A Project Report on

PG ACCOMODATION SERVICE

ROOMIES

DEVELOPED BY:

IT057 KAKADIYA HARSHAL
IT059 KANTARIA JAY

Guided By
Internal Guide:
Prof. Archana N. Vyas

Department of Information Technology

Faculty of Technology

DDU

Department of Information Technology Faculty of Technology,

Dharmsinh Desai University College Road, Nadiad-387001

October-2022

DHARMSINH DESAI UNIVERSITY
NADIAD-387001, GUJARAT

CERTIFICATE

This is to certify that the project entitled "**PG ACCOMODATION SERVICE**" is a bonafide report of the work carried out by

1) KAKADIYA HARSHAL BABULAL Student id no.:20ITUOS064
2) KANTARIYA JAY MANISHBHAI Student id no.:20ITUOS097

of Department of Information Technology, semester V, under the guidance and supervision for the subject Database Management System. They were involved in Project training during the academic year 2022-2023.

Prof. Archana N. Vyas

Project Guide, Department of Information Technology,
Faculty of Technology,

Dharmsinh Desai University, Nadiad
Date: 05/10/2022

Prof. Vipul Dabhi

Head, Department of Information Technology

COMMENDATION

We would like to express our heartfelt gratitude to everyone who contributed to the successful completion of our project "PG ACCOMODATION SERVICE"

We are tremendously grateful to have gotten all of the guidance and assistance needed for this project's development and successful finish, and we are lucky to have done so concurrently with the project's completion.

We owe **Prof. Archana N. Vyas**, our project advisor, a debt of gratitude for taking an interest in our project work and guiding us through it till it was finished by providing all of the necessary support for developing a strong Database System.

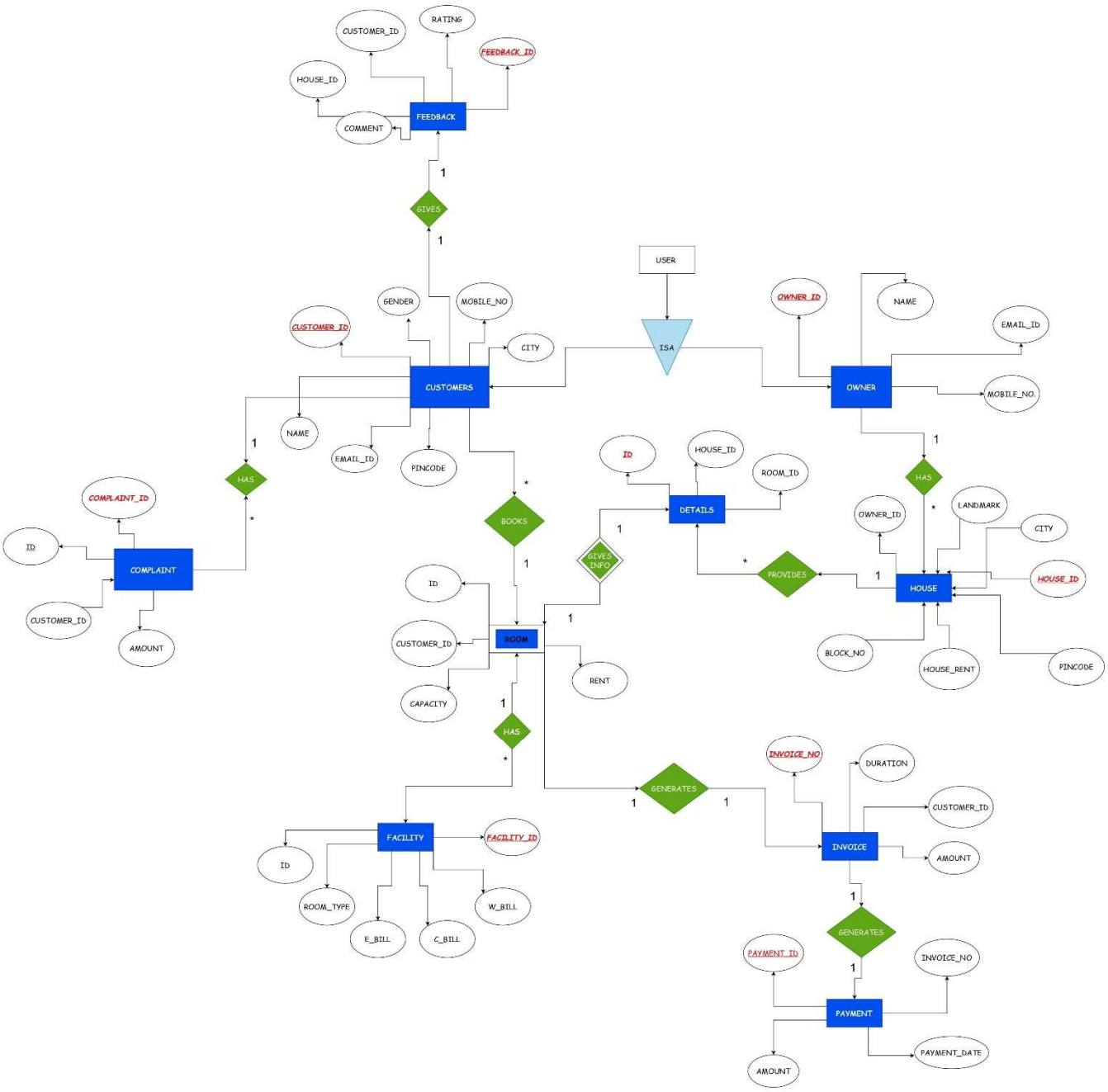
Finally, we want to thank all of our friends supportive faculty.

