Project Group W

Intelligent Building Management System

Final Project Report

04/25/22

Group Members

Ridhima Saxena

Siyuan Zhao

Yihong Yu

Sitanshu Rupani

Kushal Shah

Table of Contents

Final Project Report

Business Problem

Currently, the traditional building industry is undergoing digital transformation, which means a variety of different information systems are needed as auxiliary tools, and one of them is a highly integrated system called Intelligent Building Management System (IBMS). An intelligent building is one that uses technology to enable efficient and economical use of resources, while creating a safe and comfortable environment for occupants. Smart buildings may use a wide range of existing technologies and are designed or retrofitted in a way that allows for the integration of future technological developments. Internet of Things (IoT) sensors, building management systems, artificial intelligence (AI), and augmented reality are amongst some of the mechanisms and robotics that may be used in a smart building to control and optimize its performance. The most fundamental feature of a smart building is that the core systems within it are linked. Connecting smart technology, such as real-time IoT occupancy sensors and building management systems together, means you can share information that can be used to automate various processes, including, but not limited to, heating, ventilation, lighting, air conditioning, and security. This is what makes a building "smart" – the ability of the systems within it to talk to one another. Of course, to achieve a complete system is very much needed to support the business knowledge, so in this project, we choose to solve a small part of the content, that is, the management of maintenance work orders in the building.

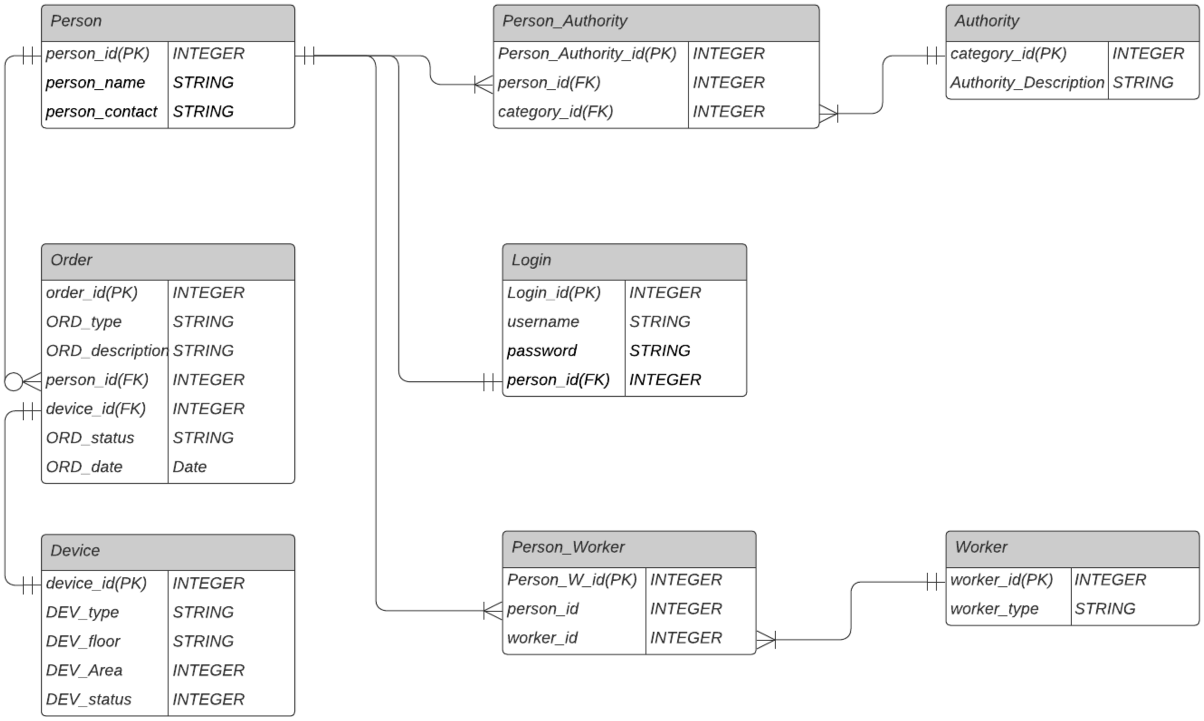
In this project, a realistic shopping mall will be used as a building model for business analysis. During the pre-construction to post-operation of the mall, various maintenance-related problems will occur, such as water leakage, HVAC equipment maintenance, some facilities needing to be replaced, etc.

For the manager, he needs to know what problems exist in the mall, so that he can quickly arrange the corresponding workers to carry out maintenance, ensure the safety of people in the mall, and provide a comfortable environment for customers. In this project, the manager can check and assign all the work orders and check the details of each piece of equipment, while the workers can check the work orders assigned to them so that they can go to the repair quickly, and this can be used as their performance evaluation standard.

In conclusion, sharing and integrating data between building systems enables the value of the combined smart building to be greater than the sum of its parts. That's our purpose.

Database Design

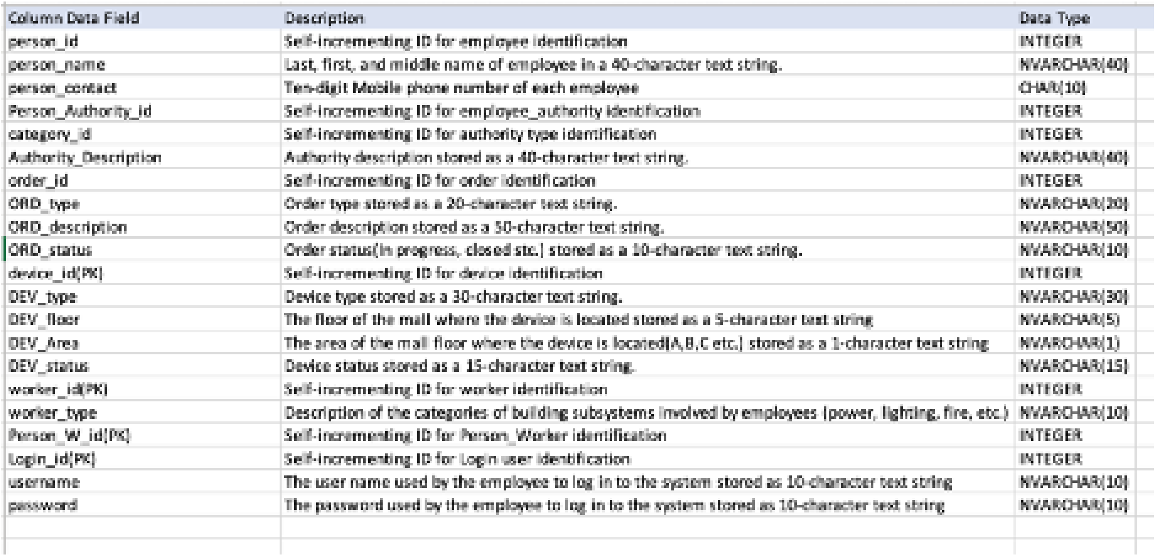
1. ERD



2. Normalization

Person (person_id, person_name, person_contact)
Order (order_id, ORD_type, ORD_description, ORD_status, *person_id*, *device_id*)
Device (device_id, DEV_type, DEV_floor, DEV_Area, DEV_status)
Login (Login_id, username, password, *person_id*)
Worker (worker_id, worker_type)
Authority (category_id, Authority_Description)
Person_ Authority (Person_Authority_id, *person_id*, *category_id*)
Person_Worker (Person_W_id, person_id, *person_id*, *worker_id*)

3. Data dictionary and Details



Column Data Field	Description	Data Type
person_id	Self-incrementing ID for employee identification	INTEGER
person_name	Last, first, and middle name of employee in a 40-character text string.	NVARCHAR(40)
person_contact	Ten-digit Mobile phone number of each employee	CHAR(10)
Person_Authority_id	Self-incrementing ID for employee_authority identification	INTEGER
category_id	Self-incrementing ID for authority type identification	INTEGER
Authority_Description	Authority description stored as a 40-character text string.	NVARCHAR(40)
order_id	Self-incrementing ID for order identification	INTEGER
ORD_type	Order type stored as a 20-character text string.	NVARCHAR(20)
ORD_description	Order description stored as a 50-character text string.	NVARCHAR(50)
ORD_status	Order status(in progress, closed stc.) stored as a 10-character text string.	NVARCHAR(10)
device_id(PK)	Self-incrementing ID for device identification	INTEGER
DEV_type	Device type stored as a 30-character text string.	NVARCHAR(30)
DEV_floor	The floor of the mall where the device is located stored as a 5-character text string	NVARCHAR(5)
DEV_Area	The area of the mall floor where the device is located(A,B,C etc.) stored as a 1-character text string	NVARCHAR(1)
DEV_status	Device status stored as a 15-character text string.	NVARCHAR(15)
worker_id(PK)	Self-incrementing ID for worker identification	INTEGER
worker_type	Description of the categories of building subsystems involved by employees (power, lighting, fire, etc.)	NVARCHAR(10)
Person_W_id(PK)	Self-incrementing ID for Person_Worker identification	INTEGER
Login_id(PK)	Self-incrementing ID for Login user identification	INTEGER
username	The user name used by the employee to log in to the system stored as 10-character text string	NVARCHAR(10)
password	The password used by the employee to log in to the system stored as 10-character text string	NVARCHAR(10)

