**Amar Harris (1310247)**

**Library Management System Project**

**CIS 3368**

**Professor Gandhimathi Velusamy**

**University of Houston**

Link to github repo 🡪 <https://github.com/amarmharris/CougarLibraryProject_Amar.git>

# Introduction

This software design project focuses on the connection between database design and implementation into application logic. The project objective is to create a library database application using Java and SQL. Core functionality for the application should include:

* Create account
  + Accounts can vary from Under-graduate, Graduate, Faculty, Staff, and admin
* View books/media items from database
* Checkout items from database
  + Users should be limited to the number of items based on their account
* View checked-out items per user
* Return checked-out item per user

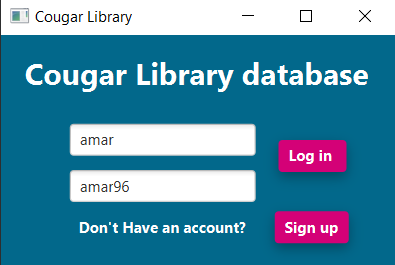
# Methodology

## Software used with versions

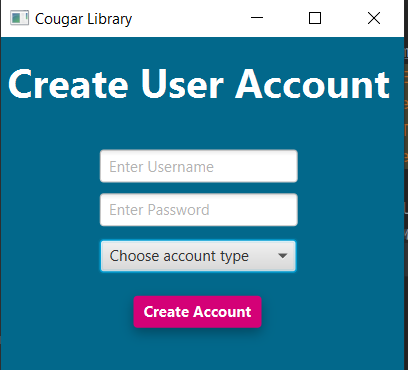
* + MySQL WorkBench 8.0 CE
  + MySQL Server 5.6
  + AWS RDS
  + IntelliJ IDEA 2020.1
  + Java JDK 11.0.6
  + JavaFX 11.0.2
  + Scene Builder (Gluon) 11.0.0
  + GitHub

