**Project Report**

**Languages and Frameworks Used**

Languages: Python (for backend), SQL (for database interactions), JavaScript (for frontend).

Frameworks: Flask , React

**Schema**

No modifications were made to the schema.

**Additional Constraints, Triggers, Stored Procedures**

None were utilized.

**Main Queries Implemented**

SQL queries are present in psql\_mappings.py:

**get item by id**

SELECT \*

FROM Item

WHERE ItemID = ${ItemID}

**get order by id**

"SELECT \*

FROM Ordered

WHERE orderID = ${orderID}

**get delivery status by orderid**

SELECT \*

FROM delivered

WHERE orderid=${orderID}

**get all pieces locations**

SELECT p.ItemID, p.pieceNum, p.roomNum, p.shelfNum, l.shelfDescription

FROM Piece AS p

NATURAL JOIN location AS l

WHERE p.ItemID=${ItemID}

**get all pieces locations using orderid**

SELECT itemin.itemid,piecenum,orderid,roomnum,shelfnum,shelfdescription

FROM itemin JOIN piece

NATURAL JOIN location ON(itemin.ItemId=piece.ItemID)

WHERE OrderId=${orderID}

**get person by username**

SELECT \*

FROM Person

WHERE userName='${userName}'

**get role by username**

SELECT roleID

FROM Act

WHERE userName='${userName}'

**get distinct item ids in itemin**

SELECT DISTINCT(ItemID)

FROM ItemIn

**get all orders related to user**

SELECT o.OrderId,orderDate,ordernotes,supervisor,client,username AS volunteer, status, date

FROM Ordered AS o

LEFT JOIN Delivered AS d ON o.OrderID=d.OrderID

WHERE supervisor='${username}' OR client='${username}' OR username='${username}'

**get role by orderid and user id**

SELECT supervisor,username AS volunteer, status, date

FROM Delivered AS d LEFT JOIN Ordered AS o ON o.OrderID=d.OrderID

WHERE d.OrderID=${orderID} AND (supervisor='${username}' OR username='${username}')

**get items in order**

SELECT ItemID

FROM itemin

WHERE orderID=${orderID}

**get order by id**

SELECT \*

FROM ordered

WHERE orderID=${orderID}

**get highest orderid**

SELECT max(orderid) AS max order id

FROM Ordered

**get inventory items**

SELECT \*

FROM Item

WHERE ItemId NOT IN (

SELECT distinct(ItemID) from ItemIn)

**get inventory items with category**

SELECT \*

FROM Item

WHERE mainCategory='${mainCategory}' AND ItemId NOT IN (

SELECT distinct(ItemID) from ItemIn)

**get inventory items with subcategory**

SELECT \*

FROM Item

WHERE mainCategory='${mainCategory}' AND subCategory='${subCategory}' AND ItemId NOT IN (

SELECT distinct(ItemID) from ItemIn);

**get volunteer task ranking**

WITH task record(orderID,username) AS ((

SELECT orderID, act.username

FROM Ordered

JOIN Act ON (Ordered.supervisor=Act.username)

WHERE Act.roleid='2')

UNION (

SELECT orderID, act.username

FROM Delivered

Join Act ON (Delivered.username=Act.username)

WHERE Act.roleid='2'))

SELECT COUNT(DISTINCT orderID) AS task count,username

FROM task record

GROUP BY username

ORDER BY COUNT(DISTINCT orderID)

**get volunteer task ranking between dates**

WITH task record(orderID,username) AS ((

SELECT orderID, act.username

FROM Ordered

JOIN Act ON (Ordered.supervisor=Act.username)

WHERE Act.roleid='2' AND (orderDate BETWEEN '${start date}' AND '${end date}'))

UNION

(SELECT orderID, act.username

FROM Delivered

Join Act ON (Delivered.username=Act.username)

WHERE Act.roleid='2' AND (date BETWEEN '${start date}' AND '${end date}')))

SELECT COUNT(DISTINCT orderID) AS task count,username

FROM task record

GROUP BY username

ORDER BY COUNT(DISTINCT orderID)

**insert person**

INSERT INTO person(userName,password,fname,lname,email)

VALUES ('${userName}','${password}','${fname}','${lname}','${email}')

**insert person phone**

INSERT INTO PersonPhone(userName,phone)

VALUES ('${userName}','${phone}')

**insert act**

INSERT INTO Act(userName,roleID)

VALUES ('${userName}','${roleID}')

**insert item**

INSERT INTO Item(ItemID,iDescription,photo,color,isNew,hasPieces,material, mainCategory,subCategory)

VALUES (${ItemID},'${iDescription}','${photo}','${color}',${isNew},${hasPieces}, '${material}','${mainCategory}','${subCategory}')

**insert item without id**

INSERT INTO Item(iDescription,photo,color,isNew,hasPieces,material,mainCategory, subCategory)

VALUES ('${iDescription}','${photo}','${color}',${isNew},${hasPieces},'${material}', '${mainCategory}','${subCategory}')

RETURNING ItemID

**insert donatedby**

INSERT INTO DonatedBy(ItemID,userName,donateDate)

VALUES (${ItemID},'${userName}','${donateDate}')

**insert piece**

INSERT INTO Piece(ItemID,pieceNum,pDescription,length,width,height,roomNum, shelfNum,pNotes)

VALUES (${ItemID},${pieceNum},'${pDescription}',${length},${width},${height}, ${roomNum},${shelfNum},'${pNotes}')

insert itemin

INSERT INTO ItemIn(ItemID,orderID,found)

VALUES (${ItemID},${orderID},${found})

**insert ordered**

INSERT INTO Ordered(orderID,orderDate,orderNotes,supervisor,client)

VALUES (${orderID},'${orderDate}','${orderNotes}','${supervisor}','${client}')

**update person**

UPDATE person SET userName='${userName}', password='${password}', fname='${fname}', lname='${lname}', email='${email}'

WHERE userName='${userName}'

**update person phone**

UPDATE PersonPhone SET userName='${userName}', phone='${phone}'

WHERE userName='${userName}' AND phone='${phone}'

**update act**

UPDATE Act SET userName='${userName}', roleID='${roleID}'

WHERE userName='${userName}' AND roleID='${roleID}'

**update ordered**

UPDATE Ordered SET orderID=${orderID}, orderDate='${orderDate}', orderNotes='${orderNotes}', supervisor='${supervisor}',client='${client}'

WHERE orderID=${orderID}

**update itemin**

UPDATE ItemIn SET ItemID=${ItemID}, orderID=${orderID},found=${found}

WHERE orderID=${orderID} AND ItemID=${ItemID}

**update order status by orderid**

UPDATE Delivered SET status='${status}', date='${date}'

WHERE orderID=${orderID}

**Difficulties Encountered and Lessons Learned**

**Challenges**

* Tracing dependencies between Python modules and SQL queries.
* Using Cookies to store temporary data
* Error Handling of queries while sending and receiving payloads
* Testing for edge cases revealed problems

**Lessons**

* Collaborating and working on the project at the same time
* Building reusable components, such as query functions, helped reduce repetitive code.
* Detailed error logs helped to fix bugs faster

**Team Contributions**

Sai Akilesh Venigalla: Implemented features 1,2,5,6

Akilan Yohendiran: Implemented features 3,7,8

Kanishk Aggarwal: Implemented features 4,9,10