Huy Nguyen and Rebekah Koon

Hawkins Public Library

CS 340

Summary

Library database for Hawkins Public Library that records the information of library customers, books present at the library, and upcoming events.  
Link to website: https://hawkins-library.now.sh

Table of Contents

[Design Changes 2](#_Toc34658321)

[Upgrades from Step 5 to Step 6 Draft 2](#_Toc34658322)

[Actions Based on Feedback – Step 6 Draft 2](#_Toc34658323)

[Upgrades from Step 4 Draft to Step 5 Draft 2](#_Toc34658324)

[Actions Based on Feedback – Step 5 Draft 3](#_Toc34658325)

[Upgrades from Step 3 Final to Step 4 Draft 3](#_Toc34658326)

[Actions Based on Feedback—Step 4 Draft 4](#_Toc34658327)

[Upgrades from Step 3 Draft to Step 3 Final 4](#_Toc34658328)

[Actions Based on Feedback—Step 3 Final 4](#_Toc34658329)

[Upgrades from Step 3 Draft to Step 3 Final 4](#_Toc34658330)

[Actions Based on Feedback—Step 3 Draft 5](#_Toc34658331)

[Upgrades from Step 2 to Step 3 Draft 5](#_Toc34658332)

[Actions Based on Feedback—Step 2 5](#_Toc34658333)

[Upgrades from Step 1 to Step 2 6](#_Toc34658334)

[Overview 7](#_Toc34658335)

[Outline 8](#_Toc34658336)

[customers 8](#_Toc34658337)

[books 8](#_Toc34658338)

[checkoutOrders 8](#_Toc34658339)

[events 9](#_Toc34658340)

[eventRegistrations 9](#_Toc34658341)

[Feedback from Peer Reviewers—Step 5 10](#_Toc34658342)

[Feedback from Peer Reviewers—Step 4 12](#_Toc34658343)

[Feedback from Peer Reviewers—Step 3 16](#_Toc34658344)

[Feedback from Peer Reviewers—Step 2 18](#_Toc34658345)

[Entity-Relationship Diagram 21](#_Toc34658346)

[Schema 22](#_Toc34658347)

# Executive Summary

In turning in step 2, we received feedback to remove the Employees and Audiobooks entities. We decided to take the advice of our reviewers and remove both of these entities. In order to fulfill our entities requirement, we added an Authors entity and created a new many-to-many relationship between Authors and Books. We also decided to change the description of our library in order to reflect a smaller town.

Our reviews for step 3 suggested displaying the tables from our website and creating a more user-friendly way of displaying the data from our database. We decided to keep the tables on our admin page, but we decided to change how books and events were displayed on our library and events pages. A TA recommended that we change the presentation of our document, so we decided to add a table of contents and cover page as well as place each section of our document on a separate page. In terms of other changes, we removed the Authors entity and added an Events entity and created a many-to-many relationship between Events and Customers. In addition, we decided to rename our “add a book” page on our website to “manage books.” Our added a style.css file to add style to our website. Other changes we made included renaming of foreign keys within our tables and uncapitalizing the names of our tables.

We did not receive any feedback for step 4. However, we decided to make several changes to our website. We initially used bootstrap to style our pages, but we decided to overhaul the website and use React on the frontend and Next.js to handle routing. After doing so, we connected our database to our website to display data on our website from each table in our database. Create functionality was also implemented for books and events. In addition, tables were updated in order to reflect new ideas for our website. In changing our entities, we added an imageUrl attribute to both the books entity and the events entity in order to display images on our website for each book and event. Because we decided to use a customer’s email to checkout and return books, we changed the email attribute within customers to unique. The phone attribute within the customers entity was also changed from an int to a varchar in order to prevent values inputted from going over the maximum value for integers. In terms of the events table, the guest and description attributes are no longer required.

In implementing the feedback we received for step 5, we decided that the lateFee attribute within the customers entity was not necessary with the current design of our website, so this attribute was removed. Feedback also noted that we should add “ON DELETE CASCADE” to the eventRegistrations and checkoutOrders table in order to handle the effect of deleting a customer or event. This will delete event registrations or checkouts tied to any customers if they are deleted. Similarly, any deleted events will delete event registrations tied to those events, and any deleted books will delete their corresponding checkout order. Our DDL file was edited in order to remove any syntax errors that were encountered by our reviewers while also adding “DROP TABLE” statements for each of our tables. In addition, the delete queries in our DML file were changed to reflect deleting rows based on the ID. In upgrading our website, we added separate select queries for our library page and manage books page. We did not need to display images on the manage books page, so the imageUrl is not displayed for this page. In terms of the appearance of our website, we implemented modals for forms and error handling that displays a message to users describing whether or not an action was successful. We also decided the dueDate attribute was not necessary since we removed the lateFee attribute.