## UI Design

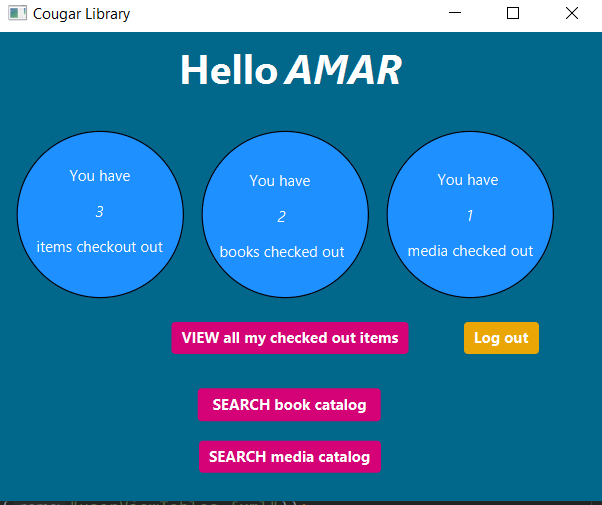
* + Main login page



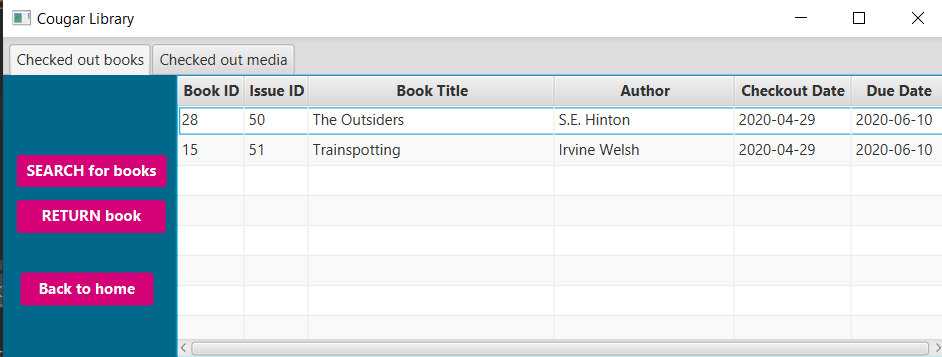
* + User creation form



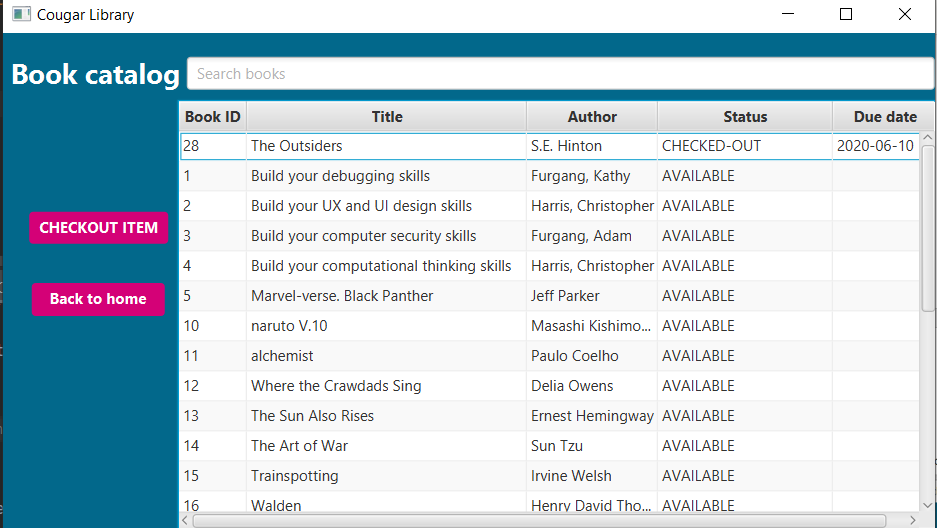
* + User home page



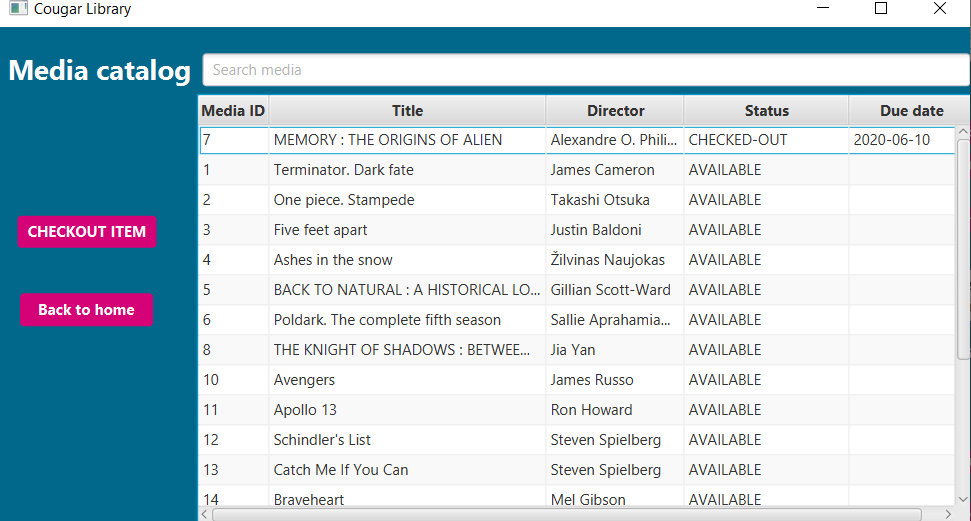
* + Page for users to view their checked out items and make changes



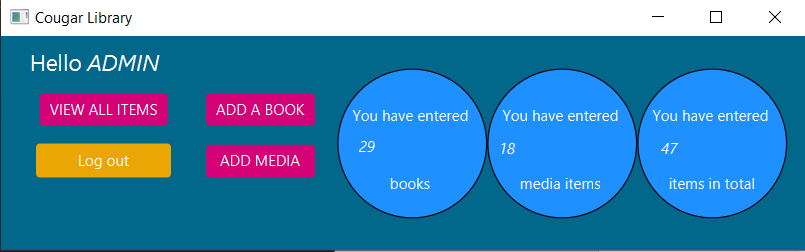
* + Book catalog page to search for books to check out



