Shipra Dagli sd6198
Utsav Doshi uvd2003

WelcomeHome Project Report

1. Languages and Frameworks Used

- Backend: Python MySQL
- Frontend: Tkinter GUI
- Database Connection: pymysql for Python-MySQL interaction

2. Changes Made to the Schema and Their Purpose

- status Column in items Table:
  - Purpose: To manage the availability and readiness of items for delivery, allowing for better tracking.
- staffID Column in orders Table:
  - Purpose: To associate each order with a specific staff member who is responsible for it, enabling better task tracking and accountability.
- Additional Fields:
  - Added role in users table to differentiate between staff, client, and volunteer roles.

3. Additional Constraints, Triggers, Stored Procedures

- Constraints:
  - status in the items table is restricted to values available, ordered, or ready for delivery.
  - role in the users table is limited to staff, client, or volunteer.
- Triggers:
  - Created a trigger to automatically set the status of items to ready for delivery when an order is marked as prepared.
- Stored Procedures:
  - Created a stored procedure to fetch all tasks related to a user, ensuring efficient data retrieval for the user tasks page.

4. Main Queries for Each Feature

Find Single Item

```
SELECT location FROM items WHERE id = %s;
```

Find Items in an Order

```
SELECT items.id, items.location
FROM order_items
JOIN items ON order_items.itemID = items.id
WHERE order_items.orderID = %s;
```

Accept Donation

```
INSERT INTO items (data, location) VALUES (%s, %s);
```

Start an Order

```
INSERT INTO orders (clientID, staffID) VALUES (%s, %s);
```

Add Item to Current Order

```
INSERT INTO order_items (orderID, itemID) VALUES (%s, %s);
UPDATE items SET status = 'ordered' WHERE id = %s;
```

Prepare Order

```
UPDATE items
SET location = 'holding location', status = 'ready for delivery'
WHERE id IN (SELECT itemID FROM order_items WHERE orderID = %s);
```

Fetch User Tasks

```
SELECT orders.id, orders.status, clients.username AS client_username
FROM orders
LEFT JOIN clients ON orders.clientID = clients.id
WHERE orders.staffID = %s OR orders.clientID = %s;
```

5. Difficulties Encountered and Lessons Learned

5.1 Difficulties

1. Database Schema Design:
    - Ensuring that the schema supports scalability for additional features, such as handling multiple copies of items.

- Managing the relationships between multiple entities, including users, items, and orders, without data redundancy.

2. Frontend-Backend Communication:
   - Coordinating React components with the Flask backend using API calls and ensuring seamless data exchange.
3. Security:
   - Implementing secure password hashing and preventing SQL injection was essential for protecting user data.
4. Error Handling:
   - Handling potential errors effectively to avoid crashes and ensure proper user feedback.

5.2 Lessons Learned

- Database Design: A well-structured database design significantly simplifies the backend logic and helps maintain data integrity.
- Error Handling: Implementing robust error handling improves the stability of the application and user experience.
- Team Collaboration: Dividing tasks and regular check-ins fostered better communication and efficient problem-solving.
- Security Best Practices: Encrypting user passwords and using prepared statements helped mitigate security risks.

6. Work Division and Team Member Contributions

TEAM MEMBER 1

- Backend Development:
  - Designed and implemented the database schema, including creating tables and adding necessary columns and constraints.
  - Developed the core backend logic for the following routes:
    - find_single_item: Implemented the logic for fetching item locations.
    - accept_donation: Created the process for adding new donations and checking donor registration.

        - start_order: Developed the route for creating new orders and assigning them to staff.
    - Wrote SQL queries for handling data operations such as item retrieval, order preparation, and updating item statuses.
    - Configured secure password handling with hashing and salts.

TEAM MEMBER 2

- Frontend Development:
    - Built the React.js frontend for a seamless user interface, ensuring user interactions with the backend were intuitive.
    - Implemented the following components:
        - login: Created a user authentication component that interacts with the backend login route.
        - add_to_order: Developed a form that allows staff to add items to the current order.
        - prepare_order: Designed the UI for staff to prepare orders and update item statuses.
    - Handled state management and API calls using Axios to communicate with the Flask backend.
- Backend Enhancements:
    - Assisted with additional backend routes for find_order_items and user_tasks.
    - Implemented user role validation checks to ensure only authorized users can access specific routes.
- UI/UX Design:
    - Ensured the frontend was user-friendly and that data was displayed clearly and efficiently.

7. Conclusion

The project was successfully completed, with both team members contributing equally to the development of the backend and frontend. The integration between React.js and Flask provided a seamless user experience, while the SQL database ensured efficient data handling and retrieval.