INDEX

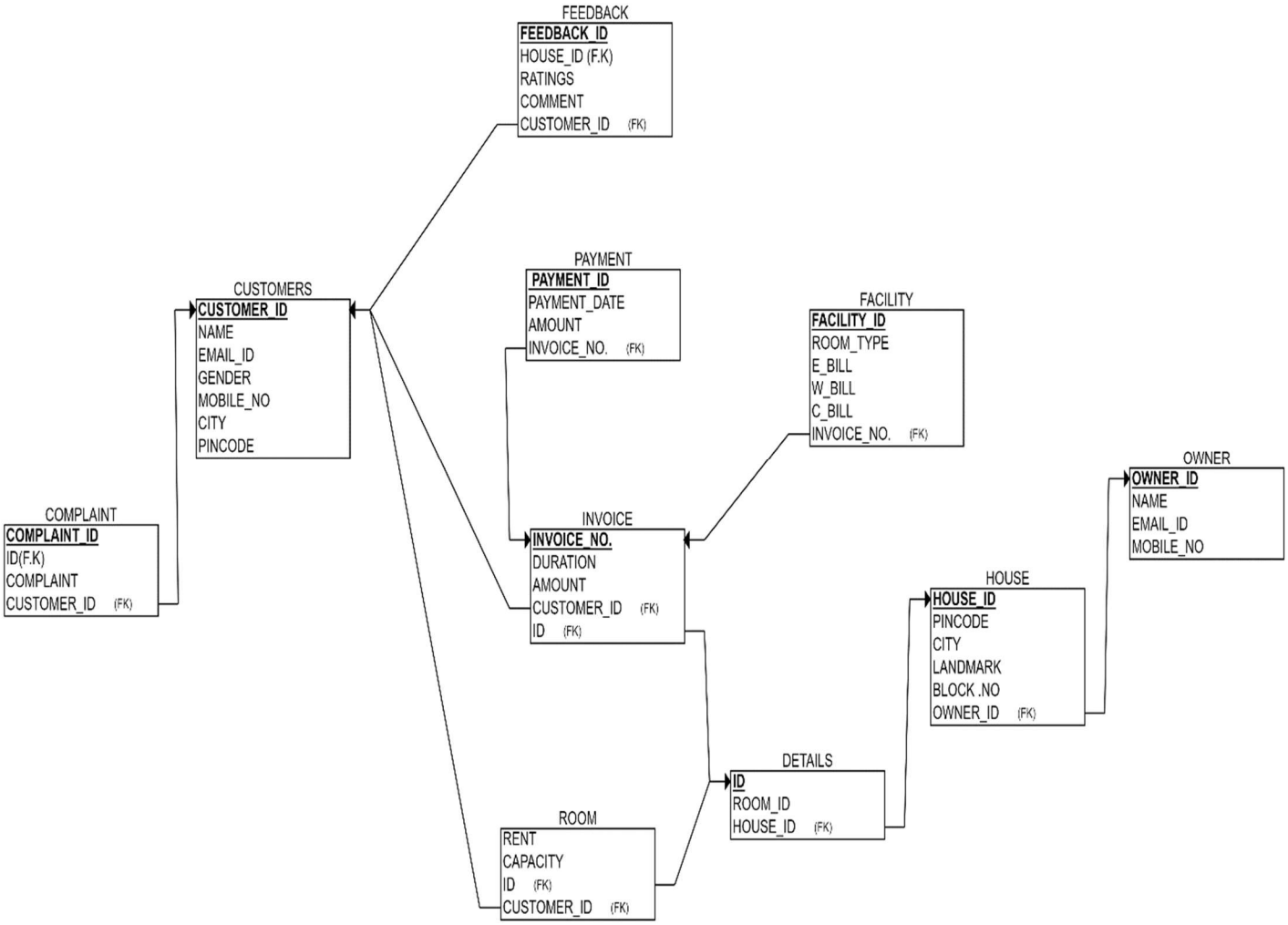
(1) SYSTEM OVERVIEW

- The Project we designed is on the PG ACCOMODATION SYSTEM.
- Main motive to create this project is to reduce searching of PG's.
- As a student, accommodations provide the biggest challenge, thus this idea was conceived to make things easier.
- This project's benefit is that it saves time and money on Pg searches.
- This project is designed such that students can see the details of the PG's that are available in particular city.
- Owner of the Pg register their house with every detail that's required.
- Students can see details like owner name, PG location, Address, number of room, facility provided, rent, ac/non-ac.
- If student is interested, invoice is generated and then payment is to be made. After the living time is about to over, student can repay for extending living period.

