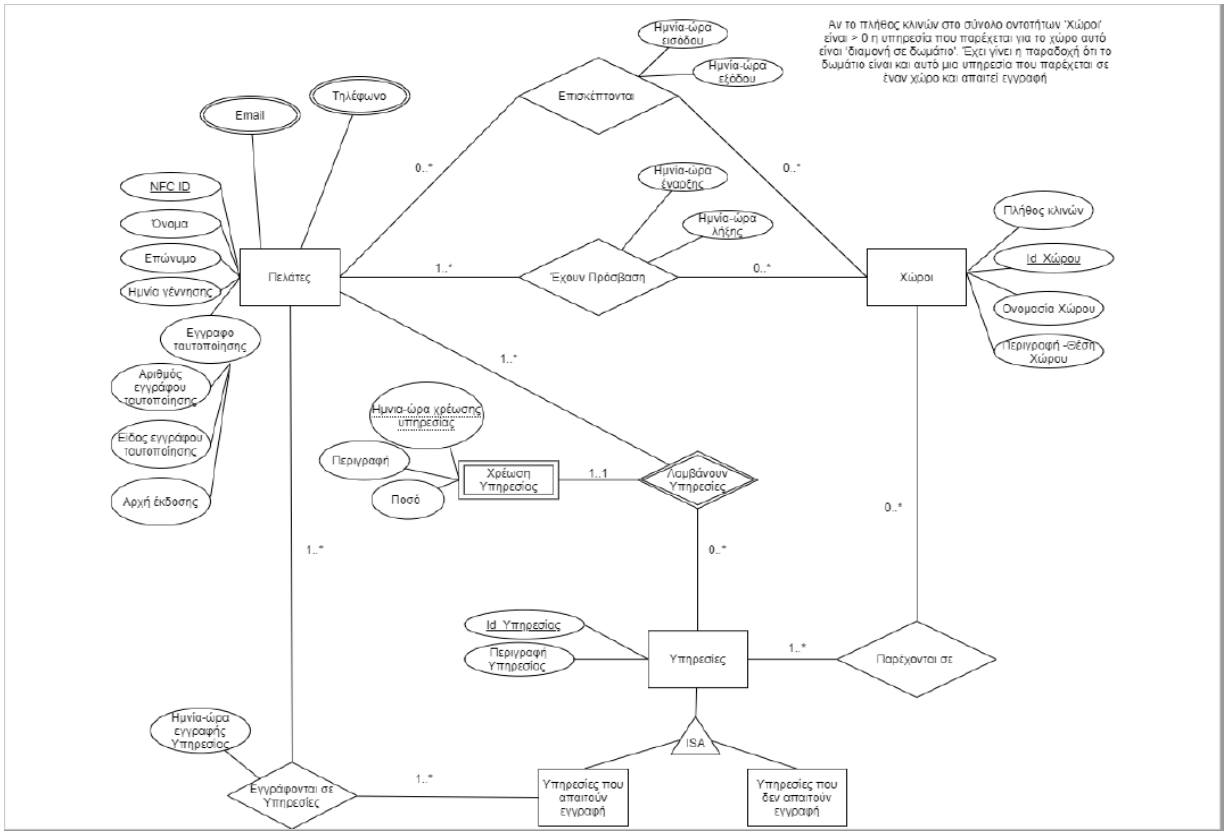
Member: *Βαλουξής Σπυρίδων (03117003)*

Member: *Γεωργούτσος Αθανάσιος (03117151)*

Date: *13/6/21*

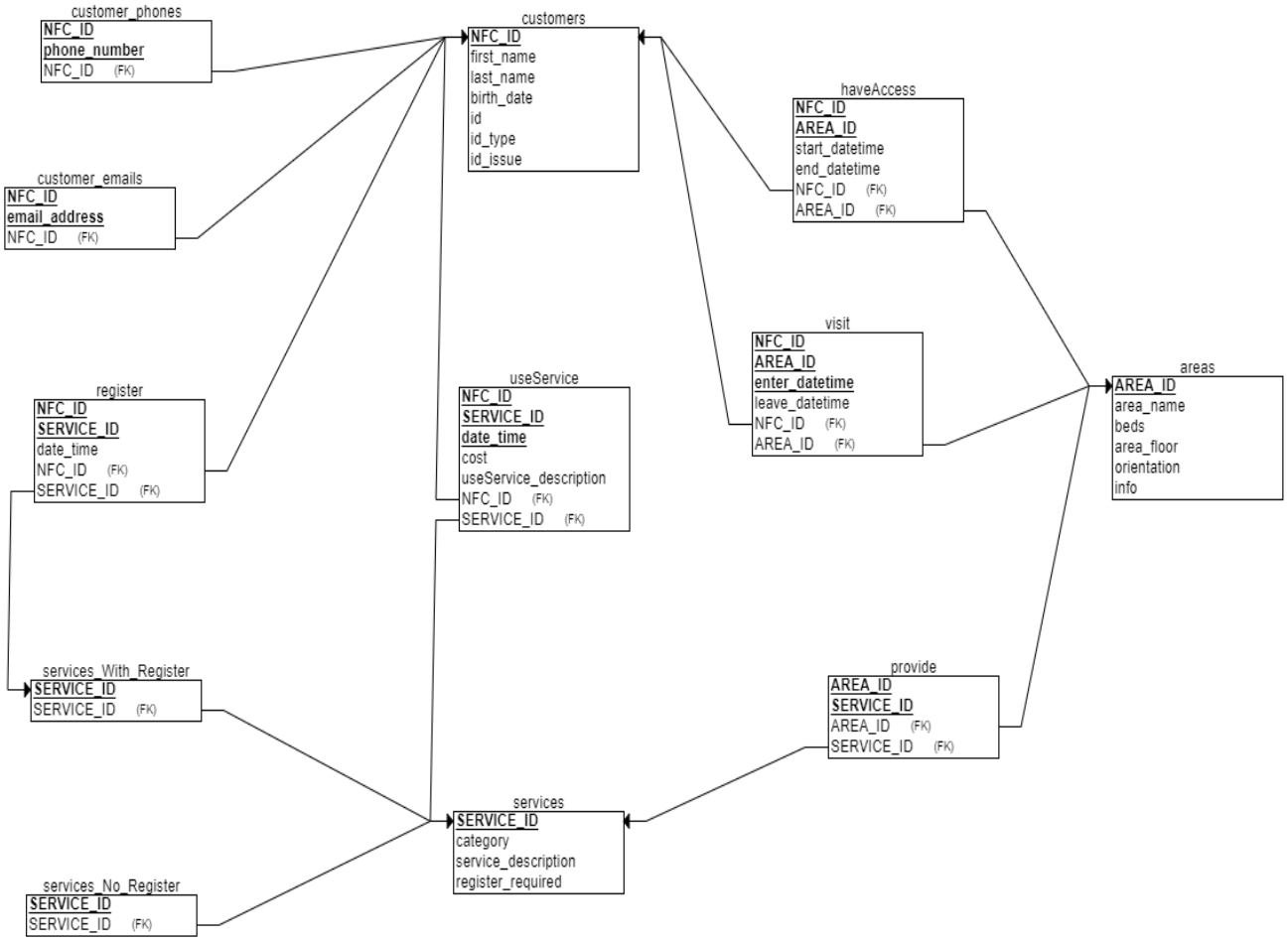
Github Repository Link: https://github.com/ThanosBb3/Database_Project

Below follows the ER diagram (in Greek) that was given to us and on which we based our database structure.



As you will see in the next Relational Diagram, we maintained the structure of the ER, even though we could use less tables in our application. The reason is that, in any expansion of the app, these “not-so-useful” tables may prove to be crucial in the development of new functions and services.

1. Relational Diagram



a. Constraints

NOT NULL

In MySQL NOT NULL constraint allows to specify that a column can not contain any NULL value.

PRIMARY KEY

A PRIMARY KEY constraint for a table enforces the table to accept unique data for a specific column and this constraint creates a unique index for accessing the table faster.

FOREIGN KEY

A FOREIGN KEY in MySQL creates a link between two tables by one specific column of both tables. The specified column in one table must be a PRIMARY KEY and referred by the column of another table known as FOREIGN KEY.

CHECK

A CHECK constraint controls the values in the associated column. The CHECK constraint determines whether the value is valid or not from a logical expression.

For example we present the following two tables that use all of the above constraints:

```
-- Create the table Phones
CREATE TABLE customer_phones
(
 NFC_ID INT NOT NULL,
 phone_number BIGINT NOT NULL,
 CONSTRAINT PKcustomer_phones PRIMARY KEY(NFC_ID, phone_number),
 CONSTRAINT FKcustomer_phones FOREIGN KEY (NFC_ID) REFERENCES customers
(NFC_ID) ON UPDATE CASCADE ON DELETE CASCADE
);

-- Create the table Areas
CREATE TABLE areas
(
 AREA_ID INT NOT NULL AUTO_INCREMENT,
 area_name VARCHAR(40) NOT NULL,
 beds TINYINT NOT NULL,
 area_floor TINYINT,
 orientation VARCHAR(20),
 info VARCHAR(100),
 CONSTRAINT PKareas PRIMARY KEY(AREA_ID),
```

```
 CONSTRAINT CHKareas CHECK (orientation IN ('N', 'W', 'E', 'S', 'NE', '
NW', 'SE', 'SW')),
 CONSTRAINT CHK2areas CHECK (area_floor < 6)
);
```

b. Indexes

Considering that MySQL indexes automatically primary and foreign keys, we chose the following indexes because of their frequent use in our SQL queries.

```
USE HotelDB;
CREATE INDEX visit_time ON visit(enter_datetime, leave_datetime)
CREATE INDEX use_cost ON useService(cost)
CREATE INDEX use_time ON useService(date_time)
```

For example the first index is obviously useful in the following query:

```
SELECT A.NFC_ID, B.last_name, B.first_name, C.category, A.cost, D.enter
_datetime, D.leave_datetime
            FROM useService A, customers B, services C, visit D
            WHERE
                A.date_time >= '{}'
                AND A.date_time <= '{}'
                AND A.cost >= {}
                AND A.cost <= {}
                AND A.SERVICE_ID <> 1
                AND C.SERVICE_ID = A.SERVICE_ID
                AND A.NFC_ID = B.NFC_ID
                AND D.NFC_ID = A.NFC_ID
                AND D.enter_datetime <= A.date_time
                AND D.leave_datetime >= A.date_time
            GROUP BY A.date_time
            ORDER BY A.date_time DESC;
```

c. System and Programming Languages Used

Requirements

```
mysql 8.0.25
flask 1.1.2
mysql_connector 2.2.9
numpy 1.20.1
pandas 1.2.3
```

We used MySQL for our Database and Python Flask for the app. We also used mysql_connector for their combination. Numpy and Pandas were also used during the Database population stage.

Regarding the Programming Languages we used SQL for the Database operations, Python for the backend of the app and basic HTML and CSS for the frontend. We also use the special Flask language called Jinja for the frontend of our app, which allows for base and extended templates.

d. Installation

Step 1: At first, initialize a mysql database at a localhost.

Step 2: Then, run the folowing command in terminal, using your credentials in order to connect in mysql host:

```
mysql --user="your user name" --password="your password" --
host=localhost
```

Step 3: Run the following inside mysql command prompt, strictly at this order:

- Source "path of the HotelDB.sql file"
- Source "path of the indexes.sql file"
- Source "path of the customers_view.sql file"
- Source "path of the services_view.sql file"

Step 4: Back at the Terminal run:

```
git clone https://github.com/ThanosBb3/Database_Project
```

Step 5: Add your Database credentials in the following files:

- app\backend\home.py
- app\backend\register.py
- app__init__.py
- db_initialization\connection.py

Step 6: Run

```
python main_db.py
```

to add the data on your Database.

Step 7: Now that everything is set up run:

```
python main_app.py
```

and open your browser on http://localhost:8765/ to see the app.