(1) Person Table

Table Name	Person
Primary Key	person_id
Foreign Key	None
Purpose	Storing every person’s information
Indexed attributes	person_name, person_contact
Indexed attributes’ reasons	For querying persons’ details
Attribute Detail	**person_id(INTEGER):** Self-incrementing ID for employee identification **person_name(NVARCHAR(10)):** Last, first, and middle name of employee in a 40-character text string **person_contact(CHAR(10)):** Ten-digit Mobile phone number of each employee

(2) Order Table

Table Name	Order
Primary Key	order_id
Foreign Key	Person_id, device_id

Purpose	Storing every order has been added
Indexed attributes	ORD_type, ORD_description, ORD_status
Indexed attributes’ reasons	For querying orders’ currently status and contents
Attribute Detail	**order_id(INTEGER):** Self-incrementing ID for order identification **ORD_type(NVARCHAR(20)):** Order type stored as a 20-character text string **ORD_description(NVARCHAR(50)):** Order description stored as a 50-character text string **ORD_status(NVARCHAR(10)):** Order status(In progress, closed stc.) stored as a 10-character text string

(3) Device Table

Table Name	Device
Primary Key	device_id
Foreign Key	None
Purpose	Storing every device’s information
Indexed attributes	DEV_type, DEV_floor, DEV_area, DEV_status
Indexed attributes’ reasons	For querying devices’ details and currently status
Attribute Detail	**device_id(INTEGER):** Self-incrementing ID for device identification **DEV_type(NVARCHAR(30)):** Device type stored as a 30-character text string **DEV_floor(NVARCHAR(5)):** The floor of tha mall where the device is located stored as a 5-character text string **DEV_area(NVARCHAR(1)):** The area of the mall floor where the device is located(A, B, C etc.) stored as a 1-character text string **DEV_status(NVARCHAR(15)):** Device status stored as a 15-character text string

(4) Login Table

Table Name	Login

Primary Key	login_id
Foreign Key	person_id
Purpose	Storing every person who has logined to the system
Indexed attributes	Username, password
Indexed attributes’ reasons	For querying users’ login details
Attribute Detail	**login_id(INTEGER):** Self-incrementing ID for login user identification **username(NVARCHAR(10)):** The username used by the employee to log in to the system stored as 10-character text string **password(NVARCHAR(10)):** The password used by the employee to log in to the system stored as 10-character text string

(5) Worker Table

Table Name	Worker
Primary Key	worker_id
Foreign Key	None
Purpose	Storing every worker’s information
Indexed attributes	DEV_type
Indexed attributes’ reasons	Different workers will be allcated by their type
Attribute Detail	**worker_id(INTEGER):** Self-incrementing ID for worker identification **worker_type(NVARCHAR(10)):** Description of the categories of building subsystems involved by employees(power, light, fire, etc.)

(6) Authority Table

Table Name	Authority
Primary Key	category_id

Foreign Key	None
Purpose	For identifying every employee’s Authority
Indexed attributes	Authority_Description
Indexed attributes’ reasons	Storing different authority in details
Attribute Detail	**category_id(INTEGER):** Self-incrementing ID for authority type identification **Authority_Desciption(NVARCHAR(40)):** Authority description stored as a 40-character text string

(7) Person_Authority Table

Table Name	Person_Authority
Primary Key	Person_Authority_id
Foreign Key	person_id, category_id
Purpose	As a associative table for storing the relationship between person and Authority
Indexed attributes	None
Indexed attributes’ reasons	None
Attribute Detail	**Person_Authority_id(INTEGER):** Self-incrementing ID for employee_authority identification

(8) Person_Worker Table

Table Name	Person_Worker
Primary Key	Person_W_id
Foreign Key	person_id, worker_id
Purpose	As a associative table for storing the relationship between person and worker
Indexed attributes	None
Indexed attributes’ reasons	None

Attribute Detail	**Person_W_id(INTEGER):** Self-incrementing ID for person_worker identification

4. Relationships

(1) Person and Order(1:M): Each employee can participate in the order, and can participate in multiple orders, for an employee, the order can be empty, but the order recorded in the Order must be assigned by an employee.

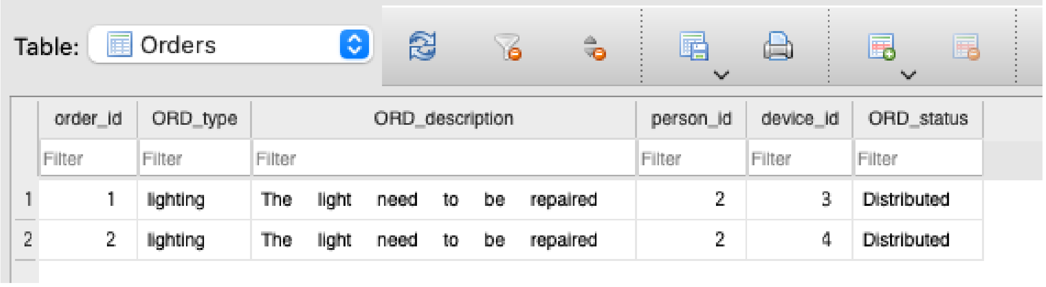


Table: Orders

	order_id	ORD_type	ORD_description	person_id	device_id	ORD_status
	Filter	Filter	Filter	Filter	Filter	Filter
1	1	lighting	The light need to be repaired	2	3	Distributed
2	2	lighting	The light need to be repaired	2	4	Distributed

(2) Person and Login(1:1): Each employee can only generate a unique account with his own unique id. One person cannot have two accounts at the same time, and one account can only be logged in by a unique employee.

(3) Person and Authority(M:M): Each employee can have multiple roles, such as floor manager can also be a worker. Therefore, an employee can have multiple permissions at the same time. And each permission can be owned by multiple people at the same time, for example, the floor manager can be held by multiple people.

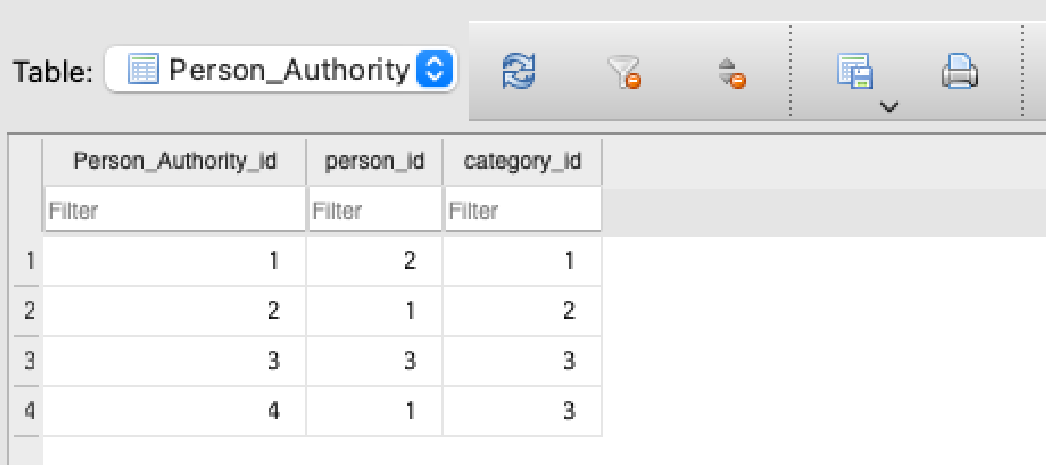
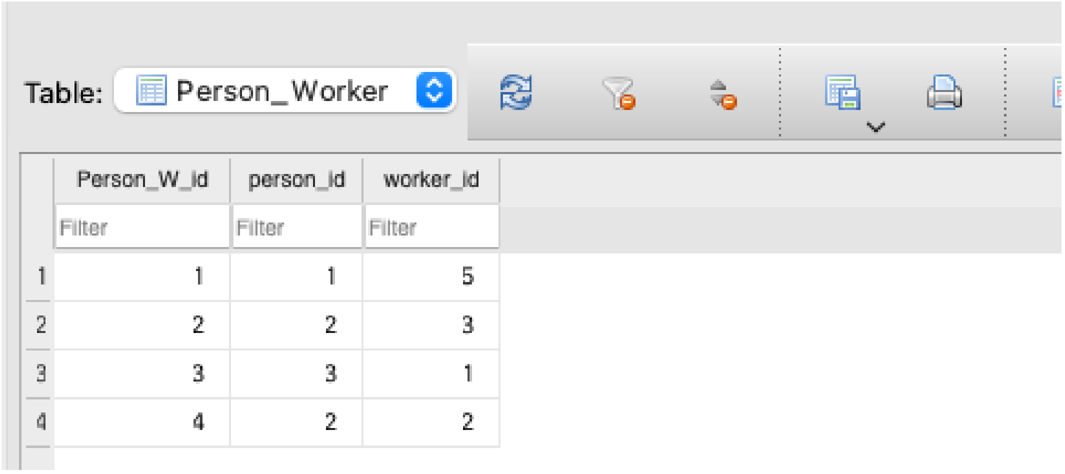


Table: Person_Authority

	Person_Authority_id	person_id	category_id
	Filter	Filter	Filter
1	1	2	1
2	2	1	2
3	3	3	3
4	4	1	3

(4) **Person and Worker(M:M):** Each worker can repair multiple types of equipment, such as water and electricity, and at the same time, a job can have multiple persons at the same time.