(2) ENTITY RELATIONAL MODEL



(3) RELATIONAL SCHEMA



(4) DATA DICTIONARY:

Customers :-

```
postgres=# \d customers
                    Table "public.customers"
   Column    |         Type          | Collation | Nullable | Default
-------------+-----------------------+-----------+----------+---------
 customer_id | character varying(20) |           | not null |
 name        | character varying(10) |           | not null |
 email_id    | character varying(30) |           |          |
 gender      | character varying(9)  |           |          |
 mobile_no   | numeric(10,0)         |           |          |
 city        | character varying     |           | not null |
 pincode     | numeric(6,0)          |           | not null |
Indexes:
    "customers_pkey" PRIMARY KEY, btree (customer_id)
Check constraints:
    "customers_gender_check" CHECK (gender::text = ANY (ARRAY['MALE'::character varying, 'FEMALE'::character varying]::t
ext[]))
Referenced by:
    TABLE "complaint" CONSTRAINT "complaint_customer_id_fkey" FOREIGN KEY (customer_id) REFERENCES customers(customer_id
)
    TABLE "feedback" CONSTRAINT "feedback_customer_id_fkey" FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
    TABLE "invoice" CONSTRAINT "invoice_customer_id_fkey" FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
    TABLE "room" CONSTRAINT "room_customer_id_fkey" FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
```

Owner :-

```
postgres=# \d owner
                      Table "public.owner"
  Column   |         Type          | Collation | Nullable | Default
-----------+-----------------------+-----------+----------+---------
 owner_id  | character varying(10) |           | not null |
 name      | character varying(10) |           | not null |
 email_id  | character varying(30) |           |          |
 mobile_no | numeric(10,0)         |           |          |
Indexes:
    "owner_pkey" PRIMARY KEY, btree (owner_id)
Referenced by:
    TABLE "house" CONSTRAINT "house_owner_id_fkey" FOREIGN KEY (owner_id) REFERENCES owner(owner_id)
```

House :-

```
postgres=# \d house
                    Table "public.house"
  Column  |         Type          | Collation | Nullable | Default
----------+-----------------------+-----------+----------+---------
 house_id | character varying(10) |           | not null |
 owner_id | character varying(10) |           | not null |
 pincode  | numeric(6,0)          |           | not null |
 city     | character varying(10) |           | not null |
 landmark | character varying(30) |           |          |
 block_no | integer               |           |          |
Indexes:
    "house_pkey" PRIMARY KEY, btree (house_id)
Foreign-key constraints:
    "house_owner_id_fkey" FOREIGN KEY (owner_id) REFERENCES owner(owner_id)
Referenced by:
    TABLE "details" CONSTRAINT "details_house_id_fkey" FOREIGN KEY (house_id) REFERENCES house(house_id)
    TABLE "feedback" CONSTRAINT "feedback_house_id_fkey" FOREIGN KEY (house_id) REFERENCES house(house_id)
```

Room :-

```
postgres=# \d room
                      Table "public.room"
   Column    |         Type          | Collation | Nullable | Default
-------------+-----------------------+-----------+----------+---------
 id          | integer               |           | not null |
 customer_id | character varying(10) |           | not null |
 rent        | integer               |           |          |
 capacity    | integer               |           |          |
Foreign-key constraints:
    "room_customer_id_fkey" FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
    "room_id_fkey" FOREIGN KEY (id) REFERENCES details(id)
```

Details :-

```
postgres=# \d details
                    Table "public.details"
  Column  |         Type          | Collation | Nullable | Default
----------+-----------------------+-----------+----------+---------
 id       | integer               |           | not null |
 room_id  | character varying(20) |           | not null |
 house_id | character varying(20) |           | not null |
Indexes:
    "details_pkey" PRIMARY KEY, btree (id)
Foreign-key constraints:
    "details_house_id_fkey" FOREIGN KEY (house_id) REFERENCES house(house_id)
Referenced by:
    TABLE "complaint" CONSTRAINT "complaint_id_fkey" FOREIGN KEY (id) REFERENCES details(id)
    TABLE "facility" CONSTRAINT "facility_id_fkey" FOREIGN KEY (id) REFERENCES details(id)
    TABLE "invoice" CONSTRAINT "invoice_id_fkey" FOREIGN KEY (id) REFERENCES details(id)
    TABLE "room" CONSTRAINT "room_id_fkey" FOREIGN KEY (id) REFERENCES details(id)
```

Invoice :-

```
postgres=# \d invoice
                      Table "public.invoice"
   Column    |         Type          | Collation | Nullable | Default
-------------+-----------------------+-----------+----------+---------
 invoice_no  | character varying(10) |           | not null |
 duration    | integer               |           | not null |
 id          | integer               |           | not null |
 customer_id | character varying(10) |           | not null |
 amount      | integer               |           | not null |
Indexes:
    "invoice_pkey" PRIMARY KEY, btree (invoice_no)
Foreign-key constraints:
    "invoice_customer_id_fkey" FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
    "invoice_id_fkey" FOREIGN KEY (id) REFERENCES details(id)
Referenced by:
    TABLE "payment" CONSTRAINT "payment_invoice_no_fkey" FOREIGN KEY (invoice_no) REFERENCES invoice(invoice_no)
```

Payment :-

```
postgres=# \d payment
                      Table "public.payment"
    Column    |         Type          | Collation | Nullable | Default
--------------+-----------------------+-----------+----------+---------
 payment_id   | character varying(10) |           | not null |
 invoice_no   | character varying(10) |           | not null |
 payment_date | date                  |           |          |
 amount       | integer               |           | not null |
Indexes:
    "payment_pkey" PRIMARY KEY, btree (payment_id)
Foreign-key constraints:
    "payment_invoice_no_fkey" FOREIGN KEY (invoice_no) REFERENCES invoice(invoice_no)
```

Complaint :-

```
postgres=# \d complaint
                     Table "public.complaint"
    Column    |         Type          | Collation | Nullable | Default
--------------+-----------------------+-----------+----------+---------
 complaint_id | character varying(10) |           | not null |
 id           | integer               |           | not null |
 customer_id  | character varying(10) |           | not null |
 complaint    | character varying(25) |           | not null |
Indexes:
    "complaint_pkey" PRIMARY KEY, btree (complaint_id)
Foreign-key constraints:
    "complaint_customer_id_fkey" FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
    "complaint_id_fkey" FOREIGN KEY (id) REFERENCES details(id)
```

