WelcomeHome

CS-GY 6083: Principles of Database Systems

Final Project Report

Bhaktram Jain (bj2411)
Aditi Aatmaja (aa12918)

Languages and Frameworks Used

1. **Languages:**
    - **Python**: Used for server-side scripting, building routes, and implementing application logic.
    - **SQL**: Used for designing and querying the database.
    - **HTML/CSS**: Used for building the user interface and styling.
    - **JavaScript**: Used for dynamic content, particularly for filtering categories and subcategories dynamically.

2. **Frameworks:**
    - **Flask**: Used for building the web application and managing HTTP requests.
    - **MySQL**: Used as the relational database management system to store and manage project data.
    - **Jinja2**: Used for rendering dynamic HTML templates.

3. **Tools**:

    - MySQL Workbench
    - Visual Studio Code (VSCode)
    - Git

Changes Made to the Schema

We didn’t made and changes to the schema (adhered to the schema ‘Project Schema-v2’ outlines in project definitions) but added a new functionality:

Added a **holding location** feature:

- **Purpose**: To designate items as "ready for delivery" by setting their roomNum and shelfNum to -1.
- **Modification**: Added functionality to update Piece table entries with a holding location when an order is marked as "prepared."

Additional Constraints, Triggers, Stored Procedures

1. **Constraints: (as in the schema, no new constraints)**
    - Foreign key relationships between Item, Piece, Category, Location, and other tables to ensure data integrity.
    - Unique constraints in tables like Person and Item to avoid duplicates.
2. **Triggers:**
    - None implemented.
3. **Stored Procedures:**
    - None used explicitly; all logic was implemented in Python and executed via SQL queries.

SQL Queries

Feature 1: Login and User Session Handling

<u>Login (/login)</u>

Purpose: Authenticate users by validating their credentials and starting a session

(i) Fetching User Data	(ii) Fetching the Role

SELECT *
FROM Person
WHERE userName = %s;
	SELECT Role.rDescription
	FROM Act
	JOIN Role ON Act.roleID = Role.roleID
	WHERE Act.userName = %s

Hashing Logic : The password is hashed using a library like <u>bcrypt</u> or <u>werkzeug.security</u>:

```
from werkzeug.security import check_password_hash
if check_password_hash(user['password'], provided_password):
    # Authentication successful
```

<u>Register (/register)</u>

Purpose: Register new users by storing hashed passwords ;

Insert user details into the Person table:

```
INSERT INTO Person (userName, password, fname, lname, email)
VALUES (%s, %s, %s, %s, %s);
```

Assign a role:

(i) to the user in the Act table	(ii) to a session

INSERT INTO Act (userName, roleID)
VALUES (%s, %s);
	session['user_id'] = user['userName']
	session['username'] = user['userName']
	session['role'] = role['rDescription'].lower() if role else 'no role'

Logout (/logout)

Purpose: Clear the current session and redirect the user to the login page.
No specific SQL query is required for this; it only clears session variables in the Flask application.

Feature 2 : Find Item (/find_item)

Purpose: Retrieve information about a specific item and its piece locations.

SQL Queries:

Retrieve item details:

```
SELECT itemID, iDescription
FROM Item
WHERE itemID = %s;
```

Retrieve the locations of all pieces of the item:

```
SELECT Piece.pieceNum, Location.roomNum, Location.shelfNum
FROM Piece
JOIN Location ON Piece.roomNum = Location.roomNum AND Piece.shelfNum =
Location.shelfNum
WHERE Piece.itemID = %s;
```

Feature 3 : Find Order (/find_order)

Purpose: Retrieve details of a specific order and its associated items.

Retrieve order details:

```
SELECT *
FROM Ordered
WHERE orderID = %s;
```

Retrieve items in the order:

```
SELECT i.ItemID, i.iDescription
FROM ItemIn ii
JOIN Item i ON ii.ItemID = i.ItemID
WHERE ii.orderID = %s;
```

Fetch piece locations for each item:

```
SELECT p.pieceNum, p.roomNum, p.shelfNum
FROM Piece p
WHERE p.ItemID = %s;
```

Feature 4 : Accept Donation (/accept_donation)

Purpose: Allow staff to accept a donated item and add it to the inventory.