	Person_W_id	person_id	worker_id
	Filter	Filter	Filter
1	1	1	5
2	2	2	3
3	3	3	1
4	4	2	2

(5) **Order and Device(1:1):** Each established order can only process a unique device, no empty order can exist.

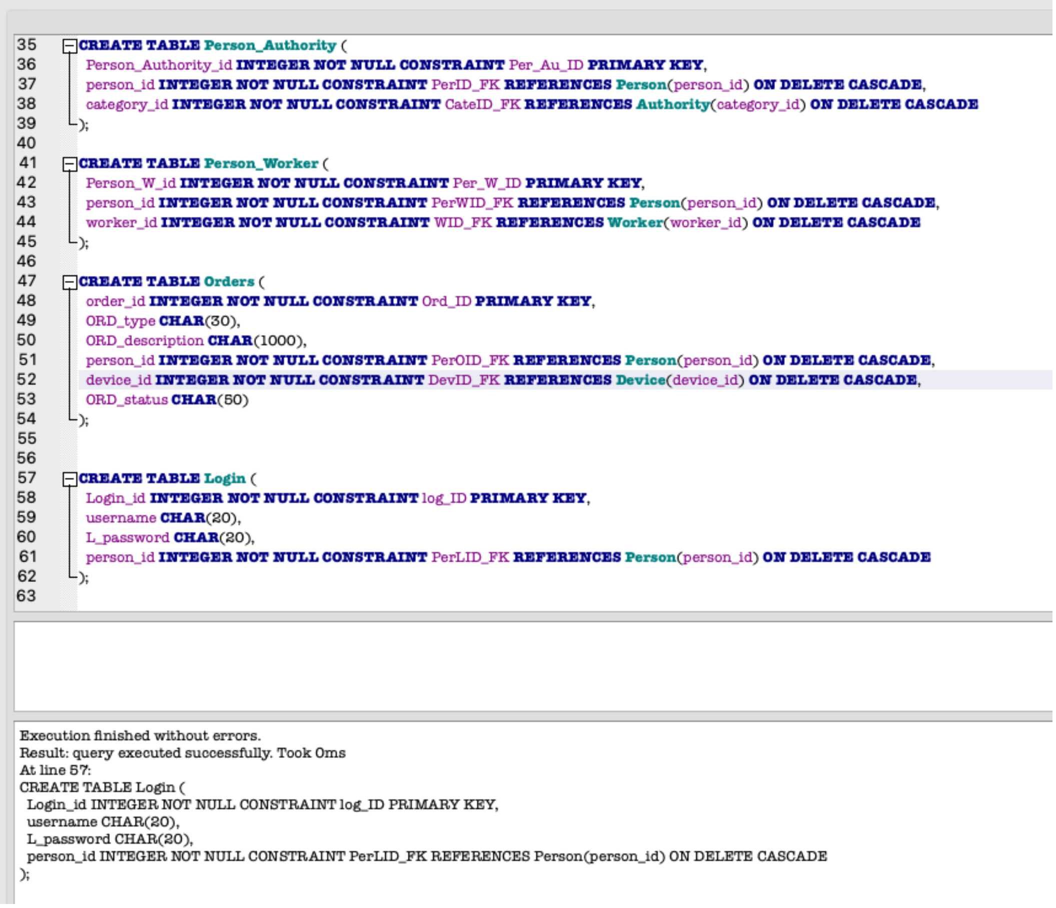
5. SQL Commands

(1) Create Tables

```
PRAGMA foreign_keys = on;
DROP TABLE IF EXISTS Orders;
DROP TABLE IF EXISTS Person_Authority;
DROP TABLE IF EXISTS Login;
DROP TABLE IF EXISTS Person_Worker;
DROP TABLE IF EXISTS Person;
DROP TABLE IF EXISTS Worker;
DROP TABLE IF EXISTS Authority;
DROP TABLE IF EXISTS Device;

CREATE TABLE Person (
  person_id INTEGER NOT NULL CONSTRAINT Per_ID PRIMARY KEY,
  person_name CHAR(20),
  person_contact CHAR(30)
);
```

```
CREATE TABLE Device (
  device_id INTEGER NOT NULL CONSTRAINT Dev_ID PRIMARY KEY,
  DEV_type CHAR(30),
  DEV_floor CHAR(30),
  DEV_Area INTEGER,
  DEV_status INTEGER
);

CREATE TABLE Authority (
  category_id INTEGER NOT NULL CONSTRAINT Cate_ID PRIMARY KEY,
  Authority_Description CHAR(1000)
);

CREATE TABLE Worker (
  worker_id INTEGER NOT NULL CONSTRAINT W_ID PRIMARY KEY,
  worker_type CHAR(30)
);

CREATE TABLE Person_Authority (
  Person_Authority_id INTEGER NOT NULL CONSTRAINT Per_Au_ID PRIMARY
KEY,
  person_id INTEGER NOT NULL CONSTRAINT PerID_FK REFERENCES
Person(person_id) ON DELETE CASCADE,
  category_id INTEGER NOT NULL CONSTRAINT CateID_FK REFERENCES
Authority(category_id) ON DELETE CASCADE
);

CREATE TABLE Person_Worker (
  Person_W_id INTEGER NOT NULL CONSTRAINT Per_W_ID PRIMARY KEY,
  person_id INTEGER NOT NULL CONSTRAINT PerWID_FK REFERENCES
Person(person_id) ON DELETE CASCADE,
  worker_id INTEGER NOT NULL CONSTRAINT WID_FK REFERENCES
Worker(worker_id) ON DELETE CASCADE
);

CREATE TABLE Orders (
  order_id INTEGER NOT NULL CONSTRAINT Ord_ID PRIMARY KEY,
  ORD_type CHAR(30),
  ORD_description CHAR(1000),
  person_id INTEGER NOT NULL CONSTRAINT PerOID_FK REFERENCES
Person(person_id) ON DELETE CASCADE,
  device_id INTEGER NOT NULL CONSTRAINT DevID_FK REFERENCES
Device(device_id) ON DELETE CASCADE,
  ORD_status CHAR(50)
);

CREATE TABLE Login (
  Login_id INTEGER NOT NULL CONSTRAINT log_ID PRIMARY KEY,
  username CHAR(20),
  L_password CHAR(20),
  person_id INTEGER NOT NULL CONSTRAINT PerLID_FK REFERENCES
Person(person_id) ON DELETE CASCADE
);
```

Output:

