Project Part - 3 Welcome Home

Languages and Frameworks Used:
SpringBoot, MySQL, React

Changes in schema: N/A

Queries Used:

Registration and Login:

Registration:

```
INSERT INTO PERSON (userName, password, fname, lname, email) VALUES (abc, abc, abc, abc, abc@example.com);
```

```
INSERT INTO PersonPhone (userName, phone) VALUES (abc, 1234567890);
```

```
INSERT INTO ACT (userName, roleID) VALUES (abc, ‘R0001’);
```

Login:

```
SELECT * FROM PERSON WHERE userName = ‘abc’;
```

Find single Item:

```
SELECT * FROM Location l JOIN Piece p ON p.roomNum = l.roomNum AND p.shelfNum = l.shelfNum WHERE p.itemID = 1
```

Find Single Order:

```
SELECT
   l.roomNum AS locRoomNum,
   l.shelfNum AS locShelfNum,
   l.shelf,
   l.shelfDescription,
   p.itemID,
   p.pieceNum,
   p.pDescription,
   p.length,
   p.height,
   p.pNotes,
   p.roomNum AS pieceRoomNum,
   p.shelfNum AS pieceShelfNum,
   i.iDescription
FROM Location l
JOIN Piece p
   ON p.roomNum = l.roomNum
   AND p.shelfNum = l.shelfNum
JOIN Item i
   ON p.itemID = i.ItemID
WHERE p.itemID IN (
   SELECT itemID
```

```
    FROM ItemIn
    WHERE orderID = 1234
);
```

Accept Donation:

```
INSERT INTO Item (iDescription, color, isNew, hasPieces, material, mainCategory,
subCategory)
VALUES (“sofa”, “green”, true, true, “cotton”, “furniture”, “sofa”);

INSERT INTO DonatedBy (ItemID, userName, donateDate)
VALUES (4, “testUser123”, CURDATE());

INSERT INTO Piece (ItemID, pieceNum, pDescription, length, width, height, roomNum,
shelfNum)
VALUES (4, 1, “sofa mattress”, 146, 56, 6, 5, 0);
```

Start an Order:

```
INSERT INTO Ordered (orderDate, orderNotes, supervisor, client) VALUES (CURDATE(), 'In
Progress', ‘kamal’, ‘testuser3’);

INSERT INTO ItemIn (ItemID, orderID, found) VALUES (?, ?, true);
```

Users Task:

```
SELECT
    o.orderID,
    o.orderDate,
    o.orderNotes,
    o.supervisor,
    o.client,
    i.ItemID,
    i.iDescription,
    i.color,
    i.material,
    i.isNew,
    CASE
        WHEN o.client = “kamal” THEN 'Client Ordered'
        WHEN o.supervisor = “kamal” THEN 'Staff Supervised'
        WHEN d.userName = “kamal” THEN 'Volunteer Delivered'
        ELSE 'No Relationship'
    END AS relationship,
    d.userName AS deliveredBy
FROM Ordered o
JOIN ItemIn ii ON o.orderID = ii.orderID
JOIN Item i ON ii.ItemID = i.ItemID
LEFT JOIN Delivered d ON o.orderID = d.orderID
LEFT JOIN Person v ON v.userName = d.userName
LEFT JOIN Act a ON a.userName = v.userName AND a.roleID = 'R0002'
WHERE o.client = “kamal”
    OR o.supervisor = “kamal”
    OR d.userName = “kamal”
    OR (v.userName = “kamal" AND a.roleID = 'R0002');
```

Difficulties and Lessons Learned:

Working with complex SQL queries presented several challenges. One difficulty was managing multiple joins across tables, ensuring that the relationships between them were accurately defined. For instance, joining tables like `Ordered`, `ItemIn`, `Delivered`, and `Person` required careful attention to ensure the correct keys were used. Additionally, handling conditional logic with `CASE` statements to determine user relationships, such as distinguishing between 'Client Ordered' and 'Volunteer Delivered', demanded precision. Another challenge was using `LEFT JOIN`, which can return `NULL` values when there’s no match in the right table, requiring special handling. Finally, parameterized queries, which use placeholders, added flexibility but also necessitated ensuring that dynamic values were correctly mapped to the query parameters during execution.
From these experiences, I learned the importance of query optimization by indexing necessary columns to improve performance. Handling `NULL` values properly when using `LEFT JOIN` and thoroughly understanding the database schema and table relationships were also crucial for constructing accurate and efficient queries.