Team ORCHID: Library Database Demo Instructions

This document covers the necessary information to run the remote demonstration, such as database access, example queries, and front-end features.

This document is intended for EECS 447 review only. All pertinent information for login is included in the assignment submission rather than within this document, as this is public.

Accessing the database

To access the database, there are two options which both require a KU login:

a. Access while on the JAYHAWK network (on-campus).
b. Access while off-campus using the KU Anywhere VPN. More information and installation/connection instructions via the link.

Once you are connected to KU's network, you can login to the database by going to mysql.eecs.ku.edu and using the login credentials provided. You can then browse the database through the backend.

Example queries

Here are some example queries to run and examine.

List items with their item types, genres, and primary contributor

```
SELECT c.item_id, it.item_type_name, i.title, g.genre_name,
MIN(c.contribution_id), cr.first_name, cr.last_name, r.role_name,
i.description

FROM contribution c

JOIN contributor cr ON cr.contributor_id = c.contributor_id

JOIN contribution_role r ON c.role_id = r.role_id

JOIN item i ON c.item_id = i.item_id

JOIN genre g ON g.genre_id = i.genre_id

JOIN item_type it ON i.item_type_id = it.item_type_id

GROUP BY c.item_id ORDER BY i.item_id;
```

List all books by a specific author

```
SELECT i.ISBN, cr.last_name, cr.first_name, i.title, i.description,
g.genre_name, it.item_type_name

FROM item i

JOIN contribution c ON i.item_id = c.item_id

JOIN contributor cr ON c.contributor_id = cr.contributor_id

JOIN genre g ON i.genre_id = g.genre_id

JOIN item_type it ON i.item_type_id = it.item_type_id

# Change last name for other authors.

WHERE cr.last_name = "Novik"

AND it.item_type_name = "Book";
```

List of movies with their actors

```
SELECT i.item_id, i.title, i.description, GROUP_CONCAT(cr.first_name,
" ", cr.last_name) as actors

FROM item i

LEFT JOIN contribution c ON c.item_id = i.item_id

LEFT JOIN contributor cr ON cr.contributor_id = c.contributor_id

WHERE c.role_id = "5"

GROUP BY i.item_id;
```

Items per publication year

```
SELECT i.publication_year, COUNT(i.item_id) FROM item i

GROUP BY i.publication_year;
```

Display all items currently available

```
SELECT i.title, i.description, i.quantity_available FROM item i
WHERE i.quantity_available != 0;
```

Display count of restricted accounts and their card numbers

```
SELECT COUNT(restricted) as "Amount", GROUP_CONCAT(card_number) as
"Card Numbers" FROM account

WHERE restricted = 1 GROUP BY restricted;
```

Items due within the next week that haven't been returned

```
SELECT * FROM loan l WHERE l.due_date BETWEEN (CURRENT_DATE) AND
(CURRENT_DATE + INTERVAL 7 DAY) AND l.return_date is NULL;
```

Overdue items and the accounts who borrowed them

```
SELECT i.item_id, i.title, l.due_date, l.return_date, a.card_number,
a.last_name FROM loan l

JOIN item i ON l.item_id = i.item_id JOIN account a ON l.account_id =
a.account_id

WHERE ((l.due_date < CURRENT_DATE) AND (l.return_date is NULL)) OR
(l.due_date < l.return_date);
```

Items with the amount of times that they have been loaned

```
SELECT i.title, loan_count FROM item i

JOIN (SELECT l.item_id, COUNT(l.loan_id) as loan_count FROM loan l

GROUP BY l.item_id) AS l2 ON l2.item_id = i.item_id;
```

Items by title with their last borrowed date

```
SELECT i.item_id, i.title, MAX(l.loan_out_date) as last_loaned

FROM item i

JOIN loan l ON l.item_id = i.item_id

GROUP BY l.item_id

ORDER BY l.loan_out_date DESC;
```