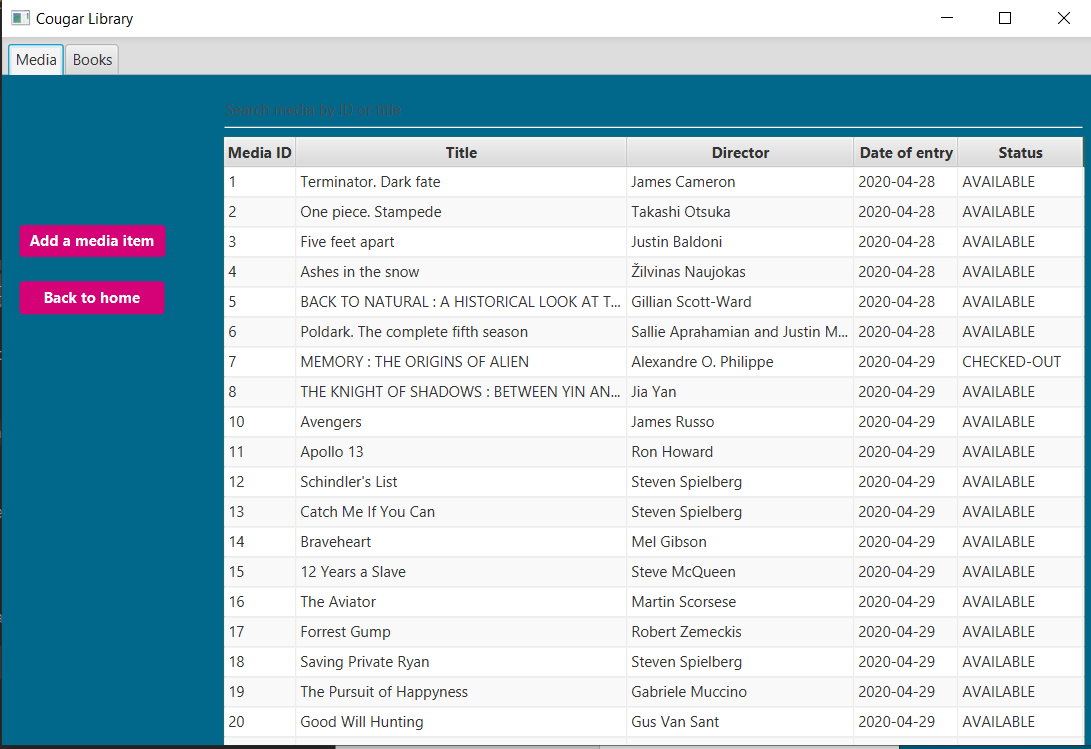
* + Media catalog page that searches library database of all available media items



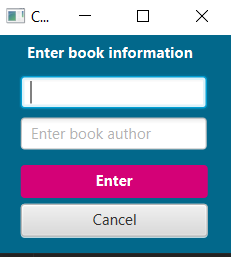
* + Admin home page



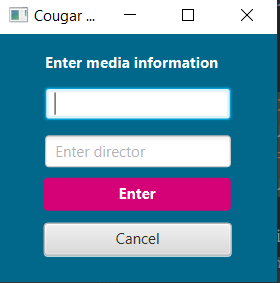
* + Viewing all items in library database on admin side



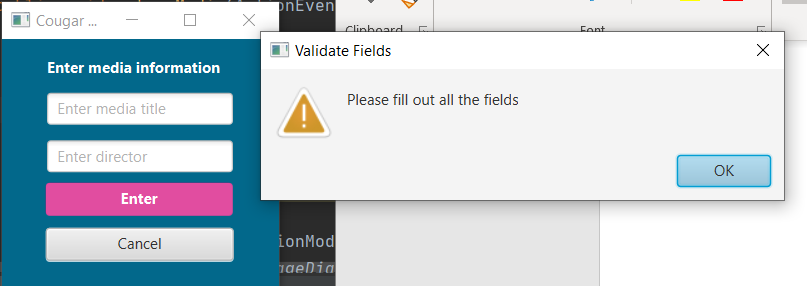
* + Form for admin to enter new book into database

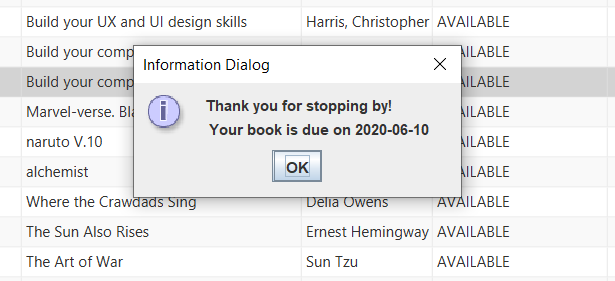


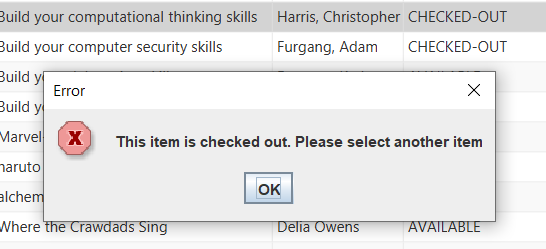
* + Form for admin to enter new media item into database