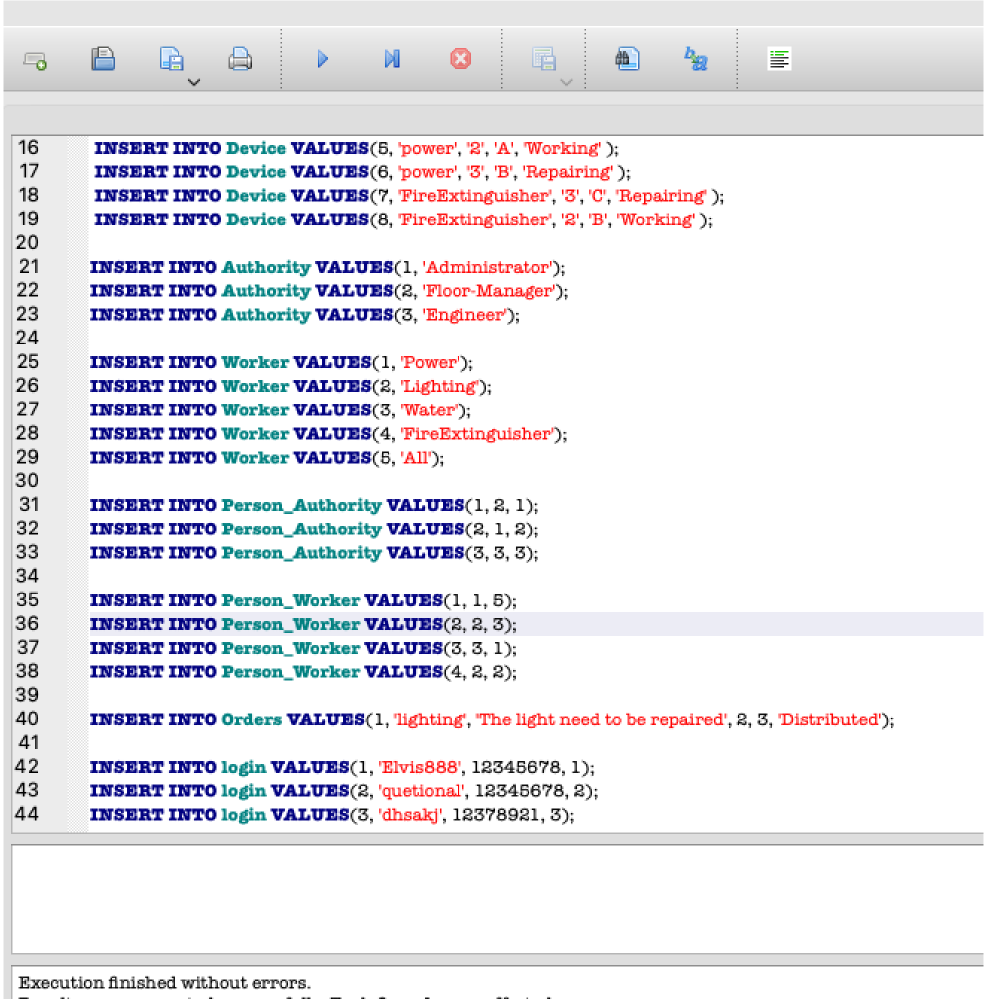


```
35 CREATE TABLE Person_Authority (
36   Person_Authority_id INTEGER NOT NULL CONSTRAINT Per_Au_ID PRIMARY KEY,
37   person_id INTEGER NOT NULL CONSTRAINT PerID_FK REFERENCES Person(person_id) ON DELETE CASCADE,
38   category_id INTEGER NOT NULL CONSTRAINT CateID_FK REFERENCES Authority(category_id) ON DELETE CASCADE
39 );
40
41 CREATE TABLE Person_Worker (
42   Person_W_id INTEGER NOT NULL CONSTRAINT Per_W_ID PRIMARY KEY,
43   person_id INTEGER NOT NULL CONSTRAINT PerWID_FK REFERENCES Person(person_id) ON DELETE CASCADE,
44   worker_id INTEGER NOT NULL CONSTRAINT WID_FK REFERENCES Worker(worker_id) ON DELETE CASCADE
45 );
46
47 CREATE TABLE Orders (
48   order_id INTEGER NOT NULL CONSTRAINT Ord_ID PRIMARY KEY,
49   ORD_type CHAR(30),
50   ORD_description CHAR(1000),
51   person_id INTEGER NOT NULL CONSTRAINT PerOID_FK REFERENCES Person(person_id) ON DELETE CASCADE,
52   device_id INTEGER NOT NULL CONSTRAINT DevID_FK REFERENCES Device(device_id) ON DELETE CASCADE,
53   ORD_status CHAR(50)
54 );
55
56
57 CREATE TABLE Login (
58   Login_id INTEGER NOT NULL CONSTRAINT log_ID PRIMARY KEY,
59   username CHAR(20),
60   L_password CHAR(20),
61   person_id INTEGER NOT NULL CONSTRAINT PerLID_FK REFERENCES Person(person_id) ON DELETE CASCADE
62 );
63
```

```
Execution finished without errors.
Result: query executed successfully. Took 0ms
At line 57:
CREATE TABLE Login (
  Login_id INTEGER NOT NULL CONSTRAINT log_ID PRIMARY KEY,
  username CHAR(20),
  L_password CHAR(20),
  person_id INTEGER NOT NULL CONSTRAINT PerLID_FK REFERENCES Person(person_id) ON DELETE CASCADE
);
```

(2) Insert data

```
INSERT  INTO Person VALUES(1, 'Jake', '5417450622')
INSERT  INTO Person VALUES(2, 'Nena', '5167400622');
INSERT  INTO Person VALUES(3, 'Horge', '9078506223');
INSERT  INTO Person VALUES(4, 'Verg', '9087651131');
INSERT  INTO Person VALUES(5, 'Hoggins', '7665309826');
INSERT  INTO Person VALUES(6, 'William', '1908786551');
INSERT  INTO Person VALUES(7, 'Norman', '7865560983');
INSERT  INTO Person VALUES(8, 'Shalia', '4509987678');
INSERT  INTO Person VALUES(9, 'Sai', '1213435096');
INSERT  INTO Person VALUES(10, 'Yuke', '9876455609');

INSERT INTO Device VALUES(1, 'Lighting', '1', 'A', 'Broken' );
INSERT INTO Device VALUES(2, 'Lighting', '1', 'B', 'Working' );
INSERT INTO Device VALUES(3, 'Lighting', '1', 'C', 'Repairing' );
INSERT INTO Device VALUES(4, 'power', '1', 'A', 'Working' );
INSERT INTO Device VALUES(5, 'power', '2', 'A', 'Working' );
INSERT INTO Device VALUES(6, 'power', '3', 'B', 'Repairing' );
INSERT INTO Device VALUES(7, 'FireExtinguisher', '3', 'C',
'Repairing' );
INSERT INTO Device VALUES(8, 'FireExtinguisher', '2', 'B', 'Working'
);

INSERT INTO Authority VALUES(1, 'Administrator');
INSERT INTO Authority VALUES(2, 'Floor-Manager');
INSERT INTO Authority VALUES(3, 'Engineer');

INSERT INTO Worker VALUES(1, 'Power');
INSERT INTO Worker VALUES(2, 'Lighting');
INSERT INTO Worker VALUES(3, 'Water');
INSERT INTO Worker VALUES(4, 'FireExtinguisher');
INSERT INTO Worker VALUES(5, 'All');

INSERT INTO Person_Authority VALUES(1, 2, 1);
INSERT INTO Person_Authority VALUES(2, 1, 2);
INSERT INTO Person_Authority VALUES(3, 3, 3);

INSERT INTO Person_Worker VALUES(1, 1, 5);
INSERT INTO Person_Worker VALUES(2, 2, 3);
INSERT INTO Person_Worker VALUES(3, 3, 1);
INSERT INTO Person_Worker VALUES(4, 2, 2);

INSERT INTO Orders VALUES(1, 'lighting', 'The light need to be
repaired', 2, 3, 'Distributed');

INSERT INTO login VALUES(1, 'Elvis888', 12345678, 1);
INSERT INTO login VALUES(2, 'quetional', 12345678, 2);
INSERT INTO login VALUES(3, 'dhsakj', 12378921, 3);
```



```
 INSERT INTO Device VALUES(5, 'power', '2', 'A', 'Working' );
 INSERT INTO Device VALUES(6, 'power', '3', 'B', 'Repairing' );
 INSERT INTO Device VALUES(7, 'FireExtinguisher', '3', 'C', 'Repairing' );
 INSERT INTO Device VALUES(8, 'FireExtinguisher', '2', 'B', 'Working' );

INSERT INTO Authority VALUES(1, 'Administrator');
INSERT INTO Authority VALUES(2, 'Floor-Manager');
INSERT INTO Authority VALUES(3, 'Engineer');

INSERT INTO Worker VALUES(1, 'Power');
INSERT INTO Worker VALUES(2, 'Lighting');
INSERT INTO Worker VALUES(3, 'Water');
INSERT INTO Worker VALUES(4, 'FireExtinguisher');
INSERT INTO Worker VALUES(5, 'All');

INSERT INTO Person_Authority VALUES(1, 2, 1);
INSERT INTO Person_Authority VALUES(2, 1, 2);
INSERT INTO Person_Authority VALUES(3, 3, 3);

INSERT INTO Person_Worker VALUES(1, 1, 5);
INSERT INTO Person_Worker VALUES(2, 2, 3);
INSERT INTO Person_Worker VALUES(3, 3, 1);
INSERT INTO Person_Worker VALUES(4, 2, 2);

INSERT INTO Orders VALUES(1, 'lighting', 'The light need to be repaired', 2, 3, 'Distributed');

INSERT INTO login VALUES(1, 'Elvis888', 12345678, 1);
INSERT INTO login VALUES(2, 'quetional', 12345678, 2);
INSERT INTO login VALUES(3, 'dhsakj', 12378921, 3);
```

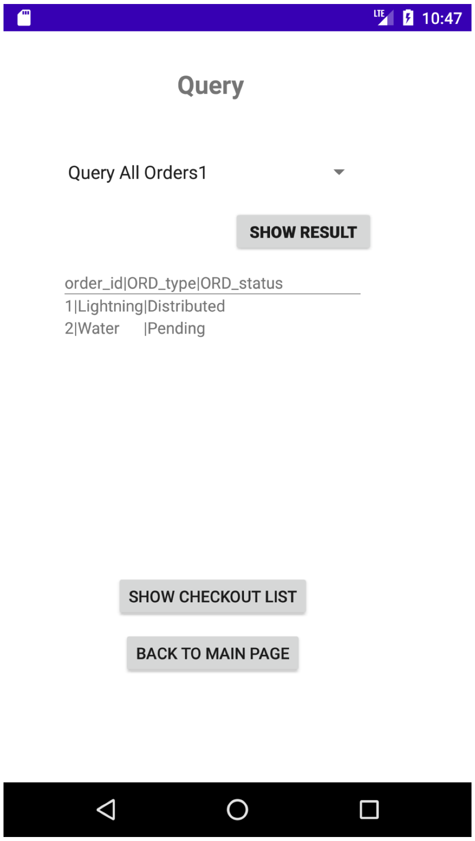
```
Execution finished without errors.
```

