40.0 hours

Project 10

Build a Library Manager

Why can't I submit this project yet?

You must first complete all prior Techdegree content before you can submit this project. Additionally, you cannot submit more than one project at a time.

Submit

Your current activity is Reporting with SQL.

- Instructions
- What makes a good project?

You've been tasked with creating a library management system for a small library. The librarian has been using a simple sqlite database and has been entering data in manually. The librarian wants a more intuitive way to handle the library's books, patrons and loans.

You'll be given static HTML designs, a set of requirements and the existing SQLite database. You'll be required to implement a dynamic website using Express, Pug, and the SQL ORM Sequelize.

Before you start

To prepare for this project you'll need to make sure you complete and understand these steps.

2 steps

- **Download the project files. We've supplied several files for you to use:**
    - Use the `library.db` as the source of your data
    - HTML mockups and CSS files. These will be the basis of the use cases described in the project instructions.
- Install sequelize using the sequelize CLI. You'll find a link to the documentation and to a Treehouse workshop in the project resources.

Project Instructions

To complete this project, follow the instructions below. If you get stuck, ask a question in the community.

13 steps

- **Models: The `library.db` file should contain 3 tables. Create a Sequelize model for a `books` table, a `patrons` table, and a `loans` table. *There are no `timestamps`.***
    - The `books` table should have the following columns: `id` an integer, `title` a string, `author` a string, `genre` a string and `first_published` an integer.
    - The `patrons` table should have the following columns: `id` an integer, `first_name` (string), `last_name` (string), `address` (string), `email` (string), library_id (string) and zip_code (integer).

    - The `loans` table should have the following columns: `id` (integer), `book_id` (integer), `patron_id` (integer), `loaned_on` (date), `return_by` (date) and `returned_on` (date).

- **Home Screen: As a librarian, I should have a home screen so I can access functionality easily with a single click. See `home.html` for an example. The home screen should include links to *all* of the following pages:**

    - Books:
        - New Book
        - List All
        - List Overdue
        - List Checked Out
    - Patrons:
        - New Patron
        - List All
    - Loans
        - New Loan
        - List All
        - List Overdue
        - List Checked Out

    NOTE: You should use Pug to render your views for this project. Avoid using a front end framework such as Angular.js.

- **Navigation: As a librarian, I should be able to access a main navigation menu from every page of my application. The navigation should include links to the Books Listing page (`all_books.html`), Patrons Listing page (`all_patrons.html`) and Loans Listing page (`all_loans.html`) so I can view this information. See navigation on all pages for examples.**

- **Books Listing Page: As a librarian, I should be able to filter books by 'overdue' and 'checked out' status on the Books Listing Page so I can quickly see the state of the library. Examples: `all_books.html`, `overdue_books.html` and `checked_books.html`.**

- **Add a New Book: As a librarian, I should be able to add a book to the database so that they can be tracked on the system. Example: `new_book.html`.**

    The required fields for user input are:

    - `title`
    - `author`
    - `genre`

    Optional fields:

    - `first_published`

    The form should check that the information is valid. If the form information is valid, the page should redirect to the Books Listing Page, and the new book should appear in the list with updated information.

- **Book Detail Page: As a librarian, I should be able to go to a book's detail page, make edits and view its loan history. Example `book_detail.html`.**

    There should be links to:

    - return checked out or overdue books.
    - each patron in the loan history.

- **Loan Listing Page: As a librarian, I should be able to filter loans by "All", "Overdue", and "Checked Out", so I can quickly see the state of the loan. Examples `all_loans.html`, `overdue_loans.html` and `checked_loans.html`.**

  There should be links to:

  - return checked out or overdue books.
  - each book in the loan history.
  - each patron in the loan history.

- **New Loan Page: As a librarian, I should be able to check out a book so I can lend books to patrons. Example `new_loan.html`.**

  The patron and book fields should be `select` boxes where you can select the `patron_id` or `book_id`.

  The `loaned_on` field should be auto populated with today's date. Example: 2016-10-20. The returned by date should also be pre-populated with a date 7 days in the future, for example: 2016-10-27.

  The required fields for the New Loan field are:

  - `book_id`
  - `patron_id`
  - `loaned_on`
  - `return_by`

  Not required: `returned_on`

- **Return Book Page: As a librarian, I should be able to return a book so we know the current state of a book in our library. Example:`return_book.html`.**

  The only field should be for the `returned_on` should be pre-populated with today's date. Example: 2016-10-20.

  `returned_on` is a required field.

- **Patron Listing Page: As a librarian, I should be able to list all patrons so I can find and access library-goers easily. Example: `all_patrons.html`.**

  There should be links to each patron detail page.

- **Patron Detail Page: As a librarian, I should be able to go to a patron's detail page, make edits and view their loan history. Example `patron_detail.html`.**

  There should be links to:

  - return checked out or overdue books.
  - each book in the loan history.

- **New Patron Page: As a librarian, I should be able to create new library patrons so they can use the facilities. Example: `new_patron.html`.**

  The required fields for user input are:

  - `first_name`
  - `last_name`
  - `address`
  - `email`

    - `library_id`
    - `zip_code`

- **As a librarian, I should be able to be notified if any of the required fields in any given form have any missing data, so that I can correct the information.**

  For example, if the first name field is empty on the new patron form and the librarian submits it, the librarian should see: “First Name is required”.

Extra Credit

To get an "exceeds" rating, you can expand on the project in the following ways:

2 steps

- **Include pagination for the loans and books listing pages.**

- **Include search fields on at least one of the books or patrons listing pages.**

  Examples:

    - `first_name`, `last_name`,`library_id`, etc for patrons
    - `title`, `author`, `genre`, etc for books

  Searching should be case insensitive and be partial matches for strings.

NOTE:

- To get an "Exceeds Expectations" grade for this project, you'll need to complete **each** of the items in this section. See the rubric in the "**How You'll Be Graded**" tab above for details on how you'll be graded.
- If you’re shooting for the "Exceeds Expectations" grade, it is recommended that you mention so in your submission notes.
- Passing grades are final. If you try for the "Exceeds Expectations" grade, but miss an item and receive a “Meets Expectations” grade, you won’t get a second chance. Exceptions can be made for items that have been misgraded in review.

Download files

Zip file

Project Resources

External Link

Using Sequelize with Express

Workshop

Install and Use Sequelize CLI

External Link

Sequelize CLI

External Link

Sequelize Documentation - Adding/removing timestamps from definitions

External Link

Sequelize Documentation - Validations

External Link

Sequelize Table Associations

External Link

Sequelize Documentation - Model associations

External Link

Sequelize Documentation - Pagination and Limiting

Need Help?

Have questions about this project? Start a discussion with the community and Treehouse staff.

Get Help