Facility :-

```
postgres=# \d facility
                    Table "public.facility"
   Column    |         Type          | Collation | Nullable | Default
-------------+-----------------------+-----------+----------+---------
 facility_id | character varying(10) |           | not null |
 id          | integer               |           | not null |
 room_type   | character varying(10) |           |          |
 e_bill      | character varying(5)  |           |          |
 w_bill      | character varying(5)  |           |          |
 c_bill      | character varying(5)  |           |          |
Indexes:
    "facility_pkey" PRIMARY KEY, btree (facility_id)
Check constraints:
    "facility_c_bill_check" CHECK (c_bill::text = ANY (ARRAY['NO'::character varying, 'YES'::character varying]::text[])
)
    "facility_e_bill_check" CHECK (e_bill::text = ANY (ARRAY['NO'::character varying, 'YES'::character varying]::text[])
)
    "facility_room_type_check" CHECK (room_type::text = ANY (ARRAY['AC'::character varying, 'NON AC'::character varying]
::text[]))
    "facility_w_bill_check" CHECK (w_bill::text = ANY (ARRAY['NO'::character varying, 'YES'::character varying]::text[])
)
Foreign-key constraints:
    "facility_id_fkey" FOREIGN KEY (id) REFERENCES details(id)
```

Feedback :-

```
postgres=# \d facility
                    Table "public.facility"
   Column    |         Type          | Collation | Nullable | Default
-------------+-----------------------+-----------+----------+---------
 facility_id | character varying(10) |           | not null |
 id          | integer               |           | not null |
 room_type   | character varying(10) |           |          |
 e_bill      | character varying(5)  |           |          |
 w_bill      | character varying(5)  |           |          |
 c_bill      | character varying(5)  |           |          |
Indexes:
    "facility_pkey" PRIMARY KEY, btree (facility_id)
Check constraints:
    "facility_c_bill_check" CHECK (c_bill::text = ANY (ARRAY['NO'::character varying, 'YES'::character varying]::text[])
)
    "facility_e_bill_check" CHECK (e_bill::text = ANY (ARRAY['NO'::character varying, 'YES'::character varying]::text[])
)
    "facility_room_type_check" CHECK (room_type::text = ANY (ARRAY['AC'::character varying, 'NON AC'::character varying]
::text[]))
    "facility_w_bill_check" CHECK (w_bill::text = ANY (ARRAY['NO'::character varying, 'YES'::character varying]::text[])
)
Foreign-key constraints:
    "facility_id_fkey" FOREIGN KEY (id) REFERENCES details(id)
```

(5) DATABASE TABLE CREATION

(5.1) CREATE SCHEMA

Customers :-

```
CREATE TABLE CUSTOMERS(
CUSTOMER_ID VARCHAR(20) PRIMARY KEY NOT NULL,
NAME VARCHAR(10) NOT NULL,
EMAIL_ID VARCHAR(30),
GENDER VARCHAR(9),
MOBILE_NO NUMERIC(10),
CITY VARCHAR NOT NULL,
PINCODE NUMERIC(6) NOT NULL,
CHECK(GENDER in ('MALE', 'FEMALE'))
);
```

Owner :-

```
CREATE TABLE OWNER(
OWNER_ID VARCHAR(10) PRIMARY KEY NOT NULL,
NAME VARCHAR(10) NOT NULL,
EMAIL_ID VARCHAR(30),
MOBILE_NO NUMERIC(10)
);
```

House :-

```
CREATE TABLE HOUSE(
HOUSE_ID VARCHAR(10) PRIMARY KEY,
OWNER_ID VARCHAR(10) NOT NULL,
PINCODE NUMERIC(6) NOT NULL,
CITY VARCHAR(10) NOT NULL,
LANDMARK VARCHAR(30),
BLOCK_NO INT,
FOREIGN KEY(OWNER_ID) REFERENCES OWNER (OWNER_ID));
```

Room :-

```
CREATE TABLE ROOM(
ID INT NOT NULL,
CUSTOMER_ID VARCHAR(10) NOT NULL,
RENT INT,
CAPACITY INT,
FOREIGN KEY(ID) REFERENCES DETAILS(ID),
FOREIGN KEY(CUSTOMER_ID) REFERENCES CUSTOMERS (CUSTOMER_ID)
);
```

Details :-

```
CREATE TABLE DETAILS(
ID INT PRIMARY KEY,
ROOM_ID VARCHAR(20) NOT NULL,
HOUSE_ID VARCHAR(20) NOT NULL,
FOREIGN KEY(HOUSE_ID) REFERENCES HOUSE(HOUSE_ID)
);
```

Invoice :-

```
CREATE TABLE INVOICE(
INVOICE_NO VARCHAR(10) PRIMARY KEY,
DURATION INT NOT NULL,
ID INT NOT NULL,
CUSTOMER_ID VARCHAR(10) NOT NULL,
AMOUNT INT NOT NULL,
FOREIGN KEY(CUSTOMER_ID) REFERENCES CUSTOMERS (CUSTOMER_ID),
FOREIGN KEY(ID) REFERENCES DETAILS(ID)
);
```

