

CPSC 2221 - Database Systems (Spring 2018)

Group Project - Implementation of a Relational Database

Project Title:	Langara Book Trade Club
Project Milestone:	Milestone 5a – Complete Project Files

#	Student Name	Student ID	Email Address
1	Elían Figueiredo	100290512	elian_gaspar@hotmail.com
2	Caue Pinheiro	100302918	cauehier@gmail.com
3	Artyom Lunyakin	100288803	artyomluu@gmail.com

By keying our names and student IDs in the above table, we certify that the work submitted with this cover page was performed solely by those whose names and student IDs are included above.

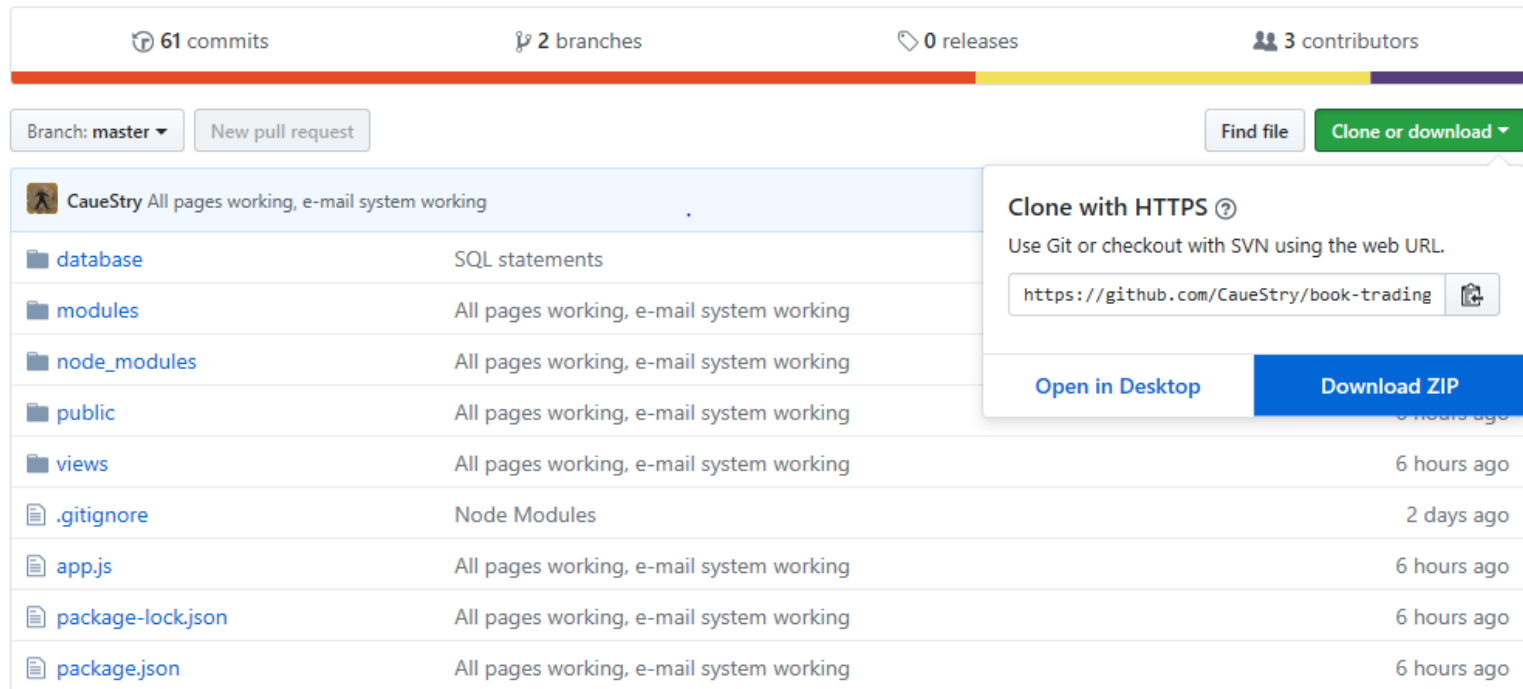
Also, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Langara College.

Source code

The source code can be easily acquired on GitHub, please use the link below:

<https://github.com/CaueStry/book-trading-club>

A book trading website



Branch: master New pull request Find file Clone or download

CaueStry All pages working, e-mail system working

database	SQL statements	
modules	All pages working, e-mail system working	
node_modules	All pages working, e-mail system working	
public	All pages working, e-mail system working	
views	All pages working, e-mail system working	6 hours ago
.gitignore	Node Modules	2 days ago
app.js	All pages working, e-mail system working	6 hours ago
package-lock.json	All pages working, e-mail system working	6 hours ago
package.json	All pages working, e-mail system working	6 hours ago

How to install your application (step-by-step)

Once the code has been downloaded and unzipped:

1. Install Node.js
 - a. Latest version can be found here: <https://nodejs.org/en/>
2. Make sure you have MySQL installed on localhost, port 3306 (default MySQL port) with the root user properly configured.