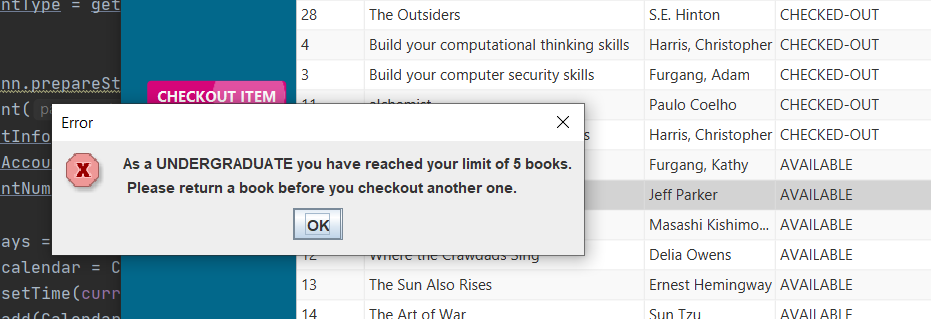


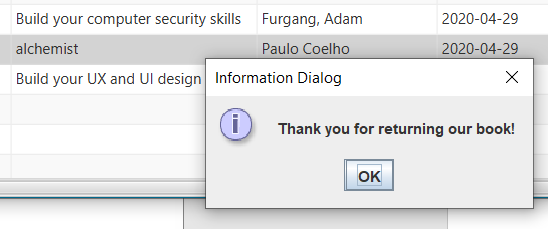
* + Validation texts



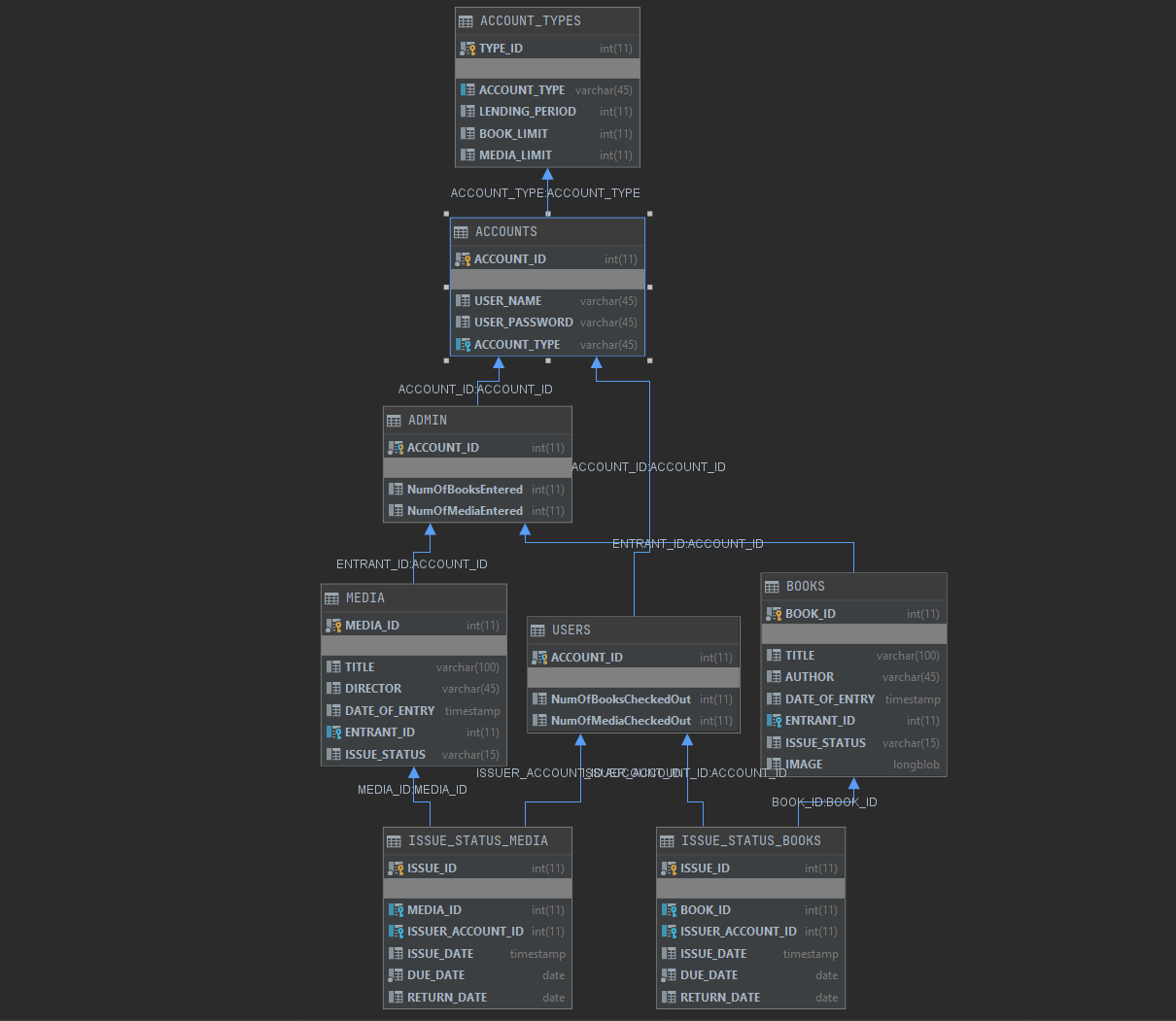








## Programming Logic

* + 
  + The logic I implemented throughout the duration of this project was strictly focused on the relationships between my tables. I also put into perspective how I would implement the core functions of the application such as checking out a book, return a book, searching for books, etc. and how I could design this into each component of my application. From here I decided I would program the logic into each component of the UI and build off of that.
  + For example: If I want to **checkout a book** I had to look at my ERD and map out all the CRUD operations I would have to execute in order to maintain the integrity of all my data. In the case of checking out a book I would have to 🡪
    1. Check to see if the user has reached their limit of book items based on their account type.
       - Solution:
         * write a join query to select from accounts and account types to get the book limit for the account that is logged in.
         * write a second select statement that counts the number of entries in the issue\_status\_books tables that matches the account id of the user that is logged in.
         * compare both results to check if user has exceed limit
         * ^ in my case I used an if statement to check these two values after passing them to two separate int variables. I would only allow an insert if the user has not exceeded the limit. (see code in mediaCatalogController for details)
    2. Update the status of the book that is being checked out to ‘AVAILABLE’
       - Solution:
         * Write an update statement that sets status of the book being checked out to “AVAILABLE”
    3. Insert a new record into issue\_status\_books with the bookID, accountID of the logged in user, and the due date based off the account type of the logged in user
       - Solution:
         * Write an insert statement with a where clause that takes in parameters for all 3 mentioned above
         * I already have the account type collected from step 1 and the lending period can be collected from that same result set.
         * The due date I calculated by getting the current date using Java’s built in Calendar class. I then used the Calendars “.add” function to add the number of days based on the lending period to get the due date
         * The due date is then passed to a java.sql.Date object to be passed into the insert statement parameter so that it can be entered into the database table.
    4. Update the number of books the logged in user has checked after checking out new item
       - Solution:
         * Write an update statement that will add 1 to the “NumOfBooksCheckedOut” for the logged in user.
    5. Keep track of the logged in user through different scenes. This functionality was crucial in keeping track of who the logged in user was through out the different scenes. In order to pass the correct parameters in my sql statements, I had to find a way to keep extract the user’s account ID from the very beginning of logging in. (credit to **Chaldun Alder** for giving the Idea)