Payment :-

```
CREATE TABLE PAYMENT(
PAYMENT_ID VARCHAR(10) PRIMARY KEY,
INVOICE_NO VARCHAR(10) NOT NULL,
PAYMENT_DATE DATE,
AMOUNT INT NOT NULL,
FOREIGN KEY(INVOICE_NO) REFERENCES INVOICE (INVOICE_NO));
```

Complaint :-

```
CREATE TABLE COMPLAINT(
COMPLAINT_ID VARCHAR(10) PRIMARY KEY,
ID INT NOT NULL,
CUSTOMER_ID VARCHAR(10) NOT NULL,
COMPLAINT VARCHAR(25) NOT NULL,
FOREIGN KEY(ID) REFERENCES DETAILS(ID),
FOREIGN KEY(CUSTOMER_ID) REFERENCES CUSTOMERS (CUSTOMER_ID)
);
```

Facility :-

```
CREATE TABLE FACILITY(
FACILITY_ID VARCHAR(10) PRIMARY KEY,
ID INT NOT NULL,
ROOM_TYPE VARCHAR(10),
E_BILL VARCHAR(5),
W_BILL VARCHAR(5),
C_BILL VARCHAR(5),
CHECK(E_BILL in ('NO', 'YES')),
CHECK(W_BILL in ('NO', 'YES')),
CHECK(C_BILL in ('NO', 'YES')),
CHECK(ROOM_TYPE in ('AC', 'NON AC')),
FOREIGN KEY(ID) REFERENCES DETAILS(ID)
);
```

Feedback :-

```
CREATE TABLE FEEDBACK(
FEEDBACK_ID VARCHAR(10) PRIMARY KEY NOT NULL,
RATING INT CHECK(RATING<=5),
COMMENT VARCHAR(20),
HOUSE_ID VARCHAR(10) NOT NULL,
CUSTOMER_ID VARCHAR(10) NOT NULL,
FOREIGN KEY(CUSTOMER_ID) REFERENCES CUSTOMERS (CUSTOMER_ID),
FOREIGN KEY(HOUSE_ID) REFERENCES HOUSE(HOUSE_ID));
```

(5.2) INSERT DATA VALUES

Customers :-

```
postgres=# SELECT * FROM CUSTOMERS;
 customer_id | name |    email_id    | gender | mobile_no  |  city   | pincode
-------------+------+----------------+--------+------------+---------+---------
 C101        | JAY  | jay@gmail.com  | MALE   | 9909123456 | DHORAJI |  371008
 C102        | SITA | sita@gmail.com | FEMALE | 9909846573 | RAJKOT  |  381006
 C103        | RAJ  | raj@gmail.com  | MALE   | 9084620037 | SURAT   |  351007
 C104        | GITA | gita@gmail.com | FEMALE | 9909198756 | VAPI    |  393002
 C105        | MEET | meet@gmail.com | MALE   | 9253647569 | SURAT   |  351007
(5 rows)
```

Owner :-

```
postgres=# SELECT * FROM OWNER;
 owner_id |  name   |     email_id      | mobile_no
----------+---------+-------------------+------------
 O101     | PARTH   | parth@gmail.com   | 9347658225
 O102     | HARSHAL | harshal@gmail.com | 9831290635
 O103     | ANIL    | anil@gmail.com    | 9347927104
(3 rows)
```

House :-

```
postgres=# SELECT * FROM HOUSE;
 house_id | owner_id | pincode |  city   |    landmark     | block_no
----------+----------+---------+---------+-----------------+----------
 H101     | O102     |  340002 | NADIAD  | KAILASH BAAG    |       10
 H102     | O101     |  340002 | NADIAD  | AKSHAR TOWNSHIP |       24
 H103     | O102     |  340002 | NADIAD  | SATYAM SOCIETY  |        7
 H104     | O103     |  340002 | NADIAD  | KAILASH BAAG    |        7
(4 rows)
```

Room :-

```
postgres=# SELECT * FROM ROOM;
 id | customer_id | rent | capacity
----+-------------+------+----------
  1 | C102        | 1500 |        2
  1 | C104        | 1500 |        2
  4 | C103        | 1800 |        3
  5 | C101        | 2100 |        1
  6 | C105        | 2500 |        3
(5 rows)
```

Details :-

```
postgres=# SELECT * FROM DETAILS;
 id | room_id | house_id
----+---------+----------
  1 | R101    | H101
  2 | R102    | H101
  3 | R103    | H101
  4 | R101    | H102
  5 | R102    | H102
  6 | R101    | H103
  7 | R101    | H104
  8 | R102    | H104
(8 rows)
```

Invoice :-

```
postgres=# SELECT * FROM INVOICE;
 invoice_no | duration | id | customer_id | amount
------------+----------+----+-------------+--------
 I1         |        6 |  1 | C102        |   1500
 I2         |        6 |  1 | C104        |   1500
 I3         |       12 |  4 | C103        |   1800
 I4         |       12 |  5 | C101        |   2000
 I5         |        6 |  6 | C105        |   2500
(5 rows)
```

Payment :-

```
postgres=# SELECT * FROM PAYMENT;
 payment_id | invoice_no | payment_date | amount
------------+------------+--------------+--------
 P1         | I1         | 2022-01-01   |   1500
 P2         | I2         | 2022-01-01   |   1500
 P3         | I3         | 2020-05-03   |   1800
 P4         | I4         | 2021-03-04   |   2000
 P5         | I5         | 2022-10-06   |   2500
(5 rows)
```