2. SQL Code

HotelDB.sql

```
CREATE DATABASE `HotelDB`;
ALTER DATABASE HotelDB CHARACTER SET utf8 COLLATE utf8_bin;
USE `HotelDB`;


-- Create the table Customers
CREATE TABLE customers
(
 NFC_ID INT NOT NULL AUTO_INCREMENT,
 first_name VARCHAR(100) NOT NULL,
 last_name VARCHAR(100) NOT NULL,
 birth_date date NOT NULL,
 id BIGINT NOT NULL,
 id_type VARCHAR(50) NOT NULL,
 id_issue VARCHAR(50) NOT NULL,
 CONSTRAINT PKcustomers PRIMARY KEY (NFC_ID),
 CONSTRAINT CHKcustomers CHECK (id_type IN ('Passport', 'Identity'))
);

-- Create the table Emails
CREATE TABLE customer_emails
(
 NFC_ID INT NOT NULL,
 email_address VARCHAR(80) NOT NULL,
 CONSTRAINT PKcustomer_emails PRIMARY KEY(NFC_ID, email_address),
 CONSTRAINT FKcustomer_emails FOREIGN KEY (NFC_ID) REFERENCES customers
(NFC_ID) ON UPDATE CASCADE ON DELETE CASCADE
);


-- Create the table Phones
CREATE TABLE customer_phones
(
 NFC_ID INT NOT NULL,
 phone_number BIGINT NOT NULL,
 CONSTRAINT PKcustomer_phones PRIMARY KEY(NFC_ID, phone_number),
 CONSTRAINT FKcustomer_phones FOREIGN KEY (NFC_ID) REFERENCES customers
(NFC_ID) ON UPDATE CASCADE ON DELETE CASCADE
);

-- Create the table Areas
CREATE TABLE areas
(
 AREA_ID INT NOT NULL AUTO_INCREMENT,
```

```
 area_name VARCHAR(40) NOT NULL,
 beds TINYINT NOT NULL,
 area_floor TINYINT,
 orientation VARCHAR(20),
 info VARCHAR(100),
 CONSTRAINT PKareas PRIMARY KEY(AREA_ID),
 CONSTRAINT CHKareas CHECK (orientation IN ('N', 'W', 'E', 'S', 'NE', '
NW', 'SE', 'SW')),
 CONSTRAINT CHK2areas CHECK (area_floor < 6)
);


-- Create the table Services
CREATE TABLE services
(
 SERVICE_ID INT NOT NULL AUTO_INCREMENT,
 category VARCHAR(30) NOT NULL,
 service_description VARCHAR(100) NOT NULL,
 register_required BOOLEAN NOT NULL,
 CONSTRAINT PKservices PRIMARY KEY(SERVICE_ID)
);

-- Create the table WithRegister
CREATE TABLE services_With_Register
(
 SERVICE_ID INT NOT NULL,
 CONSTRAINT PKservices_With_Register PRIMARY KEY(SERVICE_ID),
 CONSTRAINT FKservices_With_Register FOREIGN KEY (SERVICE_ID) REFERENCE
S services(SERVICE_ID) ON UPDATE CASCADE ON DELETE CASCADE
);


-- Create the table NoRegister
CREATE TABLE services_No_Register
(
 SERVICE_ID INT NOT NULL,
 CONSTRAINT PKservices_No_Register PRIMARY KEY(SERVICE_ID),
 CONSTRAINT FKservices_No_Register FOREIGN KEY (SERVICE_ID) REFERENCES
services(SERVICE_ID) ON UPDATE CASCADE ON DELETE CASCADE
);

-- Create the table Register
CREATE TABLE register
(
 NFC_ID INT NOT NULL,
 SERVICE_ID INT NOT NULL,
 date_time DATETIME NOT NULL,
 CONSTRAINT PKregister PRIMARY KEY (NFC_ID, SERVICE_ID),
```

```
 CONSTRAINT FK1customers FOREIGN KEY (NFC_ID) REFERENCES customers(NFC_
ID) ON UPDATE CASCADE ON DELETE CASCADE,
 CONSTRAINT FK2service FOREIGN KEY (SERVICE_ID) REFERENCES services_Wit
h_Register(SERVICE_ID) ON UPDATE CASCADE ON DELETE CASCADE
);

-- Create the table HasAccess
CREATE TABLE haveAccess
(
  NFC_ID INT NOT NULL,
  AREA_ID INT NOT NULL,
  start_datetime DATETIME NOT NULL,
  end_datetime DATETIME NOT NULL,
  CONSTRAINT PKhaveAccess PRIMARY KEY (NFC_ID, AREA_ID),
  CONSTRAINT FK2customers FOREIGN KEY (NFC_ID) REFERENCES customers(NFC
_ID) ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT FK2area FOREIGN KEY (AREA_ID) REFERENCES areas(AREA_ID) ON
 UPDATE CASCADE ON DELETE CASCADE
);

-- Create the table Visit
CREATE TABLE visit
(
  NFC_ID INT NOT NULL,
  AREA_ID INT NOT NULL,
  enter_datetime DATETIME NOT NULL,
  leave_datetime DATETIME NOT NULL,
  CONSTRAINT PKvisit PRIMARY KEY (NFC_ID, AREA_ID, enter_datetime),
  CONSTRAINT FK3customers FOREIGN KEY (NFC_ID) REFERENCES customers(NFC
_ID) ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT FK3area FOREIGN KEY (AREA_ID) REFERENCES areas(AREA_ID) ON
 UPDATE CASCADE ON DELETE CASCADE
);

-- Create the table UseService
CREATE TABLE useService
(
 NFC_ID INT NOT NULL,
 SERVICE_ID INT NOT NULL,
 date_time DATETIME NOT NULL,
 cost FLOAT NOT NULL,
 useService_description VARCHAR(200),
 CONSTRAINT PKuseService PRIMARY KEY (NFC_ID, SERVICE_ID, date_time),
 CONSTRAINT FK4customers FOREIGN KEY (NFC_ID) REFERENCES customers(NFC_
ID) ON UPDATE CASCADE ON DELETE CASCADE,
 CONSTRAINT FK3service FOREIGN KEY (SERVICE_ID) REFERENCES services(SER
VICE_ID) ON UPDATE CASCADE ON DELETE CASCADE
);
```

```
-- Create the table Provide
CREATE TABLE provide
(
 AREA_ID INT NOT NULL,
 SERVICE_ID INT NOT NULL,
 CONSTRAINT PKprovide PRIMARY KEY (AREA_ID, SERVICE_ID),
 CONSTRAINT FK1area_provide FOREIGN KEY (AREA_ID) REFERENCES areas(AREA
_ID) ON UPDATE CASCADE ON DELETE CASCADE,
 CONSTRAINT FK2service_provide FOREIGN KEY (SERVICE_ID) REFERENCES serv
ices(SERVICE_ID) ON UPDATE CASCADE ON DELETE CASCADE
);
```

Indexes.sql

```
USE HotelDB;
CREATE INDEX visit_time ON visit(enter_datetime, leave_datetime)
CREATE INDEX use_cost ON useService(cost)
CREATE INDEX use_time ON useService(date_time)
```

Customers_view.sql

```
USE HotelDB;

CREATE VIEW customers_info
AS
SELECT A.NFC_ID, A.last_name, A.first_name, A.birth_date, A.id, A.id_ty
pe, A.id_issue, GROUP_CONCAT(distinct customer_phones.phone_number), GR
OUP_CONCAT(distinct customer_emails.email_address)
FROM customers A, customer_phones, customer_emails
WHERE A.NFC_ID = customer_emails.NFC_ID
    AND A.NFC_ID = customer_phones.NFC_ID
GROUP BY A.NFC_ID
ORDER BY A.NFC_ID DESC
```

Services_view.sql

```
USE HotelDB;

CREATE VIEW services_sales
AS
SELECT A.SERVICE_ID, A.category, A.service_description, ROUND(SUM(B.cos
t),2)
FROM services A, useService B
WHERE A.SERVICE_ID = B.SERVICE_ID
GROUP BY A.SERVICE_ID
ORDER BY A.SERVICE_ID ASC
```

Delete_services_trigger.sql

```
use HotelDB;
DELIMITER //
DROP TRIGGER IF EXISTS delete_services
//
CREATE TRIGGER delete_services AFTER DELETE
ON services
FOR EACH ROW
BEGIN
   IF (not OLD.register_required) THEN
   DELETE FROM services_No_Register
      WHERE (SERVICE_ID=OLD.SERVICE_ID);

    ELSE
   DELETE FROM services_With_Register
      WHERE (SERVICE_ID=OLD.SERVICE_ID);
END IF;
END;//
DELIMITER ;
```

Insert_services_trigger.sql

```
use HotelDB;
DELIMITER //
DROP TRIGGER IF EXISTS insert_services
//
CREATE TRIGGER insert_services AFTER INSERT
ON services
FOR EACH ROW
BEGIN
   IF (not NEW.register_required) THEN
   INSERT INTO services_No_Register
      (SERVICE_ID)
   VALUES
      (NEW.SERVICE_ID);

    ELSE
   INSERT INTO services_With_Register
      (SERVICE_ID)
   VALUES
      (NEW.SERVICE_ID);
END IF;
END;//
DELIMITER ;
```

Now we present every SQL query made in each of the following files.
Note that the following files are written in python and here we present only the SQL queries made on those files.

addAccess.py

```
INSERT INTO haveAccess (NFC_ID,AREA_ID,start_datetime,end_datetime)
                            VALUES ({},{},'{}','{}')
```

```
INSERT INTO haveAccess (NFC_ID,AREA_ID,start_datetime,end_datetime)
                            VALUES ({},(SELECT AREA_ID FROM areas WHERE
 area_floor=(SELECT area_floor FROM areas WHERE AREA_ID={}) AND
                            orientation=(SELECT orientation FROM areas
WHERE AREA_ID={}) AND beds=0),'{}','{}')
```

addAreas.py

```
INSERT INTO areas (area_name,beds,area_floor,orientation,info)
                        VALUES ('{}',{},{},'{}','{}')
```

```
INSERT INTO areas (area_name,beds,area_floor)
                        VALUES ('{}',{},{})
```

addCustomers.py

```
INSERT INTO customers (first_name,last_name,birth_date,id,id_type,id_is
sue)
                        VALUES ('{}','{}','{}',{},'{}','{}')
```

```
INSERT INTO customer_phones (NFC_ID,phone_number)
                        VALUES ({},{})
```

```
INSERT INTO customer_emails (NFC_ID,email_address)
                        VALUES ({},'{}')
```

addProvide.py

```
INSERT INTO provide (AREA_ID, SERVICE_ID)
                        VALUES ({},{})
```

addRegister.py

```
INSERT INTO register (NFC_ID,SERVICE_ID,date_time)
```

```
                        VALUES ({},{},'{}')
INSERT INTO useService (NFC_ID,SERVICE_ID,date_time,cost)
                        VALUES ({},{},'{}',{})
```

addServices.py

```
INSERT INTO services (category,service_description,register_required)
                    VALUES ('{}','{}',{})
```

addVisitAndUse.py

```
INSERT INTO visit (NFC_ID,AREA_ID,enter_datetime,leave_datetime)
                                        VALUES ({},(SELECT AREA_ID FROM
 haveAccess WHERE (NFC_ID={} AND AREA_ID>400 AND AREA_ID<421)),'{}','{}
')
```

```
INSERT INTO useService (NFC_ID,SERVICE_ID,date_time,cost)
                                        VALUES ({},{},'{}',{})
```

```
SELECT AREA_ID FROM haveAccess WHERE NFC_ID={} AND (AREA_ID<401 OR AREA
_ID>420)
```

Covid.py

```
SELECT A.AREA_ID, B.area_name, A.enter_datetime, A.leave_datetime
                    FROM visit A, areas B
                    WHERE A.AREA_ID=B.AREA_ID AND NFC_ID={}
                    ORDER BY leave_datetime DESC;
```

```
SELECT A.NFC_ID, A.last_name, A.first_name, GROUP_CONCAT(distinct custo
mer_phones.phone_number), GROUP_CONCAT(distinct customer_emails.email_a
ddress)
                    FROM customers A, customer_phones, customer_emails,
 visit B, visit C
                    WHERE A.NFC_ID<>{} AND
                        A.NFC_ID=customer_phones.NFC_ID AND A.NFC_ID=cu
stomer_emails.NFC_ID AND
                        B.NFC_ID={} AND C.NFC_ID=A.NFC_ID AND B.AREA_ID
=C.AREA_ID AND
                        C.leave_datetime > B.enter_datetime AND C.enter
_datetime < CONVERT(ADDTIME(STR_TO_DATE(B.leave_datetime,'%Y-%m-
%d %H:%i:%s'),'1:0:0'), CHAR)
                    GROUP BY A.NFC_ID
                    ORDER BY A.NFC_ID DESC
```

Home.py

```
INSERT INTO customers(last_name, first_name, birth_date, id, id_type, i
d_issue)
                            VALUES ('{}', '{}', '{}', {}, '{}', '{}');
```

```
SELECT MAX(NFC_ID) FROM customers;
```

```
INSERT INTO customer_phones(NFC_ID, phone_number)
                                VALUES ((SELECT MAX(NFC_ID) FROM custom
ers), {});
```

```
INSERT INTO customer_emails(NFC_ID, email_address)
                                VALUES ((SELECT MAX(NFC_ID) FROM custom
ers), '{}');
```

```
DELETE FROM customer_phones WHERE NFC_ID={};
```

```
DELETE FROM customer_emails WHERE NFC_ID={};
```

```
INSERT INTO customer_phones(NFC_ID, phone_number)
                                VALUES ({}, {});
```

```
INSERT INTO customer_emails(NFC_ID, email_address)
                                VALUES ({}, '{}');
```

```
DELETE FROM customers WHERE NFC_ID={};
```

Records.py

```
SELECT A.NFC_ID, B.last_name, B.first_name, C.category, A.cost, D.enter
_datetime, D.leave_datetime
            FROM useService A, customers B, services C, visit D
            WHERE
                A.date_time >= '{}'
                AND A.date_time <= '{}'
                AND A.cost >= {}
                AND A.cost <= {}
                AND A.SERVICE_ID <> 1
                AND C.SERVICE_ID = A.SERVICE_ID
```

```
                AND A.NFC_ID = B.NFC_ID
                AND D.NFC_ID = A.NFC_ID
                AND D.enter_datetime <= A.date_time
                AND D.leave_datetime >= A.date_time
            GROUP BY A.date_time
            ORDER BY A.date_time DESC;
```

```
SELECT A.NFC_ID, B.last_name, B.first_name, C.category, A.cost, D.enter
_datetime, D.leave_datetime
            FROM useService A, customers B, services C, visit D
            WHERE
                A.date_time >= '{}'
                AND A.date_time <= '{}'
                AND A.cost >= {}
                AND A.cost <= {}
                AND A.SERVICE_ID = (SELECT SERVICE_ID
                                    FROM services
                                    WHERE category='{}')
                AND C.SERVICE_ID = A.SERVICE_ID
                AND A.NFC_ID = B.NFC_ID
                AND D.NFC_ID = A.NFC_ID
                AND D.enter_datetime <= A.date_time
                AND D.leave_datetime >= A.date_time
            ORDER BY D.leave_datetime DESC;
```