User Interface and Query Documentation

1. **SQL Queries**
   For our project we utilized 13 queries as on the back-end of our application. Each query helps the employees and managers to fulfill a particular business use case. The queries included are:-

a. A query that displays all orders with their order ID, order type and its current status. This query will help the user see a list of all orders irrespective of their status or type and will allow the user to have an idea of which orders have been completed and which are still in progress. This query is accessible as:-
   Admin -> Query Order -> Query All Orders -> Show Results

```
SELECT order_id, ORD_type, ORD_status FROM Orders;
```

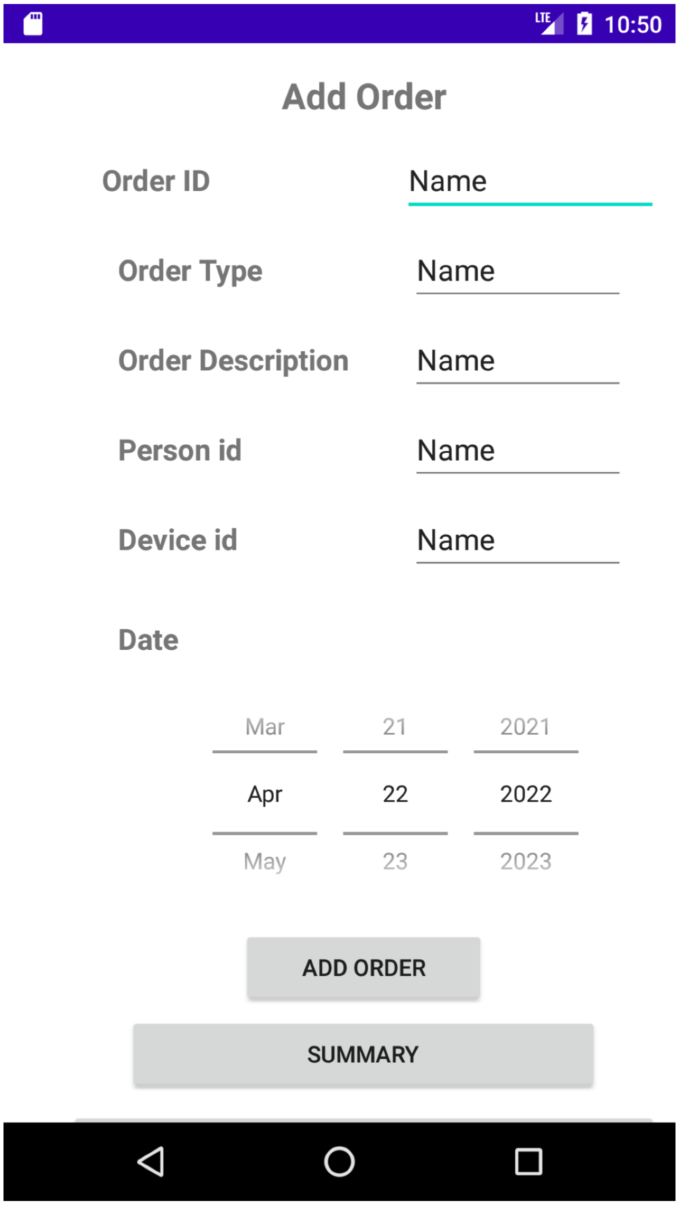


```
order_id|ORD_type|ORD_status
1|Lightning|Distributed
2|Water     |Pending
```

Query a. Output

b. This query is an update feature that helps the user to insert records into the database. Specifically, it allows the user who has access to create new entries for new orders. This query is accessible as:-
Admin -> Add Order -> Fill Order Information -> Add Order

```
INSERT INTO
Orders(order_id,ORD_type,ORD_description,person_id,device_id,ORD
_status,Created_date) VALUES(?,?,?,?,?,?,?);
```



Add Order

Order ID	Name
Order Type	Name
Order Description	Name
Person id	Name
Device id	Name

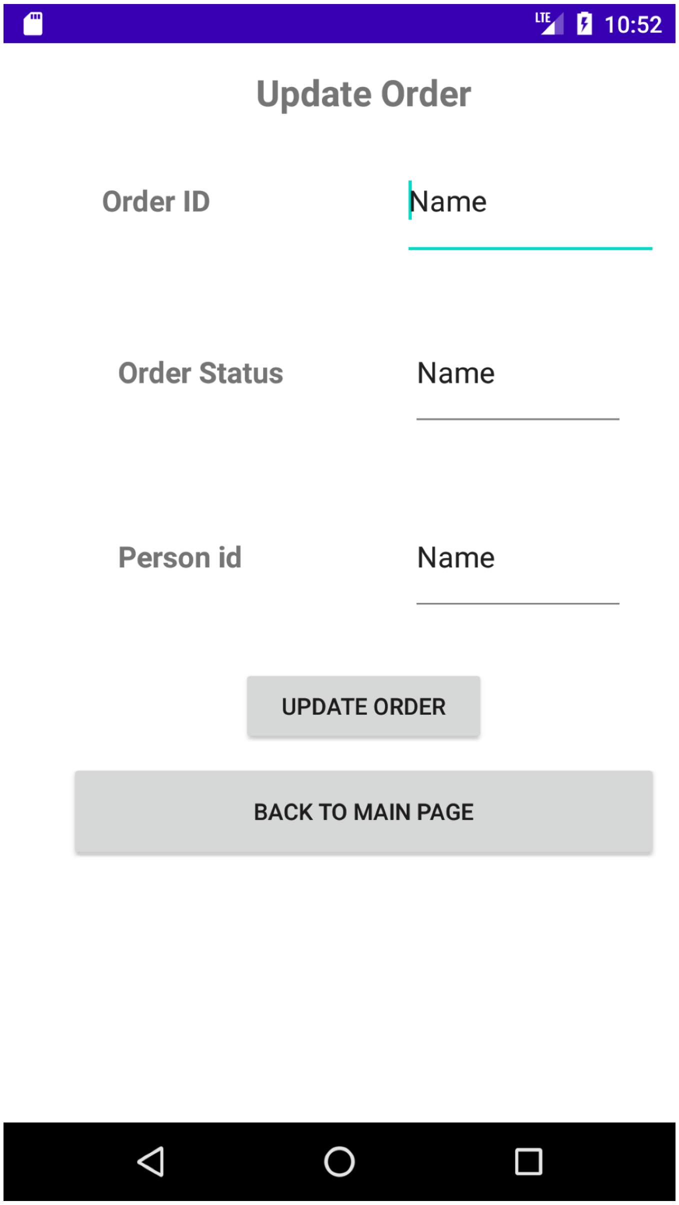
Date

Mar	21	2021
Apr	22	2022
May	23	2023

Query b. Output

c. An update query that allows engineers working on orders to update its status(eg. From “in progress” to “completed”) or to change the engineer assigned to that order. This query is accessible as:-
Floor Manager/Engineer -> Update Order -> Enter Order ID -> Updated Order information -> Update Order

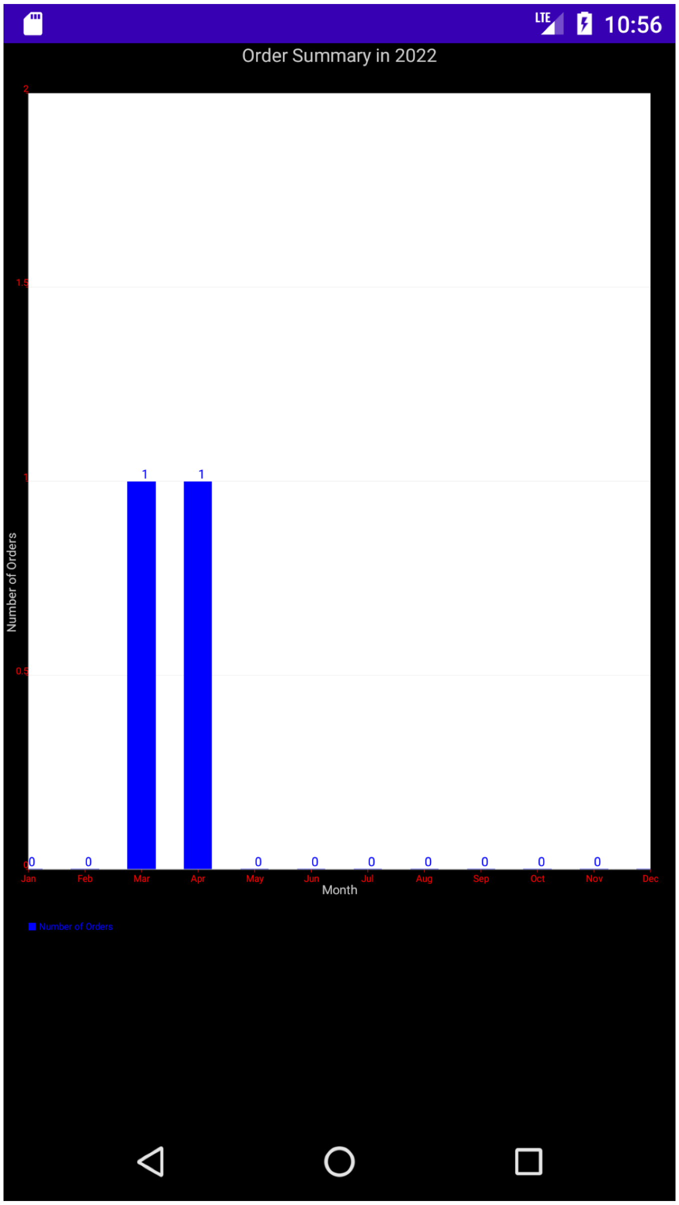
```
UPDATE Orders SET ORD_status=?,person_id=? WHERE order_id=?;
```