Most popular items with students

```
SELECT i.title, COUNT(l.loan_id) AS times_loaned FROM item i

JOIN loan l ON l.item_id = i.item_id

JOIN account a ON a.account_id = l.account_id

WHERE a.account_type_id = 3

GROUP BY l.item_id

ORDER BY COUNT(l.loan_id) DESC;
```

Amount of a contributor's works in the library and times their works have been loaned

```
SELECT cr.contributor_id, cr.first_name, cr.last_name,
COUNT(DISTINCT(c.contribution_id)) as num_works, COUNT(l.loan_id) as
times_loaned

FROM contributor cr

LEFT JOIN contribution c ON c.contributor_id = cr.contributor_id

LEFT JOIN loan l ON c.item_id = l.item_id

GROUP BY c.contributor_id

ORDER BY COUNT(c.contribution_id) DESC;
```

Items loaned in the past month by popularity

```
SELECT COUNT(l.loan_id), i.title FROM item i

JOIN loan l ON l.item_id = i.item_id

WHERE l.loan_out_date > (CURRENT_DATE - INTERVAL 1 MONTH)

GROUP BY l.item_id

ORDER BY COUNT(l.loan_id) DESC;
```

Find all accounts that had overdue loans in the past two months

```
SELECT a.account_id, a.card_number, a.first_name, a.last_name,
COUNT(l.loan_id)

AS overdue_loans FROM account a JOIN loan l ON a.account_id =
l.account_id

WHERE ((l.return_date > l.due_date) OR (l.due_date < CURRENT_DATE and
l.return_date is NULL))

AND l.loan_out_date > (CURRENT_DATE - INTERVAL 2 MONTH) GROUP BY
a.account_id;
```

Count current, unreturned loans to verify quantity_available is accurate

```
SELECT * FROM item i

JOIN (SELECT l.item_id, COUNT(l.loan_id) as loan_count FROM loan l
WHERE l.return_date is NULL GROUP BY l.item_id) AS l2 ON l2.item_id =
i.item_id

WHERE i.quantity_available != (i.total_quantity - l2.loan_count);
```

Note: This is to make sure the database is up-to-date. This should return 0 rows.

All genres that have never been loaned

```
SELECT genre_name FROM genre
WHERE NOT EXISTS
(SELECT * FROM loan JOIN item ON loan.item_id = item.item_id WHERE
item.genre_id = genre.genre_id);
```

Items not loaned in the past year

```
SELECT title FROM item WHERE NOT EXISTS

(SELECT * FROM loan WHERE loan.item_id = item.item_id AND
(YEAR(CURRENT_DATE) - loan.loan_out_date) < 1);
```

Genre Report

```
WITH loan_counts AS ( SELECT g.genre_id, g.genre_name, i.item_id,
i.title, COUNT(l.loan_id) AS loan_count, ROW_NUMBER() OVER (PARTITION
BY g.genre_id ORDER BY COUNT(l.loan_id) DESC) AS rn FROM genre g JOIN
item i ON g.genre_id = i.genre_id LEFT JOIN loan l ON l.item_id =
i.item_id GROUP BY g.genre_id, g.genre_name, i.item_id, i.title )

SELECT g.genre_name, COUNT(i.item_id) AS num_items,
AVG(YEAR(CURRENT_DATE) - i.publication_year) AS avg_age,
COUNT(l.loan_id) AS num_loans, COUNT(CASE WHEN a.account_type_id = 1
THEN 1 END) AS num_adult_borrowers, COUNT(CASE WHEN a.account_type_id
= 2 THEN 1 END) AS num_child_borrowers, COUNT(CASE WHEN
a.account_type_id = 3 THEN 1 END) AS num_staff_borrowers, lc.title AS
most_loaned_book, lc.loan_count AS most_loaned_count FROM genre g
JOIN item i ON g.genre_id = i.genre_id LEFT JOIN loan l ON l.item_id
= i.item_id LEFT JOIN account a ON l.account_id = a.account_id LEFT
JOIN ( SELECT genre_id, title, loan_count FROM loan_counts WHERE rn =
1 ) lc ON g.genre_id = lc.genre_id GROUP BY g.genre_id, g.genre_name,
lc.title, lc.loan_count;
```

Setting up the front-end

To set up the frontend, first ensure that node, npm, and mysql are up to date.