Register.py

```
SELECT C.area_name FROM areas C WHERE  C.AREA_ID<401 AND ((SELECT COUNT
(B.NFC_ID) FROM haveAccess B WHERE B.AREA_ID=C.AREA_ID AND B.end_dateti
me>'{}') < C.beds)
```

```
SELECT M.area_name FROM areas M WHERE M.AREA_ID<401 AND M.area_name NOT
 IN (SELECT C.area_name FROM areas C WHERE  C.AREA_ID<401 AND ((SELECT
COUNT(B.NFC_ID) FROM haveAccess B WHERE B.AREA_ID=C.AREA_ID AND B.end_d
atetime>'{}') < C.beds))
```

```
SELECT MAX(NFC_ID) FROM customers;
```

```
SELECT EXISTS (SELECT * FROM register WHERE NFC_ID={} AND SERVICE_ID=(S
SELECT SERVICE_ID FROM services WHERE category='{}'));
```

```
INSERT INTO register(NFC_ID, SERVICE_ID, date_time)
                                    VALUES ({}, 1, '{}');
```

```
INSERT INTO haveAccess(NFC_ID, AREA_ID, start_datetime, end_datetime)
                                    VALUES ({}, (SELECT AREA_ID FROM ar
eas WHERE area_name='{}'), '{}', '{}');
```

```
INSERT INTO useService(NFC_ID, SERVICE_ID, date_time, cost)
                                    VALUES ({}, 1, '{}', {});
```

```
INSERT INTO haveAccess (NFC_ID,AREA_ID,start_datetime,end_datetime)
                                VALUES ({},(SELECT AREA_ID FROM areas W
HERE area_floor=(SELECT area_floor FROM areas WHERE AREA_ID=(SELECT ARE
A_ID FROM areas WHERE area_name='{}')) AND
                                orientation=(SELECT orientation FROM ar
eas WHERE AREA_ID=(SELECT AREA_ID FROM areas WHERE area_name='{}')) AND
 beds=0),'{}','{}')
```

```
SELECT C.area_name FROM areas C WHERE  C.AREA_ID<401 AND ((SELECT COUNT
(B.NFC_ID) FROM haveAccess B WHERE B.AREA_ID=C.AREA_ID AND B.end_dateti
me>'{}') < C.beds)
```

```
SELECT M.area_name FROM areas M WHERE M.AREA_ID<401 AND M.area_name NOT
 IN (SELECT C.area_name FROM areas C WHERE  C.AREA_ID<401 AND ((SELECT
COUNT(B.NFC_ID) FROM haveAccess B WHERE B.AREA_ID=C.AREA_ID AND B.end_d
atetime>'{}') < C.beds))
```

```
INSERT INTO register(NFC_ID, SERVICE_ID, date_time)
                                    VALUES ({}, (SELECT SERVICE_ID FROM
 services WHERE category='{}'), '{}');
```

```
SELECT AREA_ID FROM provide WHERE SERVICE_ID=(SELECT SERVICE_ID FROM se
rvices WHERE category='{}');
```

```
INSERT INTO haveAccess(NFC_ID, AREA_ID, start_datetime, end_datetime)
                                    VALUES ({}, {}, '{}', '{}');
```

```
SELECT EXISTS (SELECT * FROM register WHERE NFC_ID={} AND SERVICE_ID=(S
ELECT SERVICE_ID FROM services WHERE category='{}'));
```

```
SELECT EXISTS (SELECT * FROM haveAccess WHERE NFC_ID={} AND AREA_ID=(SE
LECT AREA_ID FROM areas WHERE area_name='{}'));
```

```
DELETE FROM register WHERE NFC_ID={} AND SERVICE_ID=(SELECT SERVICE_ID
FROM services WHERE category='{}');
```

```
DELETE FROM haveAccess WHERE NFC_ID={} AND AREA_ID=(SELECT AREA_ID FROM
 areas WHERE area_name='{}');
```

```
DELETE FROM haveAccess WHERE NFC_ID={} AND AREA_ID={};
```

```
DELETE FROM haveAccess WHERE NFC_ID={} AND AREA_ID=460;
```

```
DELETE FROM haveAccess WHERE NFC_ID={} AND AREA_ID=461;
```

```
DELETE FROM haveAccess WHERE NFC_ID={} AND AREA_ID=(SELECT AREA_ID FROM
 areas WHERE area_floor=(SELECT area_floor FROM areas WHERE AREA_ID=(SE
LECT AREA_ID FROM areas WHERE area_name='{}')) AND
                                    orientation=(SELECT orientation FRO
M areas WHERE AREA_ID=(SELECT AREA_ID FROM areas WHERE area_name='{}'))
 AND beds=0)
```

```
DELETE FROM register WHERE NFC_ID={} AND SERVICE_ID=(SELECT SERVICE_ID
FROM services WHERE category='{}');
```

```
SELECT AREA_ID FROM provide WHERE SERVICE_ID=(SELECT SERVICE_ID FROM se
rvices WHERE category='{}');
```

Statistics.py

```
SELECT B.area_name, count(A.AREA_ID) AS visit
            FROM visit A, areas B, customers C
            WHERE A.AREA_ID=B.AREA_ID AND A.AREA_ID>400 AND A.NFC_ID=C.
NFC_ID
                AND C.birth_date<='{}-12-31' AND C.birth_date>='{}-01-
01'
                AND A.enter_datetime>='{}'
            GROUP BY A.AREA_ID
            ORDER BY visit DESC LIMIT 15;
```

```
SELECT B.category, count(A.SERVICE_ID) AS useService
            FROM useService A, services B, customers C
            WHERE A.SERVICE_ID=B.SERVICE_ID AND A.SERVICE_ID>1 AND A.NF
C_ID=C.NFC_ID
```

```
                AND C.birth_date<='{}-12-31' AND C.birth_date>='{}-01-
01'
                AND A.date_time>='{}'
            GROUP BY A.SERVICE_ID
            ORDER BY useService DESC;
```

```
SELECT B.category, count(DISTINCT A.NFC_ID) AS useService
            FROM useService A, services B, customers C
            WHERE A.SERVICE_ID=B.SERVICE_ID AND A.SERVICE_ID>1 AND A.NF
C_ID=C.NFC_ID
                AND C.birth_date<='{}-12-31' AND C.birth_date>='{}-01-
01'
                AND A.date_time>='{}'
            GROUP BY A.SERVICE_ID
            ORDER BY useService DESC;
```

Views.py

```
SELECT * FROM customers_info;
```

```
SELECT * FROM services_sales;
```

3. Rest of the Code

Database_Project/app/backend/

- __init__.py

```
from .home import home
from .records import records
from .views import views
from .covid import covid
from .statistics import statistics
from .register import register
```

- Covid.py

```
from flask import Blueprint, render_template, request, flash
from flask_mysqldb import MySQL
from .. import mysql

covid = Blueprint('covid', __name__)

@covid.route('/covid', methods=['GET', 'POST'])
def covid_check():
    status=200
    cur = mysql.connection.cursor()
    cur2 = mysql.connection.cursor()
    if request.method == 'POST':
        try:
            person_id = request.form.get("person_id")

            my_query1 = """SELECT MAX(NFC_ID) FROM customers;"""

            print(my_query1)
            cur2.execute(my_query1)
            result1 = cur2.fetchall()

            if not (person_id and person_id.isnumeric()) or int(person_
id)<=0 or int(person_id)>result1[0][0]:
                status = 400
                flash("Please enter a valid NFC ID", category='error')
                return render_template("covid.html")

            else:

                if request.form.get("areas"):
                    my_query = """SELECT A.AREA_ID, B.area_name, A.ente
r_datetime, A.leave_datetime
                        FROM visit A, areas B
                        WHERE A.AREA_ID=B.AREA_ID AND NFC_ID={}
                        ORDER BY leave_datetime DESC;
                        """.format(person_id)
```

```
                    x =  0

                elif request.form.get("contacts"):
                    my_query = """SELECT A.NFC_ID, A.last_name, A.first
_name, GROUP_CONCAT(distinct customer_phones.phone_number), GROUP_CONCA
T(distinct customer_emails.email_address)
                        FROM customers A, customer_phones, customer_ema
ils, visit B, visit C
                        WHERE A.NFC_ID<>{} AND
                            A.NFC_ID=customer_phones.NFC_ID AND A.NFC_I
D=customer_emails.NFC_ID AND
                            B.NFC_ID={} AND C.NFC_ID=A.NFC_ID AND B.ARE
A_ID=C.AREA_ID AND
                            C.leave_datetime > B.enter_datetime AND C.e
nter_datetime < CONVERT(ADDTIME(STR_TO_DATE(B.leave_datetime,'%Y-%m-
%d %H:%i:%s'),'1:0:0'), CHAR)
                        GROUP BY A.NFC_ID
                        ORDER BY A.NFC_ID DESC
                        """.format(person_id, person_id)
                    x = 1

                print(my_query)
                cur.execute(my_query)
                results = cur.fetchall()
                flash("Here are your results!", category='success')
                return render_template("covid.html", results=results, p
ick=x)

        except:
            status = 400
            flash("Please enter a valid NFC ID", category='error')
            return render_template("covid.html")




    return render_template("covid.html")
```

- Home.py

```
from flask import Blueprint, render_template, request, flash
from flask_mysqldb import MySQL
import mysql.connector as ddb

home = Blueprint('home', __name__)

@home.route('/', methods=['GET', 'POST'])
def home_fun():
```

```
    status=200

    mydb = ddb.connect(
        host = "localhost",
        user = "root",
        passwd = "boftonelly",
        database = "HotelDB"
    )

    cur = mydb.cursor()
    cur2 = mydb.cursor()

    if request.method == 'POST':

        phones = []
        emails = []

        nfc_id = request.form.get('NFC_ID')
        last_name = request.form.get('last_name')
        first_name = request.form.get('first_name')
        birth_date = request.form.get('birth_date')
        id = request.form.get('id')
        id_type = request.form.get('id_type')
        id_issue = request.form.get('id_issue')
        phone1 = request.form.get('phone1')
        if phone1:
            phones.append(phone1)
        phone2 = request.form.get('phone2')
        if phone2:
            phones.append(phone2)
        phone3 = request.form.get('phone3')
        if phone3:
            phones.append(phone3)
        email1 = request.form.get('email1')
        if email1:
            emails.append(email1)
        email2 = request.form.get('email2')
        if email2:
            emails.append(email2)
        email3 = request.form.get('email3')
        if email3:
            emails.append(email3)

        if request.form.get("new"):
            if not (last_name and first_name and birth_date and id and
id_type and id_issue and (phone1 or phone2 or phone3) and (email1 or em
ail2 or email3)):
```

```
                flash('Please complete every field (at least 1 phone an
d 1 e-mail), except NFC_ID, to add a new customer', category="error")
                status=400
                return render_template("home.html")

            elif not (id.isnumeric() and (phone1.isnumeric() or phone2.
isnumeric() or phone3.isnumeric())):
                flash('Please insert only numbers in identification and
 phone fields', category="error")
                status=400
                return render_template("home.html")

            else:

                phones = list(set(phones))
                emails = list(set(emails))


                my_query1 = """INSERT INTO customers(last_name, first_n
ame, birth_date, id, id_type, id_issue)
                            VALUES ('{}', '{}', '{}', {}, '{}', '{}');
                    """.format(last_name, first_name, birth_date, id, i
d_type, id_issue)

                print(my_query1)
                cur.execute(my_query1)
                mydb.commit()

                my_query11 = """SELECT MAX(NFC_ID) FROM customers;"""

                print(my_query11)
                cur2.execute(my_query11)
                result1 = cur2.fetchall()

                for item in phones:
                    my_query2 = """INSERT INTO customer_phones(NFC_ID,
phone_number)
                                VALUES ((SELECT MAX(NFC_ID) FROM custom
ers), {});
                        """.format(item)

                    print(my_query2)
                    cur.execute(my_query2)
                    mydb.commit()

                for item in emails:
                    my_query3 = """INSERT INTO customer_emails(NFC_ID,
email_address)
```

```
                                VALUES ((SELECT MAX(NFC_ID) FROM custom
ers), '{}');
                        """.format(item)

                    print(my_query3)
                    cur.execute(my_query3)
                    mydb.commit()

                flash('Customer added successfully with NFC ID {} !'.fo
rmat(result1[0][0]))
                return render_template("home.html")

        elif request.form.get("modify"):

            my_query1 = """SELECT MAX(NFC_ID) FROM customers;"""

            print(my_query1)
            cur.execute(my_query1)
            result = cur.fetchall()

            if not (nfc_id and nfc_id.isnumeric() and ((phone1 and phon
e1.isnumeric()) or (phone2 and phone2.isnumeric()) or (phone3 and phone
3.isnumeric()) or email1 or email2 or email3)):
                flash('Please provide a valid NFC_ID and at least 1 pho
ne or 1 e-mail', category="error")
                status=400
                return render_template("home.html")

            elif int(nfc_id)<=0 or int(nfc_id)>result[0][0]:
                flash('Please provide a valid NFC_ID and at least 1 pho
ne or 1 e-mail', category="error")
                status=400
                return render_template("home.html")

            else:

                phones = list(set(phones))
                emails = list(set(emails))


                if len(phones):
                    my_query4 = """DELETE FROM customer_phones WHERE NF
C_ID={};""".format(nfc_id)
                    print(my_query4)
                    cur.execute(my_query4)
                    mydb.commit()

                if len(emails):
```

```
                    my_query5 = """DELETE FROM customer_emails WHERE NF
C_ID={};""".format(nfc_id)
                    print(my_query5)
                    cur.execute(my_query5)
                    mydb.commit()

                for item in phones:

                    my_query2 = """INSERT INTO customer_phones(NFC_ID,
phone_number)
                                VALUES ({}, {});
                        """.format(nfc_id, item)

                    print(my_query2)
                    cur.execute(my_query2)
                    mydb.commit()

                for item in emails:
                    my_query3 = """INSERT INTO customer_emails(NFC_ID,
email_address)
                                VALUES ({}, '{}');
                        """.format(nfc_id, item)

                    print(my_query3)
                    cur.execute(my_query3)
                    mydb.commit()

                flash('Information of customer updated successfully!')
                return render_template("home.html")

        elif request.form.get("delete"):

            my_query1 = """SELECT MAX(NFC_ID) FROM customers;"""

            print(my_query1)
            cur.execute(my_query1)
            result = cur.fetchall()

            if not (nfc_id and nfc_id.isnumeric()) or int(nfc_id)<=0 or
 int(nfc_id)>result[0][0]:
                flash('Please provide a valid NFC_ID for deletion', cat
egory="error")
                status=400
                return render_template("home.html")

            else:
```

```
                my_query2 = """DELETE FROM customers WHERE NFC_ID={};""
".format(nfc_id)
                print(my_query2)
                cur.execute(my_query2)
                mydb.commit()

                flash('Customer deleted successfully!')
                return render_template("home.html")


    return render_template("home.html")
```