Query c. Output

d. This query outputs a summary report of all orders that are grouped by their creation date. A future feature could allow the user to input a particular time period and to see the summary chart of all orders upto that time period. This query is accessible as:-
   Admin/Floor Manager -> Add Order -> Summary

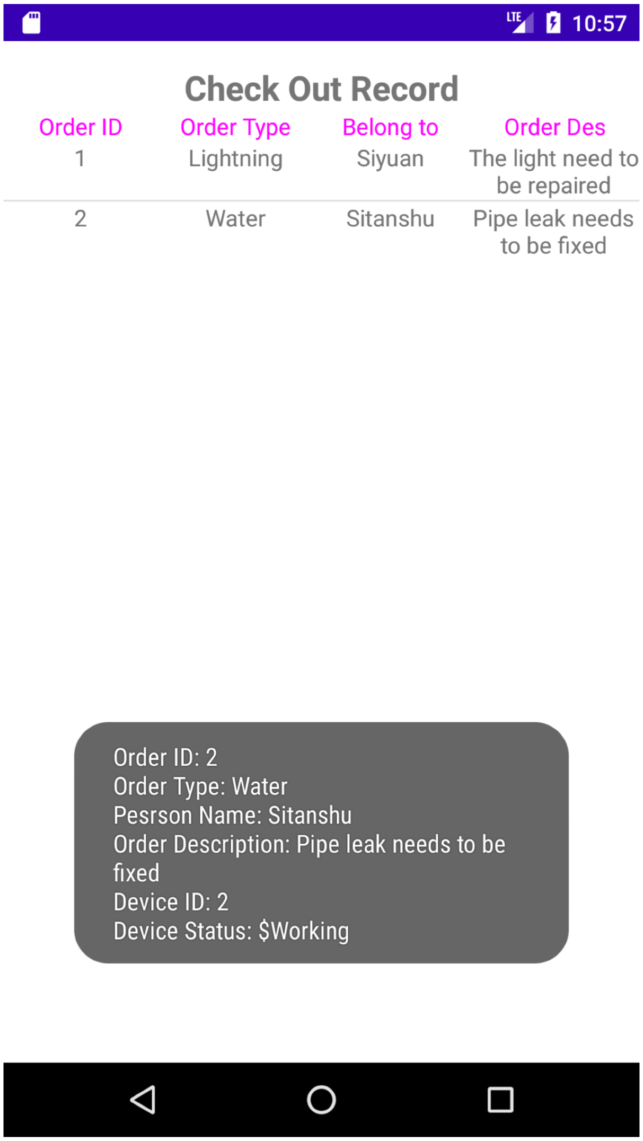
```
SELECT strftime('%m',Created_date) AS MONTH,COUNT(*) AS total
FROM Orders WHERE strftime('%Y',Created_date)='2022' GROUP BY
month ORDER BY total DESC;
```



Query d. Output

e. A query that works on admin level access. It allows the supervisor to view the list of orders and its status. Also implemented, is a Toast feature which allows the user to touch a particular order and see more information such as the engineer assigned to the order, the device ID and the device status. This query is accessible as:-
Admin -> Query Order -> Select Query from Picker -> Show Checkout List

```
SELECT Orders.order_id AS _id, ORD_type AS ordertype,
Person.person_name AS personname, ORD_description AS
orderdescription, Device.device_id AS deviceid, DEV_status AS
devicestatus FROM Orders, Person, Device WHERE
Orders.person_id=Person.person_id AND
Orders.device_id=Device.device_id;
```



Check Out Record

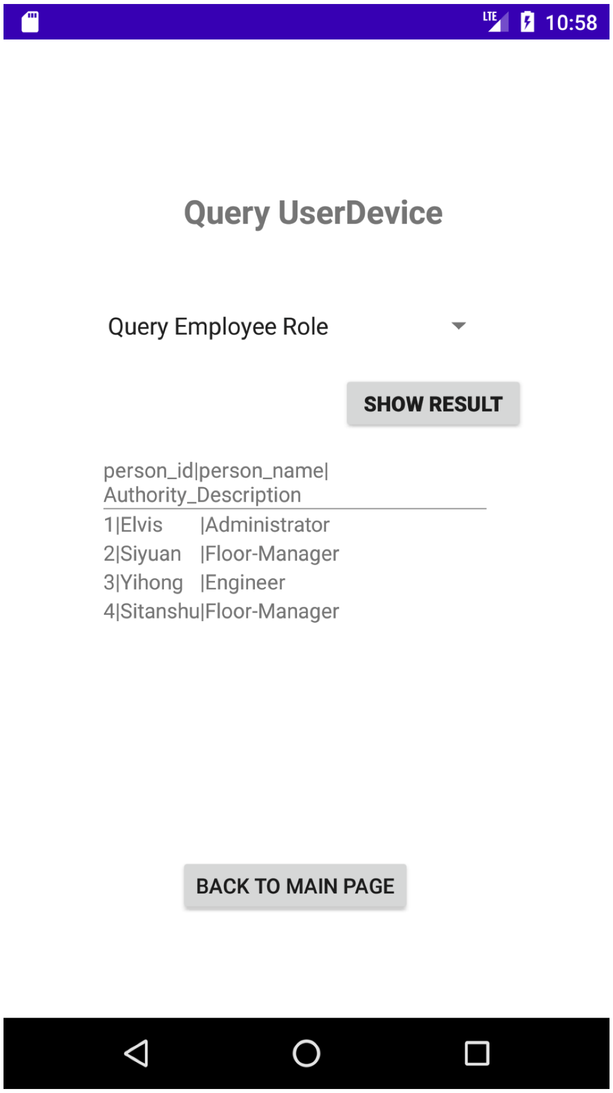
Order ID	Order Type	Belong to	Order Des
1	Lightning	Siyuan	The light need to be repaired
2	Water	Sitanshu	Pipe leak needs to be fixed

Order ID: 2
Order Type: Water
Pesrson Name: Sitanshu
Order Description: Pipe leak needs to be fixed
Device ID: 2
Device Status: $Working

Query e. Output

f. Another admin level feature, this query allows the user to view all employees in the company and their subsequent details such as employee ID, name, and designation at the company. This feature allows the user of the app to see a list of all employees at the firm. This query is accessible as:-
Admin/Floor Manager -> Query User & Device -> Query Employee Role

```
SELECT Person.person_id, person_name, Authority_Description FROM
Person, Authority, Person_Authority WHERE
Person.person_id=Person_Authority.Person_Authority_id AND
Person_Authority.category_id=Authority.category_id;
```

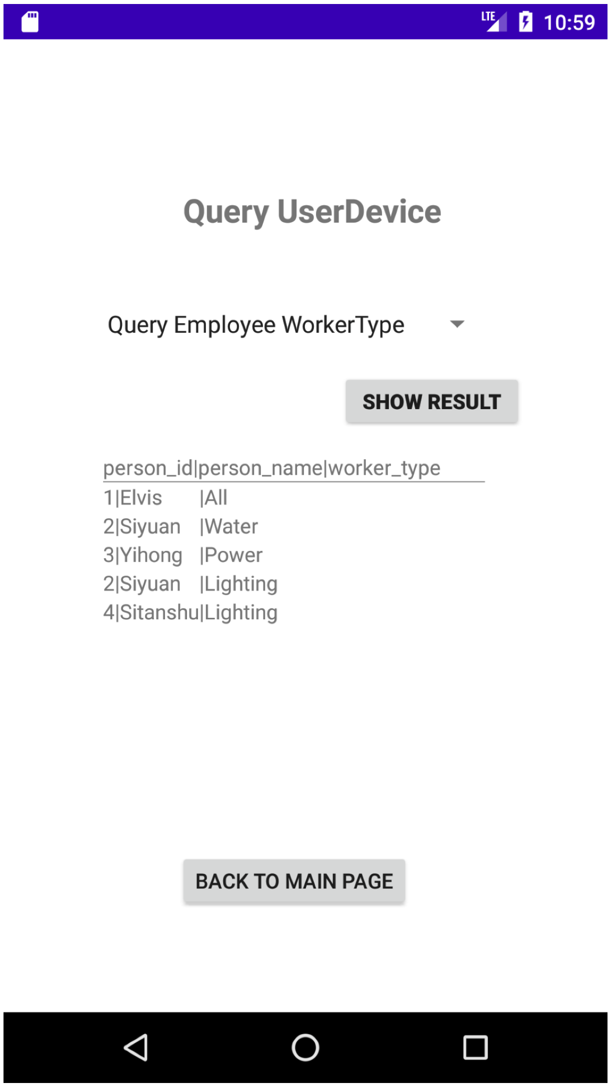


```
person_id|person_name|
Authority_Description
1|Elvis     |Administrator
2|Siyuan    |Floor-Manager
3|Yihong    |Engineer
4|Sitanshu|Floor-Manager
```