# Overview

We will be creating a website that contains a database for Hawkins Public Library. The fictional city of Hawkins has a population of 30,000 people. With 100,000 Books available for checkout, the Hawkins Public Library has a wide variety of genres to choose from. An average of 150 visitors stop by the library each day and approximately 250 Books are checked out daily. Because of the large number of Books within the library, a database will be essential in order to store their information. The Books entity will be used to store the information of each book. The database will also be used to access information related to Customers of the library through the Customers entity. Customers can check out multiple Books in their CheckoutOrders. They can also have several CheckoutOrders, so having a database will be important in tracking each of the CheckoutOrders currently open. Additionally, events will be held at the library, which will be represented with the Events entity. Events include author meet and greets, book clubs, and children’s read-alouds. Customers will be able to sign up for multiple Events, and Events will be able to hold several Customers. In this way, the information stored within the database will allow the Hawkins Public Library to determine which Books have been checked, track the CheckoutOrders of Customers, and keep track of Events being held at the library.

# Outline

customers entity: Records the information of library Customers

* customerId: int, auto\_increment, unique, not NULL, primary key
* firstName: varchar, not NULL
* lastName: varchar, not NULL
* email: varchar, unique, not NULL
* phone: varchar, not NULL
* dateJoined: date, not NULL
* One-to-many relationship between the Customers and CheckoutOrders entities: Customers can have multiple CheckoutOrders, but each of the CheckoutOrders is only associated with at most one of the Customers. Customers can have 0 or more CheckoutOrders. customerID will be implemented as a foreign key within the CheckoutOrders entity.
* Many-to-many relationship between the Events and Customers entities: Customers can sign up for multiple Events and Events can have multiple Customers.

books entity: Records the information of library Books

* bookId: int, auto\_increment, unique, not NULL, primary key
* title: varchar, unique, not NULL
* author: varchar, not NULL
* publisher: varchar, not NULL
* genre: varchar, not NULL
* imgUrl: varchar
* oid: int, foreign key from CheckoutOrders entity; used to keep track of what order the book belongs to. It will be NULL if the book does not belong to an order.
* One-to-many relationship between the Books and CheckoutOrders entities: CheckoutOrders can contain 0 or more Books, but each of the Books can only be in at most one of the CheckoutOrders. orderNumber from CheckoutOrders will be implemented as a foreign key within Books to track which of the CheckoutOrders it belongs to.

checkoutOrders entity: Records the checkout order of a Customer. CheckoutOrders can contain multiple Books and are associated with Customers.

* orderId: int, auto\_increment, unique, not NULL, primary key; used to identify a specific order
* checkoutDate: date, not NULL
* cid: int, foreign key from Customers entity. Represents the ID of the customer who made the checkout order.
* One-to-many relationship between the Customers and CheckoutOrders entities: Customers can have multiple CheckoutOrders, but each of the CheckoutOrders is only associated with at most one of the Customers. Customers can have 0 or more CheckoutOrders. customerID will be implemented as a foreign key within the CheckoutOrders entity.
* One-to-many relationship between the Books and CheckoutOrders entities: CheckoutOrders can contain 0 or more Books, but each of the Books can only be in at most one of the CheckoutOrders. orderNumber from CheckoutOrders will be implemented as a foreign key within Books to track which of the CheckoutOrders it belongs to.

eventsentity: Records upcoming library events

* eventId: int, auto\_increment, unique, not NULL, primary key
* name: varchar, not NULL
* date: date, not NULL
* guest: varchar
* description: varchar
* imgUrl: varchar
* Many-to-many relationship between the Events and Customers entities: Customers can sign up for multiple Events and Events can have multiple Customers.

eventRegistrationsentity: Represents the relationship between Events and Customers

* registrationId: int, auto\_increment, unique, not NULL, primary key
* cid: int, not NULL, foreign key from the Customers entity
* eid: int, not NULL, foreign key from the Events entity

# Entity-Relationship Diagram

A close up of a logo

Description automatically generated

# Schema

A screenshot of a cell phone

Description automatically generated