- Records.py

```
from flask import Blueprint, render_template, request, flash
from flask_mysqldb import MySQL
from .. import mysql
from datetime import date, timedelta

records = Blueprint('records', __name__)

@records.route('/service_records', methods=['GET', 'POST'])
def serv_rec():
    status=200
    cur = mysql.connection.cursor()
    if request.method == 'POST':
        datefrom = request.form.get('datefrom')
        dateto = request.form.get('dateto')
        costfrom = request.form.get('costfrom')
        costto = request.form.get('costto')
        service_type = request.form.get('service_type')


        if not datefrom:
            datefrom = "1900-00-00"
        if not dateto:
            dateto = str(date.today() + timedelta(days=5))
        if not costfrom:
            costfrom = 0
        if not costto:
            costto = 99999
        datefrom = datefrom + " 00:00:00"
        dateto = dateto + " 23:59:59"
        if service_type=="All":
            my_query = """SELECT A.NFC_ID, B.last_name, B.first_name, C
.category, A.cost, D.enter_datetime, D.leave_datetime
            FROM useService A, customers B, services C, visit D
            WHERE
                A.date_time >= '{}'
```

```
                AND A.date_time <= '{}'
                AND A.cost >= {}
                AND A.cost <= {}
                AND A.SERVICE_ID <> 1
                AND C.SERVICE_ID = A.SERVICE_ID
                AND A.NFC_ID = B.NFC_ID
                AND D.NFC_ID = A.NFC_ID
                AND D.enter_datetime <= A.date_time
                AND D.leave_datetime >= A.date_time
            GROUP BY A.date_time
            ORDER BY A.date_time DESC;
            """.format(datefrom, dateto, costfrom, costto)

        else:
            my_query = """SELECT A.NFC_ID, B.last_name, B.first_name, C
.category, A.cost, D.enter_datetime, D.leave_datetime
            FROM useService A, customers B, services C, visit D
            WHERE
                A.date_time >= '{}'
                AND A.date_time <= '{}'
                AND A.cost >= {}
                AND A.cost <= {}
                AND A.SERVICE_ID = (SELECT SERVICE_ID
                                    FROM services
                                    WHERE category='{}')
                AND C.SERVICE_ID = A.SERVICE_ID
                AND A.NFC_ID = B.NFC_ID
                AND D.NFC_ID = A.NFC_ID
                AND D.enter_datetime <= A.date_time
                AND D.leave_datetime >= A.date_time
            ORDER BY D.leave_datetime DESC;
            """.format(datefrom, dateto, costfrom, costto, service_type
)
        print(my_query)
        cur.execute(my_query)
        results = cur.fetchall()
        flash("Here are your results!", category='success')
        return render_template("serv_rec.html", results=results, servic
e=service_type)

    return render_template("serv_rec.html")
```

- Register.py

```
from flask import Blueprint, render_template, request, flash
from flask_mysqldb import MySQL
import mysql.connector as ddb
import datetime
```

```
register = Blueprint('register', __name__)

@register.route('/register', methods=['GET', 'POST'])
def register_fun():
    status=200

    mydb = ddb.connect(
        host = "localhost",
        user = "root",
        passwd = "boftonelly",
        database = "HotelDB"
    )

    cur = mydb.cursor()
    cur2 = mydb.cursor()
    cur3 = mydb.cursor()
    cur4 = mydb.cursor()
    cur5 = mydb.cursor()
    cur6 = mydb.cursor()
    cur7 = mydb.cursor()

    availables = """ SELECT C.area_name FROM areas C WHERE  C.AREA_ID<4
01 AND ((SELECT COUNT(B.NFC_ID) FROM haveAccess B WHERE B.AREA_ID=C.ARE
A_ID AND B.end_datetime>'{}') < C.beds)""".format(datetime.datetime.now
())
    cur3.execute(availables)
    av_rooms = cur3.fetchall()

    taken = """ SELECT M.area_name FROM areas M WHERE M.AREA_ID<401 AND
 M.area_name NOT IN (SELECT C.area_name FROM areas C WHERE  C.AREA_ID<4
01 AND ((SELECT COUNT(B.NFC_ID) FROM haveAccess B WHERE B.AREA_ID=C.ARE
A_ID AND B.end_datetime>'{}') < C.beds))""".format(datetime.datetime.no
w())
    cur6.execute(taken)
    tk_rooms = cur6.fetchall()


    if request.method == 'POST':

        nfc_id = request.form.get('NFC_ID')
        service_name = request.form.get('service_name')
        room = request.form.get('room')
        room2 = request.form.get('room2')
        days1 = request.form.get('days')


        if request.form.get("register"):
```

```
            my_query1 = """SELECT MAX(NFC_ID) FROM customers;"""

            print(my_query1)
            cur.execute(my_query1)
            result = cur.fetchall()

            if not (nfc_id and service_name):
                flash('Please input an NFC_ID and a provided service',
category="error")
                status=400
                return render_template("register.html", x=av_rooms, y=t
k_rooms)

            elif (not nfc_id.isnumeric()) or (not days1.isnumeric()):
                flash('Please insert a valid NFC_ID and days duration',
 category="error")
                status=400
                return render_template("register.html", x=av_rooms, y=t
k_rooms)

            elif int(nfc_id)<1 or int(nfc_id)>result[0][0]:
                flash('Please insert a valid NFC_ID', category="error")
                status=400
                return render_template("register.html", x=av_rooms, y=t
k_rooms)

            else:

                query22 = """SELECT EXISTS (SELECT * FROM register WHER
E NFC_ID={} AND SERVICE_ID=(SELECT SERVICE_ID FROM services WHERE categ
ory='{}'));
                 """.format(nfc_id, service_name)
                print(query22)
                cur5.execute(query22)
                check = cur5.fetchall()

                if check[0][0]==True:
                    flash('Customer with NFC ID {} is already registere
d at {}!'.format(nfc_id, service_name), category="error")
                    return render_template("register.html", x=av_rooms,
 y=tk_rooms)

                else:
                    if service_name=="Room":
                        my_query2 = """INSERT INTO register(NFC_ID, SER
VICE_ID, date_time)
                                       VALUES ({}, 1, '{}');
```

```
                            """.format(nfc_id, datetime.datetime.now().
strftime("%Y-%m-%d %H:%M:%S"))

                        print(my_query2)
                        cur.execute(my_query2)
                        mydb.commit()

                        my_query3 = """INSERT INTO haveAccess(NFC_ID, A
REA_ID, start_datetime, end_datetime)
                                    VALUES ({}, (SELECT AREA_ID FROM ar
eas WHERE area_name='{}'), '{}', '{}');
                                    """.format(nfc_id, room, datetime.datetime.
now().strftime("%Y-%m-
%d %H:%M:%S"), (datetime.datetime.now()+datetime.timedelta(days=int(day
s1))).strftime("%Y-%m-%d %H:%M:%S"))

                        print(my_query3)
                        cur.execute(my_query3)
                        mydb.commit()

                        my_query4 = """INSERT INTO useService(NFC_ID, S
ERVICE_ID, date_time, cost)
                                    VALUES ({}, 1, '{}', {});
                                """.format(nfc_id, datetime.datetime.now().
strftime("%Y-%m-%d %H:%M:%S"), 30*int(days1))

                        print(my_query4)
                        cur.execute(my_query4)
                        mydb.commit()

                        for i in range(421,436):
                            my_query5 = """INSERT INTO haveAccess(NFC_I
D, AREA_ID, start_datetime, end_datetime)
                                    VALUES ({}, {}, '{}', '{}');
                                """.format(nfc_id, i, datetime.datetime.now
().strftime("%Y-%m-
%d %H:%M:%S"), (datetime.datetime.now()+datetime.timedelta(days=int(day
s1))).strftime("%Y-%m-%d %H:%M:%S"))

                            print(my_query5)
                            cur.execute(my_query5)
                            mydb.commit()

                        my_query6 = """INSERT INTO haveAccess(NFC_ID, A
REA_ID, start_datetime, end_datetime)
                                    VALUES ({}, 460, '{}', '{}');
                                """.format(nfc_id, datetime.datetime.now().
strftime("%Y-%m-
```

```
%d %H:%M:%S"), (datetime.datetime.now()+datetime.timedelta(days=int(day
s1))).strftime("%Y-%m-%d %H:%M:%S"))

                        print(my_query6)
                        cur.execute(my_query6)
                        mydb.commit()

                        my_query6 = """INSERT INTO haveAccess(NFC_ID, A
REA_ID, start_datetime, end_datetime)
                                    VALUES ({}, 461, '{}', '{}');
                                """.format(nfc_id, datetime.datetime.now().
strftime("%Y-%m-
%d %H:%M:%S"), (datetime.datetime.now()+datetime.timedelta(days=int(day
s1))).strftime("%Y-%m-%d %H:%M:%S"))

                        print(my_query6)
                        cur.execute(my_query6)
                        mydb.commit()

                        my_query7 = """INSERT INTO haveAccess (NFC_ID,A
REA_ID,start_datetime,end_datetime)
                                    VALUES ({},(SELECT AREA_ID FROM areas W
HERE area_floor=(SELECT area_floor FROM areas WHERE AREA_ID=(SELECT ARE
A_ID FROM areas WHERE area_name='{}')) AND
                                    orientation=(SELECT orientation FROM ar
eas WHERE AREA_ID=(SELECT AREA_ID FROM areas WHERE area_name='{}')) AND
 beds=0),'{}','{}')""".format(nfc_id,room,room,datetime.datetime.now(),
 (datetime.datetime.now()+datetime.timedelta(days=int(days1))).strftime
("%Y-%m-%d %H:%M:%S"))
                        print(my_query7)
                        cur.execute(my_query7)
                        mydb.commit()

                        availables = """ SELECT C.area_name FROM areas
C WHERE  C.AREA_ID<401 AND ((SELECT COUNT(B.NFC_ID) FROM haveAccess B W
HERE B.AREA_ID=C.AREA_ID AND B.end_datetime>'{}') < C.beds)""".format(d
atetime.datetime.now())
                        cur4.execute(availables)
                        av_rooms = cur4.fetchall()

                        taken = """ SELECT M.area_name FROM areas M WHE
RE M.AREA_ID<401 AND M.area_name NOT IN (SELECT C.area_name FROM areas
C WHERE  C.AREA_ID<401 AND ((SELECT COUNT(B.NFC_ID) FROM haveAccess B W
HERE B.AREA_ID=C.AREA_ID AND B.end_datetime>'{}') < C.beds))""".format(
datetime.datetime.now())
                        cur7.execute(taken)
                        tk_rooms = cur7.fetchall()
```

```
                        flash('Customer with NFC ID {} renting {}!'.for
mat(nfc_id, room))
                        return render_template("register.html", x=av_ro
oms, y=tk_rooms)

                    else:


                        my_query2 = """INSERT INTO register(NFC_ID, SER
VICE_ID, date_time)
                                       VALUES ({}, (SELECT SERVICE_ID FROM
 services WHERE category='{}'), '{}');
                                """.format(nfc_id, service_name, datetime.d
atetime.now())

                        print(my_query2)
                        cur.execute(my_query2)
                        mydb.commit()

                        my_query11 = """SELECT AREA_ID FROM provide WHE
RE SERVICE_ID=(SELECT SERVICE_ID FROM services WHERE category='{}');"""
.format(service_name)

                        print(my_query11)
                        cur2.execute(my_query11)
                        result1 = cur2.fetchall()

                        for item in result1:
                            my_query2 = """INSERT INTO haveAccess(NFC_I
D, AREA_ID, start_datetime, end_datetime)
                                           VALUES ({}, {}, '{}', '{}');
                                    """.format(nfc_id, item[0], datetime.dateti
me.now().strftime("%Y-%m-
%d %H:%M:%S"), (datetime.datetime.now()+datetime.timedelta(days=int(day
s1))).strftime("%Y-%m-%d %H:%M:%S"))

                            print(my_query2)
                            cur.execute(my_query2)
                            mydb.commit()

                    flash('Customer with NFC ID {} registered successfu
lly at {}!'.format(nfc_id, service_name))
                    return render_template("register.html", x=av_rooms,
 y=tk_rooms)

        elif request.form.get('unregister'):
            my_query1 = """SELECT MAX(NFC_ID) FROM customers;"""

            print(my_query1)
```

```
            cur.execute(my_query1)
            result = cur.fetchall()

            if not (nfc_id and service_name):
                flash('Please input an NFC_ID and a provided service',
category="error")
                status=400
                return render_template("register.html", x=av_rooms, y=t
k_rooms)

            elif (not nfc_id.isnumeric()):
                flash('Please insert a valid NFC_ID', category="error")
                status=400
                return render_template("register.html", x=av_rooms, y=t
k_rooms)

            elif int(nfc_id)<1 or int(nfc_id)>result[0][0]:
                flash('Please insert a valid NFC_ID', category="error")
                status=400
                return render_template("register.html", x=av_rooms, y=t
k_rooms)

            else:
                query22 = """SELECT EXISTS (SELECT * FROM register WHER
E NFC_ID={} AND SERVICE_ID=(SELECT SERVICE_ID FROM services WHERE categ
ory='{}'));
                 """.format(nfc_id, service_name)
                print(query22)
                cur5.execute(query22)
                check = cur5.fetchall()

                if check[0][0]==False:
                    flash('Customer with NFC ID {} is not registered at
 {}!'.format(nfc_id, service_name), category="error")
                    return render_template("register.html", x=av_rooms,
 y=tk_rooms)

                else:

                    if service_name=="Room":

                        myquery33 = """SELECT EXISTS (SELECT * FROM hav
eAccess WHERE NFC_ID={} AND AREA_ID=(SELECT AREA_ID FROM areas WHERE ar
ea_name='{}'));
                        """.format(nfc_id, room2)
                        print(myquery33)
                        cur2.execute(myquery33)
                        check2 = cur2.fetchall()
```

```
                        if check2[0][0]==False:
                            flash('Customer with NFC ID {} was not link
ed with {}!'.format(nfc_id, room2), category="error")
                            return render_template("register.html", x=a
v_rooms, y=tk_rooms)

                        else:
                            my_query2 = """DELETE FROM register WHERE N
FC_ID={} AND SERVICE_ID=(SELECT SERVICE_ID FROM services WHERE category
='{}');
                                """.format(nfc_id, service_name)

                            print(my_query2)
                            cur.execute(my_query2)
                            mydb.commit()

                            my_query3 = """DELETE FROM haveAccess WHERE
 NFC_ID={} AND AREA_ID=(SELECT AREA_ID FROM areas WHERE area_name='{}')
;
                                """.format(nfc_id, room2)

                            print(my_query3)
                            cur.execute(my_query3)
                            mydb.commit()

                            for i in range(421,436):
                                my_query5 = """DELETE FROM haveAccess W
HERE NFC_ID={} AND AREA_ID={};
                                """.format(nfc_id, i)

                                print(my_query5)
                                cur.execute(my_query5)
                                mydb.commit()

                            my_query6 = """DELETE FROM haveAccess WHERE
 NFC_ID={} AND AREA_ID=460;
                                """.format(nfc_id)

                            print(my_query6)
                            cur.execute(my_query6)
                            mydb.commit()

                            my_query6 = """DELETE FROM haveAccess WHERE
 NFC_ID={} AND AREA_ID=461;
                                """.format(nfc_id)

                            print(my_query6)
```

```
                            cur.execute(my_query6)
                            mydb.commit()

                            my_query7 = """DELETE FROM haveAccess WHERE
 NFC_ID={} AND AREA_ID=(SELECT AREA_ID FROM areas WHERE area_floor=(SEL
ECT area_floor FROM areas WHERE AREA_ID=(SELECT AREA_ID FROM areas WHER
E area_name='{}')) AND
                                        orientation=(SELECT orientation FRO
M areas WHERE AREA_ID=(SELECT AREA_ID FROM areas WHERE area_name='{}'))
 AND beds=0)""".format(nfc_id,room,room2)
                            print(my_query7)
                            cur.execute(my_query7)
                            mydb.commit()

                            availables = """ SELECT C.area_name FROM ar
eas C WHERE  C.AREA_ID<401 AND ((SELECT COUNT(B.NFC_ID) FROM haveAccess
 B WHERE B.AREA_ID=C.AREA_ID AND B.end_datetime>'{}') < C.beds)""".form
at(datetime.datetime.now())
                            cur4.execute(availables)
                            av_rooms = cur4.fetchall()

                            taken = """ SELECT M.area_name FROM areas M
 WHERE M.AREA_ID<401 AND M.area_name NOT IN (SELECT C.area_name FROM ar
eas C WHERE  C.AREA_ID<401 AND ((SELECT COUNT(B.NFC_ID) FROM haveAccess
 B WHERE B.AREA_ID=C.AREA_ID AND B.end_datetime>'{}') < C.beds))""".for
mat(datetime.datetime.now())
                            cur7.execute(taken)
                            tk_rooms = cur7.fetchall()


                            flash('Customer with NFC ID {} unregistered
 from {}!'.format(nfc_id, room2))
                            return render_template("register.html", x=a
v_rooms, y=tk_rooms)

                    else:

                        my_query2 = """DELETE FROM register WHERE NFC_I
D={} AND SERVICE_ID=(SELECT SERVICE_ID FROM services WHERE category='{}
');
                            """.format(nfc_id, service_name)

                        print(my_query2)
                        cur.execute(my_query2)
                        mydb.commit()

                        my_query11 = """SELECT AREA_ID FROM provide WHE
RE SERVICE_ID=(SELECT SERVICE_ID FROM services WHERE category='{}');"""
.format(service_name)
```

```
                        print(my_query11)
                        cur2.execute(my_query11)
                        result1 = cur2.fetchall()

                        for item in result1:
                            my_query2 = """DELETE FROM haveAccess WHERE
 NFC_ID={} AND AREA_ID={};
                            """.format(nfc_id, item[0])

                            print(my_query2)
                            cur.execute(my_query2)
                            mydb.commit()

                    flash('Customer with NFC ID {} unregistered success
fully from {}!'.format(nfc_id, service_name))
                    return render_template("register.html", x=av_rooms,
 y=tk_rooms)

    return render_template("register.html", x=av_rooms, y=tk_rooms)
```