Query f. Output

g. This query helps the user to look up all employees along with the department. This feature could help the user look up what employees are assigned to what departments to more easily assign new work orders to the correct employee. This query is accessible as:-
Admin/Floor Manager -> Query User & Device -> Query Employee Worker Type

```
SELECT Person.person_id, person_name, worker_type FROM Person,
Worker, Person_Worker WHERE
Person.person_id=Person_Worker.person_id AND
Person_Worker.worker_id=Worker.worker_id;
```

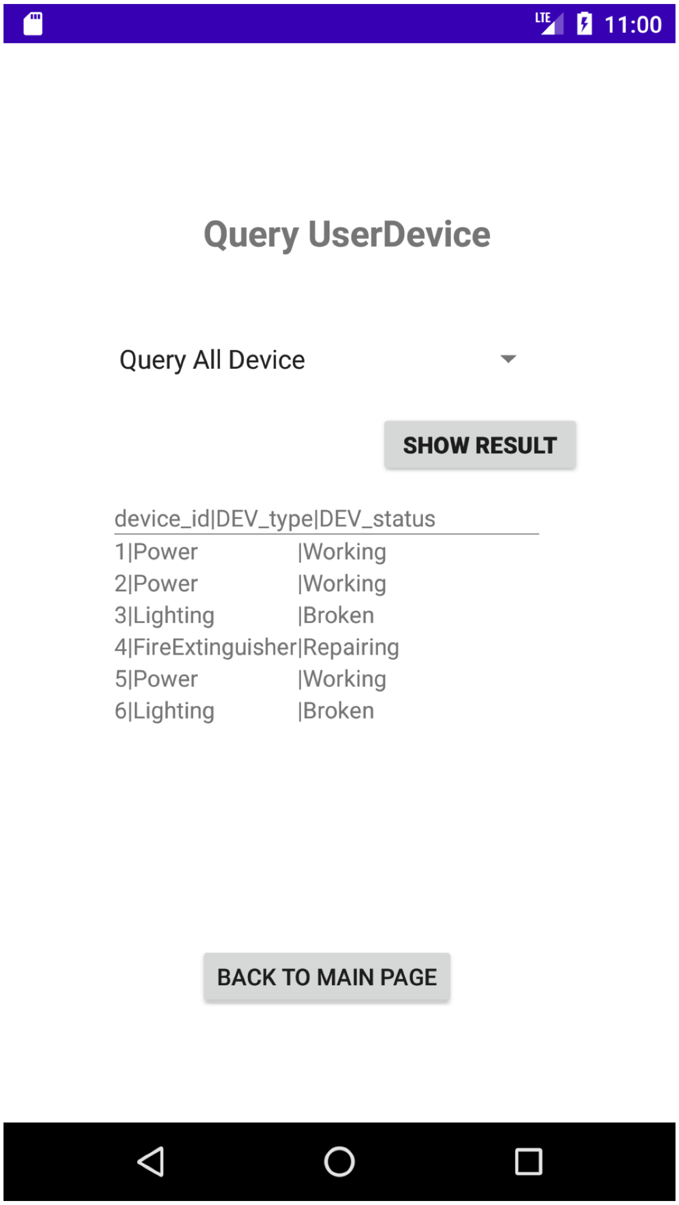


```
person_id|person_name|worker_type
1|Elvis    |All
2|Siyuan   |Water
3|Yihong   |Power
2|Siyuan   |Lighting
4|Sitanshu|Lighting
```

Query g. Output

h. This query allows the user to query all devices that are owned and inventoried by the company/building. It shows the user the device ID, the device name and its current status. This query is accessible as:-
Admin/Floor Manager -> Query User & Device -> Query Device

```
SELECT device_id, DEV_type, DEV_status FROM Device;
```

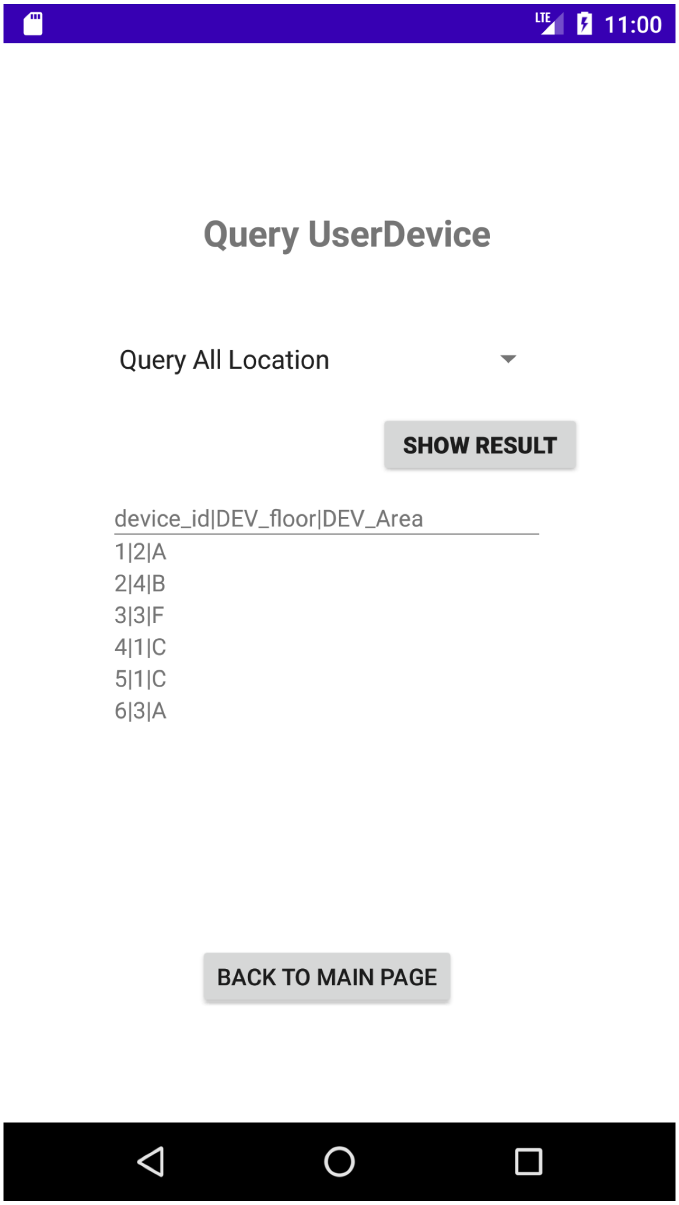


```
device_id|DEV_type|DEV_status
1|Power           |Working
2|Power           |Working
3|Lighting        |Broken
4|FireExtinguisher|Repairing
5|Power           |Working
6|Lighting        |Broken
```

Query h. Output

i. A query to show the user the location of all the devices in the building. A handy guide for the user to easily locate and keep a track of where these devices(eg. Fire extinguishers, etc.) are located. This query is accessible as:-
Admin/Floor Manager -> Query User & Device -> Query Location

```
SELECT device_id, DEV_floor, DEV_Area FROM Device;
```



```
device_id|DEV_floor|DEV_Area
1|2|A
2|4|B
3|3|F
4|1|C
5|1|C
6|3|A
```

