Project 3 report

Han Wen

- I used Python and SQL as the back-end language and Pymysql as the database framework, html and css as the front-end language and I used Flask as the front-end framework as well.
- I add a new attribute "role" to the table "user" in order to verify the identity of users when they register or log in.

I add a new table "orders" containing attributes like orderId, customerId, orderDate and status to verify concrete information when starting a new order and save status information to save it in a session variable.

I add a new table "Items" containing attributes like itemId, description, price, stockQuantity, donator, ordered and categoryId to record an item that user donates and the information about it and check if an item has been ordered when shopping.

I add a new table "OrderItems" containing attributes like orderId, itemId, quantity, category and the orderId and itemId are the foreign keys. This table is built to record a concrete item information that ordered by a user.

I also add a new table "Categories" to help classify the items and it has some initial categories like ('chair'), ('bed'), ('desk'), ('lamp'), ('closet'), ('TV'), ('sofa');

- Main queries:

```
#Query the Items table and include donor information
'SELECT i.itemId, i.description, i.price, i.stockQuantity, i.donator '
    'FROM Items i'
#Find single item
'SELECT itemId, description, price, stockQuantity, donator '
        'FROM Items '
        'WHERE itemId = %s', (item_id,)'
#Query the items in the order and their locations
SELECT i.itemId, i.description, i.price, oi.quantity, i.location
            FROM OrderItems oi
            JOIN Items i ON oi.itemId = i.itemId
            WHERE oi.orderId = %s
#Obtain the category name corresponding to the item
SELECT c.name AS categoryName
            FROM Items i
            JOIN Categories c ON i.categoryId = c.id
            WHERE i.itemId = %s
#Add the item to the order with its category
```

```
INSERT INTO OrderItems (orderId, itemId, quantity, category)
            VALUES (%s, %s, %s, %s)
#Mark the item as ordered in the Items table
'UPDATE Items SET ordered = TRUE WHERE itemId = %s',
            (item_id,)
#Fetch available items that are not already ordered in this order and not
marked as ordered
SELECT i.itemId, i.description, c.name AS categoryName
            FROM Items i
            JOIN Categories c ON i.categoryId = c.id
            LEFT JOIN OrderItems oi ON i.itemId = oi.itemId AND
oi.orderId = %s
            WHERE oi.itemId IS NULL AND i.ordered = FALSE AND
c.name = %s
#View orders
SELECT o.orderId, o.orderDate, o.status, i.itemId, i.description,
oi.quantity, oi.category
      FROM Orders o
      JOIN OrderItems oi ON o.orderId = oi.orderId
      JOIN Items i ON oi.itemId = i.itemId
      WHERE o.orderId = %s
```

- Difficulties Encountered

1. Database Connection Management:
   One of the initial challenges was establishing a stable connection between the application and the MySQL database using PyMySQL. Issues such as incorrect configurations or outdated drivers could lead to intermittent connectivity problems.
2. Integration Between Flask and Front-End Technologies:
   Integrating Flask with HTML and CSS posed its own set of challenges, especially when trying to pass dynamic content from the server-side to the client-side. Learning how to effectively use Flask's templating engine Jinja2 was crucial for rendering dynamic pages.
3. Optimizing Query Performance:
   Writing efficient SQL queries to handle large datasets without compromising performance was another challenge. It involved understanding indexing, query optimization techniques, and sometimes rethinking the database schema to better support the application's requirements.

- Lessons Learned

1. The Importance of Planning and Design:

A well-thought-out database schema and application architecture are foundational to building a robust application. Taking time upfront to plan the system design paid off in terms of reduced bugs and easier maintenance later on.

2. Understanding Frameworks and Libraries:
Deepening my knowledge of Flask, PyMySQL, and related tools allowed me to leverage their full capabilities and write more efficient code. Understanding the underlying principles of these frameworks also made debugging much easier.

3. Security Best Practices:
Implementing security best practices early in the development cycle helped prevent common vulnerabilities. This included parameterized queries to avoid SQL injection and secure password storage methods.

4. Performance Optimization:
Gaining insight into optimizing SQL queries and database interactions has been invaluable. Profiling tools and understanding how indexes work significantly improved the application’s performance.

- I finished the whole project all by myself.

Han Wen
12/2024