- Statistics.py

```
from flask import Blueprint, render_template, request, flash
from flask_mysqldb import MySQL
from .. import mysql
import datetime

statistics = Blueprint('statistics', __name__)

@statistics.route('/statistics', methods=['GET', 'POST'])
def stats():
    status=200
    cur = mysql.connection.cursor()
    if request.method == 'POST':
        age_group = request.form.get("age_group")
        if age_group == "20-40":
            age_from = 20
            age_to = 40
        elif age_group == "41-60":
            age_from = 41
            age_to = 60
        else:
            age_from = 61
            age_to = 1000

        time_period = request.form.get("time_period")
        if time_period == "Last Year":
```

```
            date_from = (datetime.datetime.now()-
datetime.timedelta(days=365)).strftime("%Y-%m-%d %H:%M:%S")
        else:
            date_from = (datetime.datetime.now()-
datetime.timedelta(days=30)).strftime("%Y-%m-%d %H:%M:%S")

        if request.form.get("areas"):
            my_query = """SELECT B.area_name, count(A.AREA_ID) AS visit

            FROM visit A, areas B, customers C
            WHERE A.AREA_ID=B.AREA_ID AND A.AREA_ID>400 AND A.NFC_ID=C.
NFC_ID
                AND C.birth_date<='{}-12-31' AND C.birth_date>='{}-01-
01'
                AND A.enter_datetime>='{}'
            GROUP BY A.AREA_ID
            ORDER BY visit DESC LIMIT 15;
                """.format(datetime.datetime.now().year-
age_from, datetime.datetime.now().year-age_to, date_from)
            x =  0

        elif request.form.get("services"):
            my_query = """SELECT B.category, count(A.SERVICE_ID) AS use
Service
            FROM useService A, services B, customers C
            WHERE A.SERVICE_ID=B.SERVICE_ID AND A.SERVICE_ID>1 AND A.NF
C_ID=C.NFC_ID
                AND C.birth_date<='{}-12-31' AND C.birth_date>='{}-01-
01'
                AND A.date_time>='{}'
            GROUP BY A.SERVICE_ID
            ORDER BY useService DESC;
                """.format(datetime.datetime.now().year-
age_from, datetime.datetime.now().year-age_to, date_from)
            x = 1

        elif request.form.get("popular_services"):
            my_query = """SELECT B.category, count(DISTINCT A.NFC_ID) A
S useService
            FROM useService A, services B, customers C
            WHERE A.SERVICE_ID=B.SERVICE_ID AND A.SERVICE_ID>1 AND A.NF
C_ID=C.NFC_ID
                AND C.birth_date<='{}-12-31' AND C.birth_date>='{}-01-
01'
                AND A.date_time>='{}'
            GROUP BY A.SERVICE_ID
            ORDER BY useService DESC;
```

```
                """.format(datetime.datetime.now().year-
age_from, datetime.datetime.now().year-age_to, date_from)
            x = 2

        print(my_query)
        cur.execute(my_query)
        results = cur.fetchall()
        flash("Here are your results!", category='success')
        return render_template("statistics.html", results=results, pick
=x, y=age_group, t=time_period)


    return render_template("statistics.html")
```

- Views.py

```
from flask import Blueprint, render_template, request, flash
from flask_mysqldb import MySQL
from .. import mysql

views = Blueprint('views', __name__)

@views.route('/views', methods=['GET', 'POST'])
def get_view():
    status=200
    cur = mysql.connection.cursor()
    if request.method == 'POST':
        x = request.form.get("info")

        if x == 'Customer Info':
            my_query = """SELECT * FROM customers_info;
                """

        elif x == 'Service Sales':
            my_query = """SELECT * FROM services_sales;
                """

        print(my_query)
        cur.execute(my_query)
        results = cur.fetchall()
        flash("Here are your results!", category='success')
        return render_template("views.html", results=results, data=x)


    return render_template("views.html")
```

Database_Project/app/frontend/static/

- Styles.css

```
*, *::before, *::after {
    box-sizing: border-box;
}

body {
    padding: 0;
    margin: 0;
}

table, th, td {
    border: 1px solid black;
  }
```

Database_Project/app/frontend/templates/

- Base.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-
scale=1" />
    <link
      rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/boot
strap.min.css"
      integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
      crossorigin="anonymous"
    />
    <link
      rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/font-
awesome/4.7.0/css/font-awesome.min.css"
      crossorigin="anonymous"
    />

    <link rel="stylesheet" href={{ url_for('static', filename='styles.c
ss') }}>

    <title>{% block title %}Home{% endblock %}</title>
```

```
  </head>
  <body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
      <button
        class="navbar-toggler"
        type="button"
        data-toggle="collapse"
        data-target="#navbar"
      >
        <span class="navbar-toggler-icon"></span>
      </button>
      <div class="collapse navbar-collapse" id="navbar">
        <div class="navbar-nav">
          <a class="nav-item nav-link" id="home" href="/">Home</a>
          <a class="nav-item nav-
link" id="register" href="/register">Register</a>
          <a class="nav-item nav-
link" id="service_records" href="/records/service_records">Records</a>
          <a class="nav-item nav-
link" id="views" href="/views">Views</a>
          <a class="nav-item nav-
link" id="covid" href="/covid">Covid</a>
          <a class="nav-item nav-
link" id="statistics" href="/statistics">Statistics</a>
        </div>
      </div>
    </nav>

    {% with messages = get_flashed_messages(with_categories=true) %} {%
 if
    messages %} {% for category, message in messages %} {% if category
==
    'error' %}
    <div class="alert alert-danger alter-
dismissable fade show" role="alert">
      {{ message }}
      <button type="button" class="close" data-dismiss="alert">
        <span aria-hidden="true">&times;</span>
      </button>
    </div>
    {% else %}
    <div class="alert alert-success alter-
dismissable fade show" role="alert">
      {{ message }}
      <button type="button" class="close" data-dismiss="alert">
        <span aria-hidden="true">&times;</span>
      </button>
    </div>
```

```
    {% endif %} {% endfor %} {% endif %} {% endwith %}

    <div class="container">{% block content %} {% endblock %}</div>
    <script
      src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
      integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
      crossorigin="anonymous"
    ></script>
    <script
      src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/
popper.min.js"
      integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
      crossorigin="anonymous"
    ></script>
    <script
      src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap
.min.js"
      integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
      crossorigin="anonymous"
    ></script>

  </body>
</html>
```

- Covid.html

```
{% extends "base.html" %} {% block title %}Covid Search{% endblock %} {
% block content
%}

<head>
<style>
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
table {
    width: 90%;
}
</style>
</head>
<form method="POST">
  <h3 align="center">Covid Spread Check</h3>
  <div class="form-group">
```

```
    <label for="person_id">Infected Customer's NFC ID</label>
    <input
      type="integer"
      class="form-control"
      id="person_id"
      name="person_id"
      placeholder="Infected Customer's NFC ID"
    />

  <br />
  <div align="center">
    <button type="submit" name="areas" value="areas" class="btn btn-
primary" style="display:inline-block;">Area Visits</button>
    <button type="submit" name="contacts" value="contacts" class="btn b
tn-primary" style="display:inline-block;margin-
top:0;">Customer Contacts</button>
</div>
</form>
<br />
<br />
{% if pick == 0%}
  <h3>List of Area Visits</h3>
  <div align="center" style="width:100%, overflow: auto;">
    <table>
      <tr>
        <th>AREA ID</th>
        <th>Area Name</th>
        <th>Entered at</th>
        <th>Left at</th>
      </tr>
      {% if results != None%}
        {% for result in results %}
        <tr>
        <td>{{result[0]}}</td>
        <td>{{result[1]}}</td>
        <td>{{result[2]}}</td>
        <td>{{result[3]}}</td>
        </tr>
        {% endfor %}
      {% endif %}
    </table>
  </div>
{% endif %}
{% if pick == 1%}
  <h3>List of Covid Contacts</h3>
  <div align="center" style="width:100%, overflow: auto;">
    <table>
      <tr>
```

```
        <th>NFC ID</th>
        <th>Last Name</th>
        <th>First Name</th>
        <th>Phone Numbers</th>
        <th>E-mail Addresses</th>
      </tr>
      {% if results != None%}
        {% for result in results %}
        <tr>
        <td>{{result[0]}}</td>
        <td>{{result[1]}}</td>
        <td>{{result[2]}}</td>
        <td>{{result[3]}}</td>
        <td>{{result[4]}}</td>
        </tr>
        {% endfor %}
      {% endif %}
    </table>
  </div>
{% endif %}
{% endblock %}
```

- Home.html

```
{% extends "base.html" %} {% block title %}Home{% endblock %} {% block
content
%}
<h1 align="center">Welcome!!!</h1>
<h3 align="center">Add a new customer/ Change phone number or email add
ress/ Delete an existing customer</h3>
</br>
</br>
<form method="POST">

  <h5> A new customer gets automatically an NFC-
ID, so the field is not used at this case.</h5>
  </br>
  <div class="form-group">
    <label for="NFC_ID">NFC ID</label>
    <input
      type="int"
      class="form-control"
      id="NFC_ID"
      name="NFC_ID"
      placeholder="Existing customer NFC ID"
    />
  </div>
  <div class="form-group">
```