First, download the repository.

After doing so, you'll need to navigate to the *library* folder and create a new file named *.env.local*, which will follow the same format as the *.env.example* file.

You will then fill out the DB_User, DB_Password, and DB_Name fields with the user and password fields being your login criteria and the name being the name of the database you want to connect to.

You'll then open up two terminals. The first terminal will be used to create a tunnel. The command will be *ssh –L 3307:mysql.eecs.ku.edu:3306 [your KU username]@cycle3.eecs.ku.edu*.

After you successfully log in, you'll then move to the second terminal. Change your directory to the library folder, and then use the command *npm run dev*.

After these steps, you should be able to go to http://localhost:3000 to access the frontend.

Front-end features

There are three main pages that compose the frontend:

- The *Login Page* which handles user login based on their library card number.
- The *Account Page* which handles account specific details and functions such as account information, loan history, active reservations, and account information updates. It also contains the management tab for staff accounts.
- The *Inventory Page* which allows for users to search for items in the library, loan them if available, and reserve them if not.

Login Page

The Login Page allows the user to sign in via their library card number.

After doing so, their information is temporarily stored to retrieve their account details as well as making reservations under their name through the inventory page. After signing in, the user is redirected to their account page via the Go to My Account button that appears after a successful login.

Below are some demonstration accounts for testing the frontend features.

- **Staff account:** Peter b. Jones
  - *Card Number:* 1523510581525
- **Student account:** Sophia Garcia
  - *Card Number:* 1523510581528
- **Adult Account:** Noah Patel
  - *Card Number:* 1523510581529

Account Page

The account page contains multiple user functions, which are all displayed as different views that can be selected from the left divisor.

Account details

Displays the account's details, such as name, card number, and contact information.

Loan History

Displays the user's loan history as well as allowing them to return loaned items if not yet returned.

Each loan record displays relevant item information, as well as the loan period and the date when they were returned.

Reservations

Displays the user's active reservations as well as allowing them to cancel any active reservations.

Each reservation record displays relevant item information, as well as the reservation window.

Account Settings

Allows the user to update their personal information, such as their address, phone number, and name.

It also allows the user to delete their account after confirmation through a warning prompt.

Management (Staff Only)

This tab allows staff accounts to perform account lookups and item upload.

The lookup displays basic account Information, loan history, and active reservations.

To upload an item, title, description, publication date, total quantity, quantity available, publication year, and reservation amount (please always set this value to 0) are required. Every other category should be item type dependent.

Inventory Page

The Inventory page allows the user to search for items within the library's inventory.

The main search input searches by item title, while the advanced search options allow for a more refined search.

Advanced Search Fields

Advanced search fields allow the user to refine their item search by adding more search parameters. These are:

- Author Name (either first or last name; not both)

- Release year
- ISBN
- Issue Number
- Genre
- Age Rating

These all work both additively and independently of each other.

Filter By and Sort By

Below the advanced search drop down menu there are two additional picklists: *Filter By,* which allows the user to filter by the item type, and *Sort By*, which allows the user to sort the items by predefined orders such as alphabetical, reverse alphabetical, oldest, and newest items based on release year.

Search Results

Displays the results of the user's search. Each result is stored within a container that displays item title, description, main contributor, and amount available.

For additional information, each container has a *More Info* button in the top right corner. This displays significantly more information, such as genre, additional contributors, publication date, and more.

If an item currently is available, the *Make a* Loan button allows the user to make a loan. Otherwise, the button is not visible and is replaced instead by the *Make Reservation* button.

The user makes the loan through a popup window. The earliest loan day available is the current date, while there is a max 2-week loan period enforced.

The reservation functions in a similar way; however, the earliest reservation date is based on the earliest loan return date present within the database for the desired item. After that minimum date, there is a maximum 4-week reservation window to account for the potential of an item not being returned by their due date.