Complaint :-

```
postgres=# SELECT * FROM COMPLAINT;
 complaint_id | id | customer_id |    complaint
--------------+----+-------------+-----------------
 C1           |  1 | C104        | LIGHT GONE DOWN
 C2           |  4 | C103        | PIPE LEAKAGE
(2 rows)
```

Facility :-

```
postgres=# SELECT * FROM FACILITY;
 facility_id | id | room_type | e_bill | w_bill | c_bill
-------------+----+-----------+--------+--------+--------
 F1          |  1 | NON AC    | NO     | YES    | NO
 F2          |  2 | NON AC    | NO     | YES    | NO
 F3          |  3 | AC        | NO     | YES    | NO
 F4          |  4 | NON AC    | YES    | YES    | NO
 F5          |  5 | AC        | YES    | YES    | NO
 F6          |  6 | AC        | YES    | YES    | YES
(6 rows)
```

Feedback :-

```
postgres=# SELECT * FROM FEEDBACK;
 feedback_id | rating | comment | house_id | customer_id
-------------+--------+---------+----------+-------------
 FD1         |      3 | GOOD    | H101     | C102
 FD2         |      2 | BAD     | H102     | C103
 FD3         |      5 | BEST    | H102     | C101
(3 rows)
```

(5.3) QUERIES USING BASIC DBMS CONSTRUCTS:

1. **DISPLAY NUMBER OF AC ROOMS .**

```
postgres=# select count(id) as countofACroom from facility where room_type = 'AC';
 countofacroom
---------------
             3
(1 row)


postgres=#
```

2. **DISPLAY MAX RENT OF ROOM FROM ALL ROOMS.**

```
postgres=#  select max(amount) from payment;
 max
------
 2500
(1 row)


postgres=#
```

3. **DISPLAY CUSTOMER NAME IN ASENCENDING ORDER BY THEIR NAME .**

```
postgres=#  select * from customers order by name asc;
 customer_id | name |     email_id     | gender | mobile_no  |  city   | pincode
-------------+------+------------------+--------+------------+---------+---------
 C104        | GITA | gita@gmail.com   | FEMALE | 9909198756 | VAPI    |  393002
 C101        | JAY  | jay@gmail.com    | MALE   | 9909123456 | DHORAJI |  371008
 C105        | MEET | meet@gmail.com   | MALE   | 9253647569 | SURAT   |  351007
 C103        | RAJ  | raj@gmail.com    | MALE   | 9084620037 | SURAT   |  351007
 C102        | SITA | sita@gmail.com   | FEMALE | 9909846573 | RAJKOT  |  381006
(5 rows)


postgres=#
```

4. DISPLAY HOUSE_ID WITH CORRESPONDING ROOMS IN IT.

```
postgres=# select house_id,count(house_id) from details group by house_id;
 house_id | count
----------+-------
 H101     |     3
 H103     |     1
 H104     |     2
 H102     |     2
(4 rows)


postgres=#
```

5. DISPLAY MOBILE_NO,CUSTOMER NAME OF CUSTOMER HAVING “91” IN THEIR NO.

```
postgres=# select mobile_no,name from customers where mobile_no::text LIKE '%91%';
 mobile_no  | name
------------+------
 9909123456 | JAY
 9909198756 | GITA
(2 rows)


postgres=#
```

(5.4) QUERIES USING JOIN AND SUBQUERIES

6. DISPLAY OWNER DETAILS HAVING 3 BHK HOUSE IN AREA “KAILASH BAAG” .

```
postgres=# select *from owner where owner_id in
postgres-#  (select owner_id from house where landmark='KAILASH BAAG' AND house_id in
postgres(#  (select house_id from details group by house_id having count(house_id)=3));
 owner_id |  name   |     email_id      | mobile_no
----------+---------+-------------------+------------
 O102     | HARSHAL | harshal@gmail.com | 9831063509
(1 row)


postgres=#
```

7. DISPLAY COUNT OF CUSTOMER WHO GAVE RATING MORE THAN 3

```
postgres=#  select count(customer_id) as customer_count from customers where customer_id in(
postgres(# select customer_id from feedback where rating>3);
 customer_count
----------------
              1
(1 row)
```

8. DISPLAY OWNER AND CUSTOMER DETAILS WHO BOOKED HOUSE IN 2022.

```
postgres=# select * from customers,owner where customer_id in
postgres-# (select room.customer_id from room inner join details on room.id = details.id inner join house on details.house_id = hous
e.house_id where details.id in
postgres(# (select id from invoice where invoice_no in(select invoice_no from payment where payment_date in
postgres(# (select payment_date from payment where payment_date>='2022-01-01' and payment_date<'2023-01-01'))))
postgres-#  and owner_id in
postgres-# (select house.owner_id from room inner join details on room.id = details.id inner join house on details.house_id = house.
house_id where details.id in ( select id from invoice where invoice_no in
postgres(# (select invoice_no from payment where payment_date in
postgres(# (select payment_date from payment where payment_date>='2022-01-01' and payment_date<'2023-01-01'))));
 customer_id | name |    email_id    | gender | mobile_no  |  city  | pincode | owner_id |  name   |     email_id      | mobile_no
-------------+------+----------------+--------+------------+--------+---------+----------+---------+-------------------+------------
 C102        | SITA | sita@gmail.com | FEMALE | 9909846573 | RAJKOT |  381006 | O102     | HARSHAL | harshal@gmail.com | 9831063509
 C104        | GITA | gita@gmail.com | FEMALE | 9909198756 | VAPI   |  393002 | O102     | HARSHAL | harshal@gmail.com | 9831063509
 C105        | MEET | meet@gmail.com | MALE   | 9253647569 | SURAT  |  351007 | O102     | HARSHAL | harshal@gmail.com | 9831063509
(3 rows)
```