```
    <label for="last_name">Last Name</label>
    <input
      type="text"
      class="form-control"
      id="last_name"
      name="last_name"
      placeholder="Last Name"
    />
  </div>
  <div class="form-group">
    <label for="first_name">First Name</label>
    <input
      type="text"
      class="form-control"
      id="first_name"
      name="first_name"
      placeholder="First Name"
    />
  </div>
  <div class="form-group">
    <label for="birth_date">Birthdate</label>
    <input
      type="date"
      class="form-control"
      id="birth_date"
      name="birth_date"
      placeholder="Birthdate"
    />
  </div>
  <div class="form-group">
    <label for="id">Identification Number</label>
    <input
      type="bigint"
      class="form-control"
      id="id"
      name="id"
      placeholder="ID"
    />
  </div>
  <div class="form-group">
    <label for="id_type">Identification Papers</label>
    <select name="id_type" id="id_type">
      <option value="Identity">Identity</option>
      <option value="Passport">Passport</option>
    </select>
  </div>
  <div class="form-group">
    <label for="id_issue">ID Issue Authority</label>
```

```
    <input
      type="text"
      class="form-control"
      id="id_issue"
      name="id_issue"
      placeholder="ID Issue"
    />
  </div>
  <div class="form-group">
    <label for="phones">Phone Numbers</label>
    <input
      type="bigint"
      class="form-control"
      id="phone1"
      name="phone1"
      placeholder="Phone1"
    />
    <input
      type="bigint"
      class="form-control"
      id="phone2"
      name="phone2"
      placeholder="Phone2"
    />
    <input
      type="bigint"
      class="form-control"
      id="phone3"
      name="phone3"
      placeholder="Phone3"
    />
  </div>
  <div class="form-group">
    <label for="emails">Email Addresses</label>
    <input
      type="text"
      class="form-control"
      id="email1"
      name="email1"
      placeholder="Email1"
    />
    <input
      type="text"
      class="form-control"
      id="email2"
      name="email2"
      placeholder="Email2"
    />
```

```
    <input
      type="text"
      class="form-control"
      id="email3"
      name="email3"
      placeholder="Email3"
    />
  </div>

  <br />
  <div align="center">
    <button type="submit" name="new" value="new" class="btn btn-
primary" style="display:inline-block;">Add Customer</button>
    <button type="submit" name="modify" value="modify" class="btn btn-
primary" style="display:inline-block;margin-
top:0;">Change Phone/Email</button>
    <button type="submit" name="delete" value="delete" class="btn btn-
primary" style="display:inline-block;margin-
top:0;">Delete Customer</button>
  </div>
  <br />
  <br />
</form>
{% endblock %}
```

- Register.html

```
{% extends "base.html" %} {% block title %}Register{% endblock %} {% bl
ock content
%}
<h1 align="center">Register a customer to a provided service</h1>
</br>
</br>
<form method="POST">
  <div class="form-group">
    <label for="NFC_ID">NFC ID</label>
    <input
      type="int"
      class="form-control"
      id="NFC_ID"
      name="NFC_ID"
      placeholder="Existing customer NFC ID"
    />
  </div>
  <div class="form-group">
    <label for="service_name">Service</label>
    <select name="service_name" id="service_name">
```

```
      <option value="Room" {% if service == "Room" %} selected {% endif
 %}>Room</option>
      <option value="Conference Room" {% if service == "Conference Room
" %} selected {% endif %}>Conference Room</option>
      <option value="Gym" {% if service == "Gym" %} selected {% endif %
}>Gym</option>
      <option value="Sauna" {% if service == "Sauna" %} selected {% end
if %}>Sauna</option>
    </select>
  </div>
  <div class="form-group">
    <label for="room">Available rooms</label>
    <select name="room" id="room">
      {% for item in x%}
      <option value="{{item[0]}}" >{{item[0]}}</option>
      {%endfor%}
    </select>
  </div>
  <div class="form-group">
    <label for="room2">Taken rooms</label>
    <select name="room2" id="room2">
      {% for item in y%}
      <option value="{{item[0]}}" >{{item[0]}}</option>
      {%endfor%}
    </select>
  </div>
  <div class="form-group">
    <label for="days">Duration</label>
    <input
      type="int"
      class="form-control"
      id="days"
      name="days"
      placeholder="Duration"
    />
  </div>

  <br />
  <div align="center">
    <button type="submit" name="register" value="register" class="btn b
tn-primary" style="display:inline-block;">Register</button>
    <button type="submit" name="unregister" value="unregister" class="b
tn btn-primary" style="display:inline-block;margin-
top:0;">Unregister</button>
  </div>
  <br />
  <br />
</form>
```

```
{% endblock %}
```

- Serv_rec.html

```
{% extends "base.html" %} {% block title %}Records{% endblock %} {% blo
ck content
%}

<head>
<style>
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
table {
    width: 90%;
}
</style>
</head>
<form method="POST">
  <h3 align="center">Filters</h3>
  <div class="form-group">
    <label for="service_type">Service</label>
    <select name="service_type" id="service_type">
      <option value="All" {% if service == "All" %} selected {% endif %
}>All</option>
      <option value="Bar" {% if service == "Bar" %} selected {% endif %
}>Bar</option>
      <option value="Conference Room" {% if service == "Conference Room
" %} selected {% endif %}>Conference Room</option>
      <option value="Gym" {% if service == "Gym" %} selected {% endif %
}>Gym</option>
      <option value="Hair Salon" {% if service == "Hair Salon" %} selec
ted {% endif %}>Hair Salon</option>
      <option value="Restaurant" {% if service == "Restaurant" %} selec
ted {% endif %}>Restaurant</option>
      <option value="Sauna" {% if service == "Sauna" %} selected {% end
if %}>Sauna</option>
    </select>

  </div>
  <div class="form-group">
    <label for="datefrom">Date From</label>
    <input
      type="date"
      class="form-control"
      id="datefrom"
      name="datefrom"
```

```
      placeholder="Date From"
    />

  </div>
    <div class="form-group">
    <label for="dateto">Date To</label>
    <input
      type="date"
      class="form-control"
      id="dateto"
      name="dateto"
      placeholder="Date To"
    />

  </div>
  <div class="form-group">
    <label for="costfrom">Minimum Cost</label>
    <input
      type="number"
      step="0.01"
      min="0"
      class="form-control"
      id="costfrom"
      name="costfrom"
      placeholder="Minimum Cost"
    />

  </div>
  <div class="form-group">
    <label for="costto">Maximum Cost</label>
    <input
      type="number"
      step="0.01"
      min="0"
      class="form-control"
      id="costto"
      name="costto"
      placeholder="Maximum Cost"
    />

  </div>
  <br />
  <div align="center">
    <button type="submit" class="btn btn-primary">View Records</button>
</div>
</form>
<br />
<br />
```

```
<h3>List of Records of Service Usages</h3>
<div align="center" style="width:100%, overflow: auto;">
  <table>
    <tr>
      <th>NFC ID</th>
      <th>Last Name</th>
      <th>First Name</th>
      <th>SERVICE</th>
      <th>PAYMENT AMOUNT</th>
      <th>Entered at</th>
      <th>Left at</th>
    </tr>
    {% if results != None%}
      {% for result in results %}
      <tr>
      <td>{{result[0]}}</td>
      <td>{{result[1]}}</td>
      <td>{{result[2]}}</td>
      <td>{{result[3]}}</td>
      <td>{{result[4]}}</td>
      <td>{{result[5]}}</td>
      <td>{{result[6]}}</td>
      </tr>
      {% endfor %}
    {% endif %}
  </table>
</div>
{% endblock %}
```

- Statistics.html

```
{% extends "base.html" %} {% block title %}Statistics{% endblock %} {%
block content
    %}

    <head>
    <style>
    table, th, td {
      border: 1px solid black;
      border-collapse: collapse;
    }
    table {
        width: 90%;
    }
    </style>
    </head>
    <form method="POST">
      <h3 align="center">Statistics</h3>
```

```
      <div class="form-group">
        <label for="age_group">Age Group</label>
        <select name="age_group" id="age_group">
          <option value="20-40" {% if y == "20-
40" %} selected {% endif %}>20-40</option>
          <option value="41-60" {% if y == "41-
60" %} selected {% endif %}>41-60</option>
          <option value="61+" {% if y == "61+" %} selected {% endif %}>
61+</option>
        </select>
      </div>

      <div class="form-group">
        <label for="time_period">Time Period</label>
        <select name="time_period" id="time_period">
          <option value="Last Year" {% if t == "Last Year" %} selected
{% endif %}>Last Year</option>
          <option value="Last Month" {% if t == "Last Month" %} selecte
d {% endif %}>Last Month</option>
        </select>
      </div>

      <br />
      <div align="center">
        <button type="submit" name="areas" value="areas" class="btn btn
-primary" style="display:inline-block;">Busiest Areas</button>
        <button type="submit" name="services" value="services" class="b
tn btn-primary" style="display:inline-block;margin-
top:0;">Busiest Services</button>
        <button type="submit" name="popular_services" value="popular_se
rvices" class="btn btn-primary" style="display:inline-block;margin-
top:0;">Most Popular Services</button>
    </div>
    </form>
    <br />
    <br />
    {% if pick == 0%}
      <h3>List of Busiest Shared Areas</h3>
      <div align="center" style="width:100%, overflow: auto;">
        <table>
          <tr>
            <th>Area Name</th>
            <th>No of Visits</th>
          </tr>
          {% if results != None%}
            {% for result in results %}
            <tr>
            <td>{{result[0]}}</td>
```

```
            <td>{{result[1]}}</td>
            </tr>
            {% endfor %}
          {% endif %}
        </table>
      </div>
    {% endif %}
    {% if pick == 1%}
      <h3>List of Busiest Services</h3>
      <div align="center" style="width:100%, overflow: auto;">
        <table>
          <tr>
            <th>Service</th>
            <th>No of Times Used</th>
          </tr>
          {% if results != None%}
            {% for result in results %}
            <tr>
            <td>{{result[0]}}</td>
            <td>{{result[1]}}</td>
            </tr>
            {% endfor %}
          {% endif %}
        </table>
      </div>
    {% endif %}
    {% if pick == 2%}
      <h3>List of Most Popular Services</h3>
      <div align="center" style="width:100%, overflow: auto;">
        <table>
          <tr>
            <th>Service</th>
            <th>No of People Who Used It</th>
          </tr>
          {% if results != None%}
            {% for result in results %}
            <tr>
            <td>{{result[0]}}</td>
            <td>{{result[1]}}</td>
            </tr>
            {% endfor %}
          {% endif %}
        </table>
      </div>
    {% endif %}
    {% endblock %}
```

- Views.html

```
{% extends "base.html" %} {% block title %}Info{% endblock %} {% block
content
%}

<head>
<style>
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
table {
    width: 90%;
}
</style>
</head>
<form method="POST">
  <h3 align="center">Choose the data to be presented</h3>
  <div class="form-group">
    <label for="info">Data</label>
    <select name="info" id="info">
      <option value="Customer Info" {% if data == "Customer Info" %} se
lected {% endif %}>Customer Info</option>
      <option value="Service Sales" {% if data == "Service Sales" %} se
lected {% endif %}>Service Sales</option>
    </select>
  </div>
  <br />
  <div align="center">
    <button type="submit" class="btn btn-primary">View Data</button>
</div>
</form>
<br />
<br />
{% if results != None and data=="Customer Info"%}
  <div align="center" style="width:100%, overflow: auto;">
    <table>
      <tr>
        <th>NFC ID</th>
        <th>Last Name</th>
        <th>First Name</th>
        <th>Birth Date</th>
        <th>ID Number</th>
        <th>ID Type</th>
        <th>ID Issue Authority</th>
        <th>Phone Numbers</th>
        <th>Emails</th>
      </tr>
```

```
        {% for result in results %}
        <tr>
        <td>{{result[0]}}</td>
        <td>{{result[1]}}</td>
        <td>{{result[2]}}</td>
        <td>{{result[3]}}</td>
        <td>{{result[4]}}</td>
        <td>{{result[5]}}</td>
        <td>{{result[6]}}</td>
        <td>{{result[7]}}</td>
        <td>{{result[8]}}</td>
        <td>{{result[9]}}</td>
        </tr>
        {% endfor %}
    </table>
  </div>
{% endif %}
{% if results != None and data=="Service Sales"%}
  <div align="center" style="width:100%, overflow: auto;">
    <table>
      <tr>
        <th>SERVICE ID</th>
        <th>SERVICE NAME</th>
        <th>SERVICE DESCRIPTION</th>
        <th>Total Sales</th>
      </tr>
        {% for result in results %}
        <tr>
        <td>{{result[0]}}</td>
        <td>{{result[1]}}</td>
        <td>{{result[2]}}</td>
        <td>{{result[3]}}</td>
        </tr>
        {% endfor %}
    </table>
  </div>
{% endif %}
{% endblock %}
```

app/ init .py

```
from flask import Flask, render_template, request
from flask_mysqldb import MySQL

mysql = MySQL()


def create_app():

    App = Flask(__name__, static_folder='./frontend/static', template_f
older='./frontend/templates')
    App.config["MYSQL_USER"] = "root"
    App.config["MYSQL_PASSWORD"] = "boftonelly"
    App.config["MYSQL_HOST"] = "localhost"
    App.config["MYSQL_DB"] = "HotelDB"
    App.config['SECRET_KEY'] = 'hjhjhhjhjhj'
    mysql.init_app(App)

    from app.backend import home, register, records, views, statistics,
 covid

    App.register_blueprint(home, url_prefix='/')
    App.register_blueprint(register, url_prefix='/')
    App.register_blueprint(records, url_prefix='/records')
    App.register_blueprint(views, url_prefix='/')
    App.register_blueprint(covid, url_prefix='/')
    App.register_blueprint(statistics, url_prefix='/')

    return App
```

Database Project/db initialization/

- addAccess.py

```
import datetime
import random

def add_access():
    from .connection import mydb, mycursor
    #haveAccess

    j=0
    for i in range(100):