       - Solution:
         * Create a function that will pass an int object which can be accessed throughout the different scenes. Sort of like a cookie in web browsing.
         * My function was called “PassData” and you will find this as the first function in almost all of my scenes.
         * As soon as the user logs in, I instantiate an instance of the next scene’s controller to call the “PassData” function and pass the users accountID into the function’s parameter. I then set that accountID in the next scene and from there I can use a select statement to collect that user’s information 😊
  + The logic and thought process for the rest of my application pretty much follows the same pattern as the above steps. I first determine what function I want to implement/what function does the users want, and then determine what CRUD operations I need to execute in my database in order to maintain the integrity of data and its relationships. From there I begin to program the logic into the different components of my application using whatever resources that SceneBuilder and JavaFX has to offer.

# Conclusion

After completing this project, one major lesson I took from it was to have a solid database design before beginning to program your application. Having a deep understanding of the different ways the data would be used was essential for me in designing my application. After creating my relationships and writing various SQL statements, I was able to confidently begin programming my database application. Another import lesson I took was to have a consistent design and drawing out wireframes before-hand that way you’re nor scrambling to figure how you want your application to behave/look while programming it. In conclusion the entire project was extremely fun and if I were to continue designing this application there are some additional features and fixes I would like to implement such as:

* Keeping track of multiple books/media with the same title and author/director in one record
  + I would have to update the quantity each time a user checks out one of these items
* Adding an image attribute for books and media
  + Admin would be allowed to upload an image of the item
  + Users can then see the image item in the catalog
* Adding additional profile attributes to the user account such as profile image and other key attributes
  + Users can upload a profile image and edit their personal account information

P.S

Thank you for this opportunity Professor. It was a pleasure learning under you. You truly cared for your students and gave it your best to ensure that every student understood the concepts you were teaching. Your instructions were always easy to follow, especially when I was in class. Thank you for everything professor. I hope I have the opportunity to learn more from you in the future. Stay safe and God bless.