9. DISPLAY CUSTOMER DETAIL WHO PAYS RENT >= 2000.

```
postgres=# select * from customers where customer_id in(select customer_id from invoice where amount<=2000);
 customer_id | name |    email_id     | gender | mobile_no  |  city    | pincode
-------------+------+-----------------+--------+------------+----------+---------
 C101        | JAY  | jay@gmail.com   | MALE   | 9909123456 | DHORAJI  |  371008
 C102        | SITA | sita@gmail.com  | FEMALE | 9909846573 | RAJKOT   |  381006
 C103        | RAJ  | raj@gmail.com   | MALE   | 9084620037 | SURAT    |  351007
 C104        | GITA | gita@gmail.com  | FEMALE | 9909198756 | VAPI     |  393002
(4 rows)
```

10. DISPLAY CUSTOMER DETAILS WHO GAVE FEEDBACK AND COMPLAIT.

```
postgres=#  SELECT * FROM CUSTOMERS WHERE CUSTOMER_ID IN (SELECT FEEDBACK.CUSTOMER_ID FROM FEEDBACK INNER JOIN COMPLAINT ON FEEDBACK
.CUSTOMER_ID = COMPLAINT.CUSTOMER_ID);
 customer_id | name |   email_id     | gender | mobile_no  | city  | pincode
-------------+------+----------------+--------+------------+-------+---------
 C103        | RAJ  | raj@gmail.com  | MALE   | 9084620037 | SURAT |  351007
(1 row)
```

(5.5) PL/SQL:

VIEW :-

```
postgres=# create view duration as
postgres-# select * from invoice where duration=12;
CREATE VIEW
postgres=# select *from duration;
 invoice_no | duration | id | customer_id | amount
------------+----------+----+-------------+--------
 I3         |       12 |  4 | C103        |   1800
 I4         |       12 |  5 | C101        |   2000
(2 rows)


postgres=#
```

(5.6) CURSOR:

1. CREATE A CURSOR WHOSE RENT GREATER THAN 1700.

```
postgres=# BEGIN;
BEGIN
postgres=*# DECLARE mycur CURSOR FOR
postgres-*# SELECT *FROM ROOM WHERE rent>1700;
DECLARE CURSOR
postgres=*# FETCH ALL FROM mycur;
 id | customer_id | rent | capacity
----+-------------+------+----------
  4 | C103        | 1800 |        3
  5 | C101        | 2100 |        1
  6 | C105        | 2500 |        3
(3 rows)


postgres=*# FETCH PRIOR FROM mycur;
 id | customer_id | rent | capacity
----+-------------+------+----------
  6 | C105        | 2500 |        3
(1 row)


postgres=*# FETCH PRIOR FROM mycur;
 id | customer_id | rent | capacity
----+-------------+------+----------
  5 | C101        | 2100 |        1
(1 row)


postgres=*# FETCH PRIOR FROM mycur;
 id | customer_id | rent | capacity
----+-------------+------+----------
  4 | C103        | 1800 |        3
(1 row)


postgres=*# FETCH NEXT FROM mycur;
 id | customer_id | rent | capacity
----+-------------+------+----------
  5 | C101        | 2100 |        1
(1 row)
```

```
postgres=*# CLOSE mycur;
CLOSE CURSOR
postgres=*# END;
COMMIT
postgres=#
```

(5.7) FUNCTION AND TRIGGERS

1. **CREATE A FUNCTION AND TRIGGERS WHICH CHECK IF AMOUNT LESS AND EQUAL TO 0 THEN IT WON'T ALLOW TO INSERT OR UPDATE THE PAYMENT TABLE.**

FUNCTION:-

```
create function check_payment() returns trigger as $$
BEGIN
if NEW.amount <=0 then
raise exception 'Min amount should be greater than 0';
end if;
return NEW;
END;
$$
LANGUAGE plpgsql;
```

TRIGGERS:-

```
create trigger amount_check
BEFORE INSERT OR UPDATE
ON payment
FOR EACH ROW
EXECUTE PROCEDURE check_payment();
```

```
postgres=# create function check_payment() returns trigger as $$
postgres$#  BEGIN
postgres$#  if NEW.amount <=0 then
postgres$#  raise exception 'Min amount should be greater than 0’;
postgres$#  end if;
postgres$# return NEW;
postgres$#  END;
postgres$# $$
postgres-#  LANGUAGE plpgsql;
CREATE FUNCTION
postgres=# create trigger amount_check
postgres-#  BEFORE INSERT OR UPDATE
postgres-#  ON payment
postgres-#  FOR EACH ROW
postgres-#  EXECUTE PROCEDURE check_payment();
CREATE TRIGGER
postgres=# INSERT INTO PAYMENT(PAYMENT_ID,INVOICE_NO,PAYMENT_DATE,AMOUNT) VALUES('P6','I6','01-01-2019',0);
ERROR:  Min amount should be greater than 0
CONTEXT:  PL/pgSQL function check_payment() line 4 at RAISE
postgres=# INSERT INTO PAYMENT(PAYMENT_ID,INVOICE_NO,PAYMENT_DATE,AMOUNT) VALUES('P6','I6','01-01-2019',1600);
INSERT 0 1
postgres=# select * from payment;
 payment_id | invoice_no | payment_date | amount
------------+------------+--------------+--------
 P1         | I1         | 2022-01-01   |   1500
 P2         | I2         | 2022-01-01   |   1500
 P3         | I3         | 2020-05-03   |   1800
 P4         | I4         | 2021-03-04   |   2000
 P5         | I5         | 2022-10-06   |   2500
 P6         | I6         | 2019-01-01   |   1600
(6 rows)


postgres=#
```