        #arrival = f"2021-5-{15+j//36} {9+4*((j%36)//12)}:00:00"
```

```
        #leave = f"2021-5-{15+j//36+random.randint(3,7)} 13:00:00"

        entry = datetime.datetime.now()-
datetime.timedelta(days=(((44*7)+1)-22*(j//12)))
        arrival = entry.strftime("%Y-%m-%d %H:%M:%S")
        leave = (entry+datetime.timedelta(days=3)).strftime("%Y-%m-
%d %H:%M:%S")

        if i < 20 or (i > 59 and i <80):
            sqlFormula = """INSERT INTO haveAccess (NFC_ID,AREA_ID,star
t_datetime,end_datetime)
                            VALUES ({},{},'{}','{}')""".format(j+1,i+1,
arrival,leave)
            mycursor.execute(sqlFormula)
            mydb.commit()

            sqlFormula = """INSERT INTO haveAccess (NFC_ID,AREA_ID,star
t_datetime,end_datetime)
                            VALUES ({},(SELECT AREA_ID FROM areas WHERE
 area_floor=(SELECT area_floor FROM areas WHERE AREA_ID={}) AND
                            orientation=(SELECT orientation FROM areas
WHERE AREA_ID={}) AND beds=0),'{}','{}')""".format(j+1,i+1,i+1,arrival,
leave)
            mycursor.execute(sqlFormula)
            mydb.commit()

            if j%4==0:
                for l in range(446,450):
                    sqlFormula = """INSERT INTO haveAccess (NFC_ID,AREA
_ID,start_datetime,end_datetime)
                                    VALUES ({},{},'{}','{}')""".format(
j+1,l,arrival,leave)
                    mycursor.execute(sqlFormula)
                    mydb.commit()

            if j%9==0:
                for l in range(450,460):
                    sqlFormula = """INSERT INTO haveAccess (NFC_ID,AREA
_ID,start_datetime,end_datetime)
                                    VALUES ({},{},'{}','{}')""".format(
j+1,l,arrival,leave)
                    mycursor.execute(sqlFormula)
                    mydb.commit()

            if j%15==0:
                for l in range(436,446):
                    sqlFormula = """INSERT INTO haveAccess (NFC_ID,AREA
_ID,start_datetime,end_datetime)
```

```
                                    VALUES ({},{},'{}','{}')""".format(
j+1,l,arrival,leave)
                    mycursor.execute(sqlFormula)
                    mydb.commit()
            j = j+1
        elif (i >= 20 and i < 40) or (i >=80 and i < 100):
            for k in range(2):
                sqlFormula = """INSERT INTO haveAccess (NFC_ID,AREA_ID,
start_datetime,end_datetime)
                                    VALUES ({},{},'{}','{}')""".format(j+1,
i+1,arrival,leave)
                mycursor.execute(sqlFormula)
                mydb.commit()

                sqlFormula = """INSERT INTO haveAccess (NFC_ID,AREA_ID,
start_datetime,end_datetime)
                                    VALUES ({},(SELECT AREA_ID FROM areas W
HERE area_floor=(SELECT area_floor FROM areas WHERE AREA_ID={}) AND
                                    orientation=(SELECT orientation FROM ar
eas WHERE AREA_ID={}) AND beds=0),'{}','{}')""".format(j+1,i+1,i+1,arri
val,leave)
                mycursor.execute(sqlFormula)
                mydb.commit()

                if j%4==0:
                    for l in range(446,450):
                        sqlFormula = """INSERT INTO haveAccess (NFC_ID,
AREA_ID,start_datetime,end_datetime)
                                            VALUES ({},{},'{}','{}')""".for
mat(j+1,l,arrival,leave)
                        mycursor.execute(sqlFormula)
                        mydb.commit()

                if j%9==0:
                    for l in range(450,460):
                        sqlFormula = """INSERT INTO haveAccess (NFC_ID,
AREA_ID,start_datetime,end_datetime)
                                            VALUES ({},{},'{}','{}')""".for
mat(j+1,l,arrival,leave)
                        mycursor.execute(sqlFormula)
                        mydb.commit()

                if j%15==0:
                    for l in range(436,446):
                        sqlFormula = """INSERT INTO haveAccess (NFC_ID,
AREA_ID,start_datetime,end_datetime)
                                            VALUES ({},{},'{}','{}')""".for
mat(j+1,l,arrival,leave)
```

```
                        mycursor.execute(sqlFormula)
                        mydb.commit()
                j = j+1
        else:
            for k in range(3):
                sqlFormula = """INSERT INTO haveAccess (NFC_ID,AREA_ID,
start_datetime,end_datetime)
                                VALUES ({},{},'{}','{}')""".format(j+1,
i+1,arrival,leave)
                mycursor.execute(sqlFormula)
                mydb.commit()

                sqlFormula = """INSERT INTO haveAccess (NFC_ID,AREA_ID,
start_datetime,end_datetime)
                                VALUES ({},(SELECT AREA_ID FROM areas W
HERE (area_floor=(SELECT area_floor FROM areas WHERE AREA_ID={}) AND
                                orientation=(SELECT orientation FROM ar
eas WHERE AREA_ID={})) AND beds=0),'{}','{}')""".format(j+1,i+1,i+1,arr
ival,leave)
                mycursor.execute(sqlFormula)
                mydb.commit()

                if j%4==0:
                    for l in range(446,450):
                        sqlFormula = """INSERT INTO haveAccess (NFC_ID,
AREA_ID,start_datetime,end_datetime)
                                        VALUES ({},{},'{}','{}')""".for
mat(j+1,l,arrival,leave)
                        mycursor.execute(sqlFormula)
                        mydb.commit()

                if j%9==0:
                    for l in range(450,460):
                        sqlFormula = """INSERT INTO haveAccess (NFC_ID,
AREA_ID,start_datetime,end_datetime)
                                        VALUES ({},{},'{}','{}')""".for
mat(j+1,l,arrival,leave)
                        mycursor.execute(sqlFormula)
                        mydb.commit()

                if j%15==0:
                    for l in range(436,446):
                        sqlFormula = """INSERT INTO haveAccess (NFC_ID,
AREA_ID,start_datetime,end_datetime)
                                        VALUES ({},{},'{}','{}')""".for
mat(j+1,l,arrival,leave)
                        mycursor.execute(sqlFormula)
                        mydb.commit()
```

```
                j = j+1
```

- addAreas.py

```
def add_areas():
    from .connection import mydb, mycursor

    orientations = ['N', 'W', 'E', 'S', 'NE', 'NW', 'SE', 'SW']

    for i in range(400):
        area_name = f"Room{i+1}"
        if i%60 < 20:
            beds = 1
        elif i%60 < 40:
            beds = 2
        else:
            beds = 3
        area_floor = ((i%20)//4) + 1
        orientation = orientations[(i%4)]
        info = f"Room{i+1} with {beds} beds on floor {area_floor}"
        sqlFormula = """INSERT INTO areas (area_name,beds,area_floor,or
ientation,info)
                        VALUES ('{}',{},{},'{}','{}')""".format(area_na
me,beds,area_floor,orientation,info)
        mycursor.execute(sqlFormula)
        mydb.commit()


    for i in range(20):
        area_name = f"Hall{i+1}"
        beds = 0
        area_floor = i//4 + 1
        orientation = orientations[i%4]
        info = f"Hall{i+1} on floor {area_floor} with {orientation} ori
entation"
        sqlFormula = """INSERT INTO areas (area_name,beds,area_floor,or
ientation,info)
                        VALUES ('{}',{},{},'{}','{}')""".format(area_na
me,beds,area_floor,orientation,info)
        mycursor.execute(sqlFormula)
        mydb.commit()


    for i in range(5):
        area_name = f"Elevator{i+1}"
        beds = 0
        area_floor = 0
        sqlFormula = """INSERT INTO areas (area_name,beds,area_floor)
```

```
                        VALUES ('{}',{},{})""".format(area_name,beds,ar
ea_floor)
        mycursor.execute(sqlFormula)
        mydb.commit()


    for i in range(6):
        area_name = f"Bar{i+1}"
        beds = 0
        area_floor = 0
        orientation = orientations[i]
        info = f"Bar{i+1} on ground floor with {orientation} orientatio
n"
        sqlFormula = """INSERT INTO areas (area_name,beds,area_floor,or
ientation,info)
                        VALUES ('{}',{},{},'{}','{}')""".format(area_na
me,beds,area_floor,orientation,info)
        mycursor.execute(sqlFormula)
        mydb.commit()


    for i in range(4):
        area_name = f"Restaurant{i+1}"
        beds = 0
        area_floor = 0
        orientation = orientations[i]
        info = f"Restaurant{i+1} on ground floor with {orientation} ori
entation"
        sqlFormula = """INSERT INTO areas (area_name,beds,area_floor,or
ientation,info)
                        VALUES ('{}',{},{},'{}','{}')""".format(area_na
me,beds,area_floor,orientation,info)
        mycursor.execute(sqlFormula)
        mydb.commit()


    for i in range(10):
        area_name = f"Conference Room{i+1}"
        beds = 0
        area_floor = 0
        orientation = orientations[i%8]
        info = f"Conference Room{i+1} on ground floor with {orientation
} orientation"
        sqlFormula = """INSERT INTO areas (area_name,beds,area_floor,or
ientation,info)
                        VALUES ('{}',{},{},'{}','{}')""".format(area_na
me,beds,area_floor,orientation,info)
        mycursor.execute(sqlFormula)
        mydb.commit()
```

```
    for i in range(4):
        area_name = f"Gym{i+1}"
        beds = 0
        area_floor = 0
        orientation = orientations[7-i]
        info = f"Gym{i+1} on ground floor with {orientation} orientatio
n"
        sqlFormula = """INSERT INTO areas (area_name,beds,area_floor,or
ientation,info)
                        VALUES ('{}',{},{},'{}','{}')""".format(area_na
me,beds,area_floor,orientation,info)
        mycursor.execute(sqlFormula)
        mydb.commit()


    for i in range(10):
        area_name = f"Sauna{i+1}"
        beds = 0
        area_floor = 0
        orientation = orientations[i%8]
        info = f"Sauna{i+1} on ground floor with {orientation} orientat
ion"
        sqlFormula = """INSERT INTO areas (area_name,beds,area_floor,or
ientation,info)
                        VALUES ('{}',{},{},'{}','{}')""".format(area_na
me,beds,area_floor,orientation,info)
        mycursor.execute(sqlFormula)
        mydb.commit()


    sqlFormula = """INSERT INTO areas (area_name,beds,area_floor,orient
ation,info)
                    VALUES ('{}',{},{},'{}','{}')""".format("Hair Salon
",0,0,'W',"Hair Salon on ground floor with W orientation")
    mycursor.execute(sqlFormula)
    mydb.commit()

    sqlFormula = """INSERT INTO areas (area_name,beds,area_floor,orient
ation,info)
                    VALUES ('{}',{},{},'{}','{}')""".format("Lobby",0,0
,'S',"Lobby on ground floor with S orientation")
    mycursor.execute(sqlFormula)
    mydb.commit()
```

- addCustomers.py

```
import pandas as pd
```

```
import numpy as np

def add_customers():
    from .connection import mydb, mycursor

    People = pd.read_csv("./db_initialization/RandomPeople.csv")

    for i in range(len(People)-20):
        first_name = People['firstname'][i].replace("'","")
        last_name = People['lastname'][i].replace("'","")
        id = People['ID_Number'][i]
        id_type = People['ID_Type'][i]
        id_issue = People['ID_Issue'][i]
        birth_date = People['birthdate'][i]

        sqlFormula = """INSERT INTO customers (first_name,last_name,bir
th_date,id,id_type,id_issue)
                        VALUES ('{}','{}','{}',{},'{}','{}')""".format(
first_name,last_name,birth_date,id,id_type,id_issue)
        mycursor.execute(sqlFormula)
        mydb.commit()
        sqlFormula = """INSERT INTO customer_phones (NFC_ID,phone_numbe
r)
                        VALUES ({},{})""".format(i+1,People['phone'][i]
)
        mycursor.execute(sqlFormula)
        mydb.commit()
        if i % 5 == 0:
            sqlFormula = """INSERT INTO customer_phones (NFC_ID,phone_n
umber)
                    VALUES ({},{})""".format(i+1,People['phone2'][i])
            mycursor.execute(sqlFormula)
            mydb.commit()


        sqlFormula = """INSERT INTO customer_emails (NFC_ID,email_addre
ss)
                        VALUES ({},'{}')""".format(i+1,People['email'][
i].replace("'",""))
        mycursor.execute(sqlFormula)
        mydb.commit()
        if i % 7 == 0:
            sqlFormula = """INSERT INTO customer_emails (NFC_ID,email_a
ddress)
                    VALUES ({},'{}')""".format(i+1,People['email2'][i].
replace("'",""))
            mycursor.execute(sqlFormula)
            mydb.commit()
```

- addProvide.py

```
import numpy as np
import datetime
import random

def add_provide():
    from .connection import mydb, mycursor
    #Provide

    for i in range(400):
        sqlFormula = """INSERT INTO provide (AREA_ID, SERVICE_ID)
                        VALUES ({},{})""".format(i+1,1)
        mycursor.execute(sqlFormula)
        mydb.commit()

    for i in range(446,450):
        sqlFormula = """INSERT INTO provide (AREA_ID, SERVICE_ID)
                        VALUES ({},{})""".format(i,2)
        mycursor.execute(sqlFormula)
        mydb.commit()

    for i in range(436,446):
        sqlFormula = """INSERT INTO provide (AREA_ID, SERVICE_ID)
                        VALUES ({},{})""".format(i,3)
        mycursor.execute(sqlFormula)
        mydb.commit()

    for i in range(450,460):
        sqlFormula = """INSERT INTO provide (AREA_ID, SERVICE_ID)
                        VALUES ({},{})""".format(i,4)
        mycursor.execute(sqlFormula)
        mydb.commit()

    sqlFormula = """INSERT INTO provide (AREA_ID, SERVICE_ID)
                        VALUES ({},{})""".format(460,5)
    mycursor.execute(sqlFormula)
    mydb.commit()

    for i in range(426,432):
        sqlFormula = """INSERT INTO provide (AREA_ID, SERVICE_ID)
                        VALUES ({},{})""".format(i,6)
        mycursor.execute(sqlFormula)
        mydb.commit()

    for i in range(432,436):
        sqlFormula = """INSERT INTO provide (AREA_ID, SERVICE_ID)
```