Query i. Output

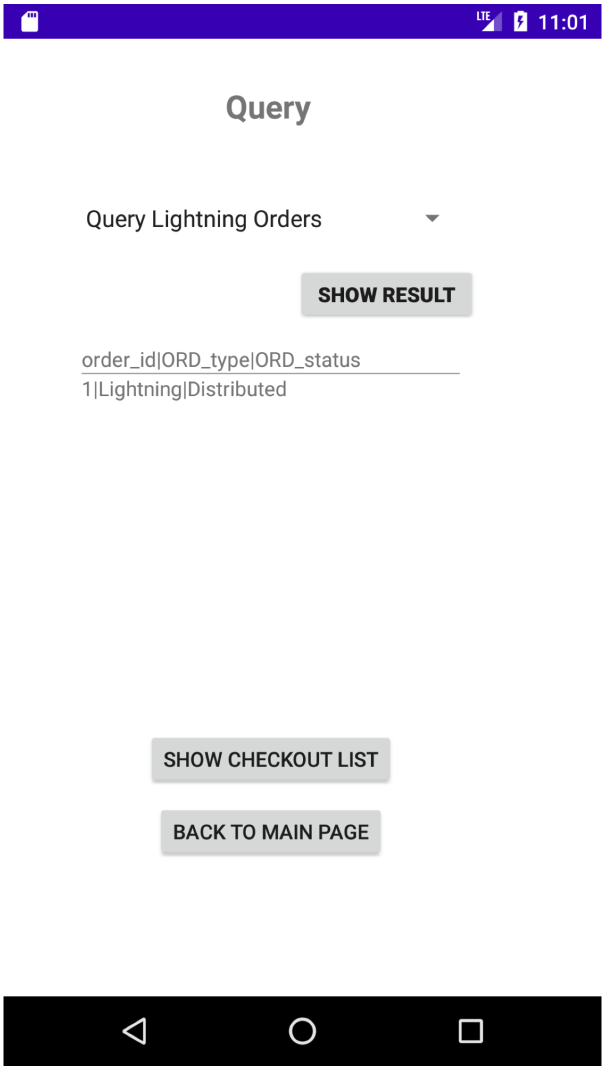
j. The next set of queries are all linked to a single functionality. These queries help the user sort and output the list of orders based on the order type. The user can choose to view Lighting, Power, Water and Firefighting devices based on what is chosen from the picker. This query is accessible as:-
Admin -> Query User & Device -> Query Power/Lighting/Firefighting Orders

```
SELECT order_id, ORD_type, ORD_status FROM Orders WHERE
ORD_type='Lightning';
```

```
SELECT order_id, ORD_type, ORD_status FROM Orders WHERE
ORD_type='Power';
```

```
SELECT order_id, ORD_type, ORD_status FROM Orders WHERE
ORD_type='Water';
```

```
SELECT order_id, ORD_type, ORD_status FROM Orders WHERE
ORD_type='Firefighting';
```



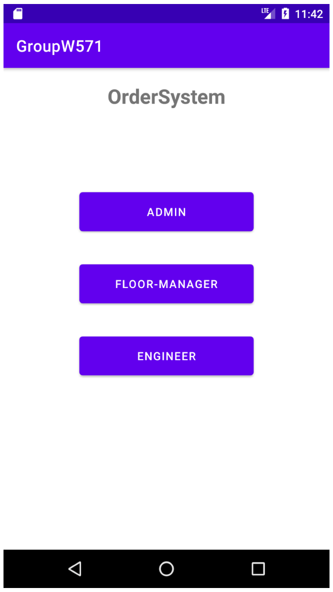
Query Lightning Orders

```
order_id|ORD_type|ORD_status
1|Lightning|Distributed
```

Query j. Output

2. **App Navigation**

   The application has 3 main screens for the particular job role of the user. An admin role for upper management and the database administrators. The Floor Manager role is created for managers in charge of a particular floor. Finally, we have the engineer role for the technicians who the orders are assigned to. Each of these roles have a button that leads to their respective pages from the main screen.



Main Screen

Admin Screen

The admin screen allows the user with admin level access to access more features than the lower tier users.

The admins can access a Query Order screen where they can choose from a query picker. These queries are all related to orders and their status in the database. The Show Checkout List button at the bottom of the screen shows a summary of all orders which can show more information by touching a particular order and utilizing Toast to popup more information.

The Query User and Device screen is similar to the Query Order screen but where the latter allows the user to query order information, the former deals with showcasing user and device information.

The Add Order button is a special feature only for admin level users, where they can add new orders into the database. A special Summary button on the Add Order Screen allows the admin level users to see a bar chart for all orders in a given year to give a macro, overall view of the whole system.

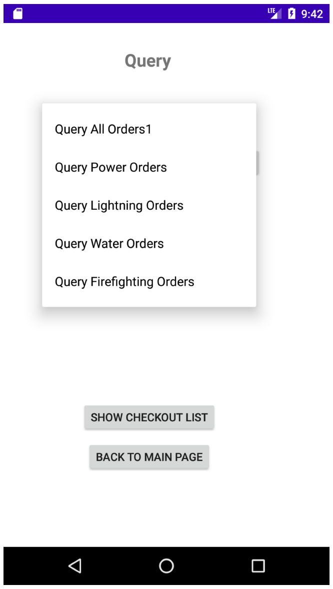
Floor Manager

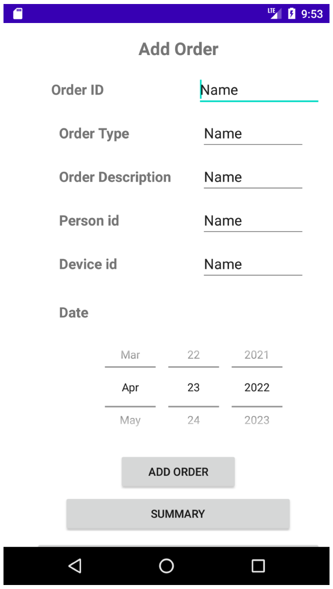
The floor manager screen is for users who have the manager designation.They cannot add new orders but have similar permissions for other features that admin level users have.

The Floor Manager role has an additional feature where they can update existing orders. The Update Order Button can achieve this task.

Engineer

The Engineer screen has limited features tied to the role. They may only Update Order based on work completed on orders assigned to them.





Order ID Name

Order Type Name

Order Description Name

Person id Name

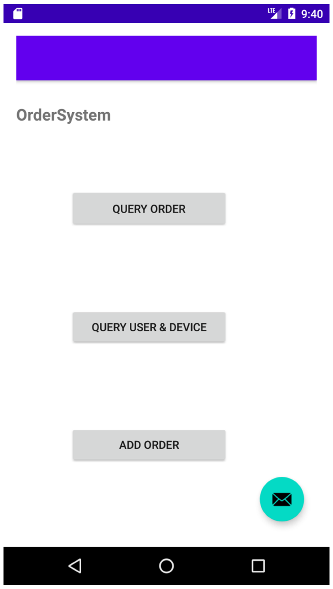
Device id Name

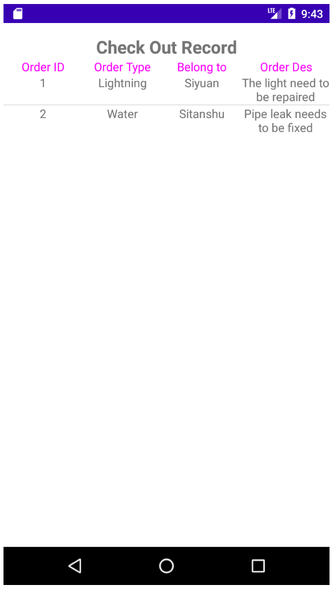
Date

Mar 22 2021

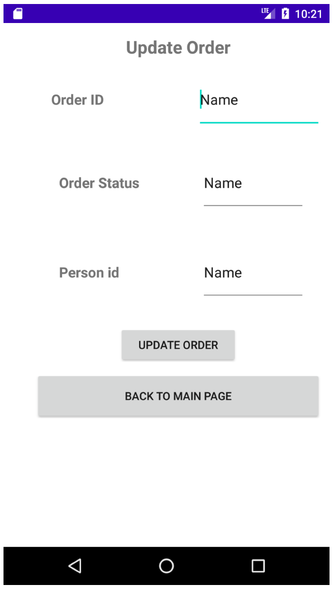
Apr 23 2022

May 24 2023





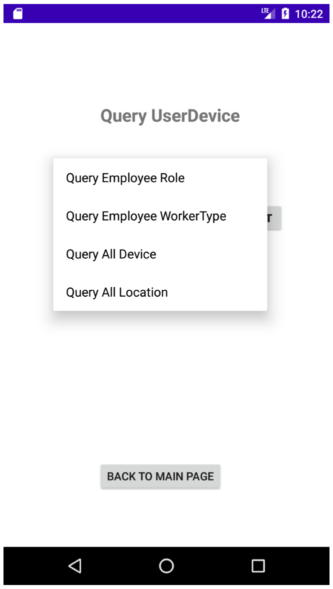
Order ID	Order Type	Belong to	Order Des
1	Lightning	Siyuan	The light need to be repaired
2	Water	Sitanshu	Pipe leak needs to be fixed



Order ID Name

Order Status Name

Person id Name



Technical Documentation and Implementation

The application in its current state as whole has a size of ~11 megabytes. But as the database for orders increases or as the number of employees increases, the app will get heavier and the memory will increase.

The app has already defined access control roles for the Admin, Manager and Engineer roles. The database also has a table to store usernames and passwords. To add an extra level of security a login functionality can be added using the Login table from the database to verify the login information. Without this access control, employees who do not have the correct credentials may edit order information causing havoc with the order tracking system.

Some potential data quality issues that could happen are duplications of orders. This could cause issues in allocation of work to engineers. A good way to mitigate this issue would be to ensure employees are trained sufficiently to use the app. This could help mitigate most data quality issues that can arise from using the app.

The application in its current state has no backup data feature. A good idea for future iterations of app development would be to periodically storing a snapshot of the database on a cloud location so that in case of a catastrophic failure of the application, there is minimum data loss.

The code can be found at this LINK.