2. CREATE A FUNCTION AND TRIGGERS WHICH KEEP LOG OF ALL THE USER WHO INSERT,UPDATE OR DELETE THE DATA FROM OWNER TABLE.

TABLE :-

```
CREATE  TABLE owner_audit(
operation character(1) not null,
stamp timestamp not null,
userid text not null,
owner_id text,
name text ,
email_id text,
mobile_no NUMERIC(10)
);
```

FUNCTION :-

```
CREATE OR REPLACE FUNCTION do_owner_audit()
RETURNS TRIGGER AS $owner_audit$
BEGIN
IF (TG_OP = 'DELETE') THEN
INSERT INTO owner_audit
SELECT 'D', now(), user , OLD.*;
RETURN OLD;
ELSIF (TG_OP = 'UPDATE') THEN
INSERT INTO owner_audit SELECT 'U', now(), user, NEW.*;
RETURN NEW;
ELSIF (TG_OP = 'INSERT') THEN
INSERT INTO owner_audit SELECT 'I', now(), user, NEW.*;
RETURN NEW;
END IF;
RETURN NULL;
END;
$owner_audit$ LANGUAGE PLPGSQL;
```

TRIGGERS :-

CREATE TRIGGER owner_audit
AFTER INSERT OR UPDATE OR DELETE ON owner
FOR EACH ROW EXECUTE PROCEDURE
do_owner_audit();

```
postgres=# create table owner_audit(
postgres(# operation character(1) not null,
postgres(# stamp timestamp not null,
postgres(# userid text not null,
postgres(# owner_id text,
postgres(# name text ,
postgres(# email_id text,
postgres(# mobile_no NUMERIC(10)
postgres(# );
CREATE TABLE
postgres=# CREATE OR REPLACE FUNCTION do_owner_audit()
postgres-# RETURNS TRIGGER AS $owner_audit$
postgres$#  BEGIN
postgres$#  IF (TG_OP = 'DELETE') THEN
postgres$#  INSERT INTO owner_audit
postgres$# SELECT 'D' , now(), user , OLD.*;
postgres$#  RETURN OLD;
postgres$# ELSIF (TG_OP = 'UPDATE') THEN
postgres$#  INSERT INTO owner_audit SELECT 'U', now(), user, NEW.*;
postgres$#  RETURN NEW;
postgres$# ELSIF (TG_OP = 'INSERT') THEN
postgres$#  INSERT INTO owner_audit SELECT 'I', now(), user, NEW.*;
postgres$#  RETURN NEW;
postgres$#  END IF;
postgres$#  RETURN NULL;
postgres$# END;
postgres$# $owner_audit$ LANGUAGE PLPGSQL;
CREATE FUNCTION
postgres=# CREATE TRIGGER owner_audit
postgres-# AFTER INSERT OR UPDATE OR DELETE ON owner
postgres-# FOR EACH ROW EXECUTE PROCEDURE
postgres-# do_owner_audit();
CREATE TRIGGER
```

```
postgres=# INSERT INTO OWNER (OWNER_ID,NAME,EMAIL_ID,MOBILE_NO) VALUES('O104','VRAJ','vraj@gmail.com',9332571225);
INSERT 0 1
postgres=# select *from owner_audit;
 operation |           stamp           |  userid  | owner_id | name |    email_id    | mobile_no
-----------+---------------------------+----------+----------+------+----------------+------------
 I         | 2022-10-05 21:33:39.55571 | postgres | O104     | VRAJ | vraj@gmail.com | 9332571225
(1 row)


postgres=# delete from owner where owner_id='O104';
DELETE 1
postgres=# select *from owner_audit;
 operation |           stamp           |  userid  | owner_id | name |    email_id    | mobile_no
-----------+---------------------------+----------+----------+------+----------------+------------
 I         | 2022-10-05 21:33:39.55571  | postgres | O104     | VRAJ | vraj@gmail.com | 9332571225
 D         | 2022-10-05 21:41:14.121916 | postgres | O104     | VRAJ | vraj@gmail.com | 9332571225
(2 rows)
```

(6) FUTURE ENHANCEMENTS OF THE SYSTEM

- We will learn html,css for fronted and javascript for backend and also use React Framework for frontend and Nodejs for backend development.

- To level up the project we will add some new entity (like login and signup for owner and customer) to our project.

(7) BIBLIOGRAPHY

We created ER-Model on Draw.io and Relational Schema on ERD+

ER-MODEL -

https://app.diagrams.net/

RELATIONAL SCHEMA -

https://erdplus.com/edit-diagram/ec867aa7-24c9-456c-81dd-f83d63e5dbbf

For the implementation of this project, we referred to materials shared by Prof. Archana N. Vyas and the following websites and books:

Book:

Database System Concepts

-Henry F. Korth & A. Silberschatz 2nd Ed. McGraw-Hill 1991

Websites:

https://www.postgresqltutorial.com/

https://www.geeksforgeeks.org/postgresql-tutorial/