```
                        VALUES ({},{})""".format(i,7)
        mycursor.execute(sqlFormula)
        mydb.commit()
```

- addRegister.py

```
import numpy as np
import datetime
import random

def add_register():
    from .connection import mydb, mycursor

    #Registers
    for i in range(180):

        #arrival = f"2021-5-{15+i//36} {9+4*((i%36)//12)}:00:00"
        arrival = (datetime.datetime.now()-
datetime.timedelta(days=(44*7+1)-22*(i//12))).strftime("%Y-%m-
%d %H:%M:%S")

        sqlFormula = """INSERT INTO register (NFC_ID,SERVICE_ID,date_ti
me)
                        VALUES ({},{},'{}')""".format(i+1,1,arrival)
        mycursor.execute(sqlFormula)
        mydb.commit()

        sqlFormula = """INSERT INTO useService (NFC_ID,SERVICE_ID,date_
time,cost)
                        VALUES ({},{},'{}',{})""".format(i+1,1,arrival,
90)
        mycursor.execute(sqlFormula)
        mydb.commit()

        if i%4==0:
            sqlFormula = """INSERT INTO register (NFC_ID,SERVICE_ID,dat
e_time)
                            VALUES ({},{},'{}')""".format(i+1,2,arrival
)
            mycursor.execute(sqlFormula)
            mydb.commit()

        if i%9==0:
            sqlFormula = """INSERT INTO register (NFC_ID,SERVICE_ID,dat
e_time)
                            VALUES ({},{},'{}')""".format(i+1,4,arrival
)
            mycursor.execute(sqlFormula)
```

```
            mydb.commit()

        if i%15==0:
            sqlFormula = """INSERT INTO register (NFC_ID,SERVICE_ID,dat
e_time)
                            VALUES ({},{},'{}')""".format(i+1,3,arrival
)
            mycursor.execute(sqlFormula)
            mydb.commit()
```

- addServices.py

```
def add_services():
    from .connection import mydb, mycursor


    sqlFormula = """INSERT INTO services (category,service_description,
register_required)
                    VALUES ('{}','{}',{})""".format("Room","Renting Roo
m for a given period of time",True)
    mycursor.execute(sqlFormula)
    mydb.commit()

    sqlFormula = """INSERT INTO services_With_Register (SERVICE_ID)
                    VALUES ({})""".format(1)
    mycursor.execute(sqlFormula)
    mydb.commit()


    sqlFormula = """INSERT INTO services (category,service_description,
register_required)
                    VALUES ('{}','{}',{})""".format("Gym","Gym subscrip
tion for a given period of time",True)
    mycursor.execute(sqlFormula)
    mydb.commit()

    sqlFormula = """INSERT INTO services_With_Register (SERVICE_ID)
                    VALUES ({})""".format(2)
    mycursor.execute(sqlFormula)
    mydb.commit()


    sqlFormula = """INSERT INTO services (category,service_description,
register_required)
                    VALUES ('{}','{}',{})""".format("Conference Room","
Conference Room renting for a given period of time",True)
    mycursor.execute(sqlFormula)
    mydb.commit()
```

```
    sqlFormula = """INSERT INTO services_With_Register (SERVICE_ID)
                    VALUES ({})""".format(3)
    mycursor.execute(sqlFormula)
    mydb.commit()


    sqlFormula = """INSERT INTO services (category,service_description,
register_required)
                    VALUES ('{}','{}',{})""".format("Sauna","Sauna subs
cription for a given period of time",True)
    mycursor.execute(sqlFormula)
    mydb.commit()

    sqlFormula = """INSERT INTO services_With_Register (SERVICE_ID)
                    VALUES ({})""".format(4)
    mycursor.execute(sqlFormula)
    mydb.commit()


    sqlFormula = """INSERT INTO services (category,service_description,
register_required)
                    VALUES ('{}','{}',{})""".format("Hair Salon","Hair
Styling at our Hair Salon",False)
    mycursor.execute(sqlFormula)
    mydb.commit()

    sqlFormula = """INSERT INTO services_No_Register (SERVICE_ID)
                    VALUES ({})""".format(5)
    mycursor.execute(sqlFormula)
    mydb.commit()


    sqlFormula = """INSERT INTO services (category,service_description,
register_required)
                    VALUES ('{}','{}',{})""".format("Bar","Drinks at th
e Bar",False)
    mycursor.execute(sqlFormula)
    mydb.commit()

    sqlFormula = """INSERT INTO services_No_Register (SERVICE_ID)
                    VALUES ({})""".format(6)
    mycursor.execute(sqlFormula)
    mydb.commit()


    sqlFormula = """INSERT INTO services (category,service_description,
register_required)
```

```
                    VALUES ('{}','{}',{})""".format("Restaurant","Dinin
g at the Restaurant",False)
    mycursor.execute(sqlFormula)
    mydb.commit()

    sqlFormula = """INSERT INTO services_No_Register (SERVICE_ID)
                    VALUES ({})""".format(7)
    mycursor.execute(sqlFormula)
    mydb.commit()
```

- addVisitAndUse.py

```
import numpy as np
import datetime
import random

def add_visit_use():
    from .connection import mydb, mycursor
    # Visit

    for i in range(180):

        entry = datetime.datetime.now()-
datetime.timedelta(days=(44*7+1)-22*(i//12))
        arrival = entry.strftime("%Y-%m-%d %H:%M:%S")

        exit = entry+datetime.timedelta(days=3)
        leave = (entry+datetime.timedelta(days=3)).strftime("%Y-%m-
%d %H:%M:%S")

        start = entry
        finish = min(exit, datetime.datetime.now())

        while start < finish:

            #endtask = start + datetime.timedelta(minutes=random.randin
t(10,120))

            x = random.randint(1,100)
            if x <= 60:
                area = np.random.choice(['Hall', 'Elevator', 'Lobby'])
                endtask = start + datetime.timedelta(minutes=random.ran
dint(2,8))
                if endtask < finish:
                    if area=='Hall':
                        sqlFormula = """INSERT INTO visit (NFC_ID,AREA_
ID,enter_datetime,leave_datetime)
```

```
                                        VALUES ({},(SELECT AREA_ID FROM
 haveAccess WHERE (NFC_ID={} AND AREA_ID>400 AND AREA_ID<421)),'{}','{}
')""".format(i+1,i+1,start.strftime("%Y-%m-
%d %H:%M:%S"),endtask.strftime("%Y-%m-%d %H:%M:%S"))
                        mycursor.execute(sqlFormula)
                        mydb.commit()
                        start = endtask

                    elif area=='Elevator':
                        sqlFormula = """INSERT INTO visit (NFC_ID,AREA_
ID,enter_datetime,leave_datetime)
                                        VALUES ({},{},'{}','{}')""".for
mat(i+1,random.randint(421,425),start.strftime("%Y-%m-
%d %H:%M:%S"),endtask.strftime("%Y-%m-%d %H:%M:%S"))
                        mycursor.execute(sqlFormula)
                        mydb.commit()
                        start = endtask
                    else:
                        sqlFormula = """INSERT INTO visit (NFC_ID,AREA_
ID,enter_datetime,leave_datetime)
                                        VALUES ({},{},'{}','{}')""".for
mat(i+1,461,start.strftime("%Y-%m-%d %H:%M:%S"),endtask.strftime("%Y-
%m-%d %H:%M:%S"))
                        mycursor.execute(sqlFormula)
                        mydb.commit()
                        start = endtask + datetime.timedelta(hours=rand
om.randint(6,12))

            elif x>85:
                area = np.random.choice(['Bar', 'Bar', 'Bar', 'Restaura
nt', 'Restaurant', 'Restaurant','Restaurant', 'HairSalon'])
                endtask = start + datetime.timedelta(hours=random.randi
nt(1,3))
                if endtask < finish:
                    if area=='Bar':
                        sqlFormula = """INSERT INTO visit (NFC_ID,AREA_
ID,enter_datetime,leave_datetime)
                                        VALUES ({},{},'{}','{}')""".for
mat(i+1,random.randint(426,431),start.strftime("%Y-%m-
%d %H:%M:%S"),endtask.strftime("%Y-%m-%d %H:%M:%S"))
                        mycursor.execute(sqlFormula)
                        mydb.commit()

                        sqlFormula = """INSERT INTO useService (NFC_ID,
SERVICE_ID,date_time,cost)
                                        VALUES ({},{},'{}',{})""".forma
t(i+1,6,(endtask-
```

```
datetime.timedelta(minutes=random.randint(5,15))).strftime("%Y-%m-
%d %H:%M:%S"),round(random.uniform(8,50),2))
                        mycursor.execute(sqlFormula)
                        mydb.commit()

                    elif area=='Restaurant':
                        sqlFormula = """INSERT INTO visit (NFC_ID,AREA_
ID,enter_datetime,leave_datetime)
                                        VALUES ({},{},'{}','{}')""".for
mat(i+1,random.randint(432,435),start.strftime("%Y-%m-
%d %H:%M:%S"),endtask.strftime("%Y-%m-%d %H:%M:%S"))
                        mycursor.execute(sqlFormula)
                        mydb.commit()

                        sqlFormula = """INSERT INTO useService (NFC_ID,
SERVICE_ID,date_time,cost)
                                        VALUES ({},{},'{}',{})""".forma
t(i+1,7,(endtask-
datetime.timedelta(minutes=random.randint(5,15))).strftime("%Y-%m-
%d %H:%M:%S"),round(random.uniform(20,45),2))
                        mycursor.execute(sqlFormula)
                        mydb.commit()

                    else:
                        sqlFormula = """INSERT INTO visit (NFC_ID,AREA_
ID,enter_datetime,leave_datetime)
                                        VALUES ({},{},'{}','{}')""".for
mat(i+1,460,start.strftime("%Y-%m-%d %H:%M:%S"),endtask.strftime("%Y-
%m-%d %H:%M:%S"))
                        mycursor.execute(sqlFormula)
                        mydb.commit()

                        sqlFormula = """INSERT INTO useService (NFC_ID,
SERVICE_ID,date_time,cost)
                                        VALUES ({},{},'{}',{})""".forma
t(i+1,5,(endtask-
datetime.timedelta(minutes=random.randint(5,15))).strftime("%Y-%m-
%d %H:%M:%S"),round(random.uniform(15,60),2))
                        mycursor.execute(sqlFormula)
                        mydb.commit()

                start = endtask
            else:
                sqlFormula = """SELECT AREA_ID FROM haveAccess WHERE NF
C_ID={} AND (AREA_ID<401 OR AREA_ID>420)""".format(i+1)
                mycursor.execute(sqlFormula)

                rooms = [item[0] for item in mycursor.fetchall()]
```

```
                area_id = np.random.choice(rooms)
                if area_id<401:
                    endtask = start + datetime.timedelta(hours=random.r
andint(4,9))
                else:
                    endtask = start + datetime.timedelta(hours=random.r
andint(1,4))
                if endtask<finish:
                    sqlFormula = """INSERT INTO visit (NFC_ID,AREA_ID,e
nter_datetime,leave_datetime)
                                    VALUES ({},{},'{}','{}')""".format(
i+1,area_id,start.strftime("%Y-%m-%d %H:%M:%S"),endtask.strftime("%Y-
%m-%d %H:%M:%S"))
                    mycursor.execute(sqlFormula)
                    mydb.commit()

                if area_id in range(446,450):
                    sqlFormula = """INSERT INTO useService (NFC_ID,SERV
ICE_ID,date_time,cost)
                                    VALUES ({},{},'{}',{})""".format(i+
1,2,(endtask-
datetime.timedelta(minutes=random.randint(5,15))).strftime("%Y-%m-
%d %H:%M:%S"),round(random.uniform(6,15),2))
                    mycursor.execute(sqlFormula)
                    mydb.commit()

                elif area_id in range(436,446):
                    sqlFormula = """INSERT INTO useService (NFC_ID,SERV
ICE_ID,date_time,cost)
                                    VALUES ({},{},'{}',{})""".format(i+
1,3,(endtask-
datetime.timedelta(minutes=random.randint(5,15))).strftime("%Y-%m-
%d %H:%M:%S"),round(random.uniform(15,30),2))
                    mycursor.execute(sqlFormula)
                    mydb.commit()

                elif area_id in range(450,460):
                    sqlFormula = """INSERT INTO useService (NFC_ID,SERV
ICE_ID,date_time,cost)
                                    VALUES ({},{},'{}',{})""".format(i+
1,4,(endtask-
datetime.timedelta(minutes=random.randint(5,15))).strftime("%Y-%m-
%d %H:%M:%S"),round(random.uniform(5,15),2))
                    mycursor.execute(sqlFormula)
                    mydb.commit()

                start = endtask
```

- connection.py

```
import mysql.connector

mydb = mysql.connector.connect(
        host = "localhost",
        user = "root",
        passwd = "boftonelly",
        database = "HotelDB"
    )

mycursor = mydb.cursor()
```

Database_Project/

- __init__.py

```
from .main_app import mysql
```

- Main_app.py

```
import os
from app import create_app

app = create_app()

if __name__ == '__main__':
    app.run(host=os.getenv("IP", "localhost"), port=int(os.getenv("PORT
", 8765)), debug=True)
```

- Main_db.py

```
from db_initialization.addCustomers import add_customers
from db_initialization.addAreas import add_areas
from db_initialization.addServices import add_services
from db_initialization.addRegister import add_register
from db_initialization.addProvide import add_provide
from db_initialization.addAccess import add_access
from db_initialization.addVisitAndUse import add_visit_use

add_customers()
add_areas()
add_services()
add_register()
add_provide()
add_access()
add_visit_use()
```