3. Create database objects:

- a. Inside repository's root folder, there's a folder called "database" as can be seen on image above. All scripts needed to create the database and schema are inside the file "tables.sql".

- i. There is no need to create the schema, simply execute the commands inside 'tables.sql' and the database will be ready to use.

To execute the commands on the terminal type:

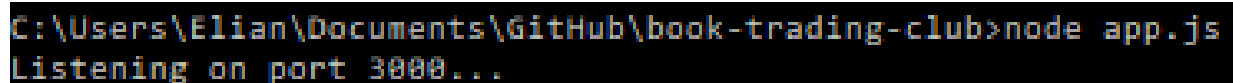
```
mysql < tables.sql -u root -p
```

[Enter the root password]

4. Now that we have Node.js installed and database ready, we just need to run the server and the application will be ready, here's how:

- a. Open command prompt, go to the root folder of the project and run the following command: **node app.js**

- i. You're going to see a message saying "Listening on port 3000..." meaning the server is ready and running.



```
C:\Users\Elian\Documents\GitHub\book-trading-club>node app.js
Listening on port 3000...
```

- b. Server is running, now we can access the application:

- i. Open your browser and type: <http://localhost:3000>

- ii. This will take you to login page, your credentials are: e-mail: 'jchu24@d2l.langara.bc.ca', password: '123'.

- c. Feel free to explore the application, create more accounts and test the functionalities.

- d. To stop the server, on command line, press 'Control + C' or 'Command + C' for MacOS.

Script that could be used to create all tables and data in the database

The Scripts can be found within the database folder: book-trading-club/database/tables.sql.

A short description of what the project accomplished

We managed to implement a fully functional web app, using modern technologies and practices. The Book Trading Club has a secure storage of credentials implementing a Hash and Salt algorithm developed by us to store the password and the salt as a 64byte hash, securing it from brute force attacks using hash lists or database break-ins. We also implemented the entire web server using Node.JS and the Express framework, along with several node modules to back up the functionalities. The database was made using MySQL and normalized to BCNF. The app also implements an e-mail system to keep the users aware of changes on their requests.

Every page loads/writes content from/to the database and is rendered dynamically using a view engine called EJS (Embedded JavaScript).

Every user can see the available featured books, request them, cancel requests, approve requests from other users or reject them. When a request is approved, both users receive an e-mail containing the information to contact their trade partner, allowing them to continue the negotiation.

A list of the SQL queries used

1. [1 Mark] Projection query:

View the title of all books: `SELECT title FROM book;`

2. [1 Mark] Selection query:

View profile information of current logged user:

```
SELECT langara_id, first_name, last_name, email FROM sys_user
WHERE langara_id = (SELECT langara_id FROM sys_user WHERE
sys_user.email='elian_gaspar@hotmail.com');
```

3. [1 Mark] Join query:

View information on books uploaded by current logged user:

```
SELECT OC.copy_id AS copy, OC.owner_langara_id AS bOwner, OC.book_price
AS bPrice, OC.user_image_url AS bUrl, B.title, B.author
FROM owned_copy OC INNER JOIN book B ON OC.book_id = B.isbn13
WHERE OC.owner_langara_id = (SELECT langara_id FROM sys_user WHERE
sys_user.email='elian_gaspar@hotmail.com');
```

4. [1 Mark] Division query:

Select titles for non-requested books owned by a specific user:

```
SELECT b.title FROM owned_copy oc INNER JOIN book b ON oc.book_id =  
b.isbn13 WHERE oc.owner_langara_id = (  
    SELECT langara_id FROM sys_user WHERE sys_user.email='${email}'  
)  
AND b.title NOT IN (  
    SELECT b.title FROM owned_copy oc INNER JOIN book b ON  
oc.book_id = b.isbn13 WHERE oc.owner_langara_id = (  
    SELECT langara_id FROM sys_user WHERE  
sys_user.email='${email}'  
)  
AND oc.requested_by_langara_id IS NOT NULL  
);
```

5. [2 Mark] Aggregation query:

Count how many books have been uploaded:

```
SELECT COUNT(isbn13) FROM book;
```

See the average price of all books:

```
SELECT ROUND(AVG(book_price), 2) AS AvgAllBooks FROM owned_copy;
```

6. [1 Mark] Nested aggregation with group-by:

```
SELECT owner_langara_id, COUNT(copy_id) AS totalBooks, AVG(book_price) AS  
avgPrice  
FROM owned_copy  
WHERE owner_langara_id = (SELECT langara_id FROM sys_user  
WHERE sys_user.email='elian_gaspar@hotmail.com');  
  
GROUP BY owner_langara_id;
```

7. [2 Marks] Delete operation: Implement a cascade-on-delete situation.

```
DELETE FROM sys_user WHERE email = 'john.doe@gmail.com';
```

By deleting user, all books uploaded by this user will also be deleted as per cascade constrain.

8. [1 Marks] Update Operation:

User can select which book they don't want anymore and cancel the request:

```
UPDATE owned_copy SET requested_by_langara_id=NULL WHERE copy_id=3;
```

9. [3 Bonus Marks] Extra features:

Application has been developed using Bootstrap and EJS.