SQL Queries:

Insert the donated item into the Item table:

```
INSERT INTO Item (iDescription, color, isNew, hasPieces, material, mainCategory,
subCategory)
VALUES (%s, %s, %s, %s, %s, %s, %s);
```

Insert the associated piece into the Piece table:

```
INSERT INTO Piece (itemID, pieceNum, pDescription, length, width, height, roomNum,
shelfNum, pNotes)
VALUES (%s, 1, %s, %s, %s, %s, %s, %s, %s);
```

Log the donation in the DonatedBy table:

```
INSERT INTO DonatedBy (itemID, userName, donateDate)
VALUES (%s, %s, NOW());
```

Additional Features :

Feature 5 : Start an Order (/start_order)

Purpose: Create a new order for Clients

Checking if the Client exists

```
SELECT COUNT(*) AS count
FROM Person p
JOIN Act a ON p.userName = a.userName
WHERE p.userName = %s AND a.roleID = 'client';
```

Inserting the order in Ordered table

```
INSERT INTO Ordered (orderDate, supervisor, client, orderNotes)
VALUES (CURRENT_DATE(), %s, %s, %s);
```

Feature 6 : Add to Order (/add_to_order)

Purpose: Add items to a currently active order.

Fetch Order Details

```
SELECT orderID, orderDate, orderNotes, supervisor, client
        FROM Ordered
        WHERE orderID = %s
```

Insert item into the order:

```
INSERT INTO ItemIn (ItemID, orderID, found)
VALUES (%s, %s, FALSE);
```

Fetch Items if Category and subcategory are selected

```
SELECT ItemID, iDescription
            FROM Item
            WHERE mainCategory = %s AND subCategory = %s
            AND ItemID NOT IN (SELECT ItemID FROM ItemIn)
```

Feature 7 : Prepare Order (/prepare_order)

Purpose: Mark an order as prepared for delivery.

Update item locations to the holding location:

```
UPDATE Piece
SET roomNum = -1, shelfNum = -1
WHERE ItemID IN (
   SELECT ItemID FROM ItemIn WHERE orderID = %s
);
```

Log preparation in the Delivered table:

```
INSERT INTO Delivered (userName, orderID, status, date)
VALUES (%s, %s, %s, CURRENT_DATE());
```

Feature 8 : User Tasks (/user_tasks)

Purpose: Display tasks associated with the current user.

Fetch Orders where User is Client

```
SELECT o.orderID, o.orderDate, o.orderNotes, o.supervisor
            FROM Ordered o
            WHERE o.client = %s
```

Fetch Orders where User is a Supervisor (Staff)

```
SELECT o.orderID, o.orderDate, o.orderNotes, o.client
            FROM Ordered o
            WHERE o.supervisor = %s
```

Fetch tasks from the Delivered table:

```
SELECT o.orderID, o.orderDate, o.orderNotes, d.status, d.date
FROM Delivered d
JOIN Ordered o ON d.orderID = o.orderID
WHERE d.userName = %s;
```

Bonus Extra Feature :

Feature 9 : Ranking System (/rank_categories)

Purpose: Retrieve the most popular category and subcategory based on the number of orders.

SQL Queries:

Fetch the most popular categories:

```
SELECT
    c.mainCategory,
    c.subCategory,
    COUNT(*) AS orderCount
FROM
    ItemIn ii
JOIN
    Item i ON ii.ItemID = i.ItemID
JOIN
    Category c ON i.mainCategory = c.mainCategory AND i.subCategory = c.subCategory
JOIN
    Ordered o ON ii.orderID = o.orderID
WHERE
    o.orderDate BETWEEN %s AND %s
GROUP BY
    c.mainCategory, c.subCategory
ORDER BY
    orderCount DESC
LIMIT 5;
```

Difficulties Encountered, Lessons Learned

1. **Challenges:**
    - Foreign key constraints caused issues when inserting data into dependent tables.
    - Debugging errors caused by missing data, especially in Piece and Item tables.
    - Handling edge cases, such as missing or invalid data during order preparation and item addition.

2. **Lessons Learned:**
    - Ensure consistent relationships between tables by validating data before insertion.
    - Build modular, reusable functions for common database operations to minimize redundant code.
    - Comprehensive testing is critical to catch and resolve issues early.

Team Member Contributions

- **Member 1 - Bhaktram Jain**
    - Main Features
      1. Developed - **Find Order item**
      2. Developed **Accept Donation**
    - Additional Features Worked on **Prepare Order** and **User Tasks**
    - Implemented bonus - **Rank System** for categories.
    - Worked on final testing and test cases for the application

- **Member 2 - Aditi Aatmaja**
    - Designed and implemented database schema.
    - Main Features
      1. Developed **Login and User Session** Handling
      2. Developed **Find Single Item** Functionality
    - Additional Features Worked on **Start an Order** and **Add to Order** features
    - Assisted in testing final version of application