2019 FBLA Coding and Programming Submission

Student: Divij Karnani

School: Troy High School, Fullerton, CA

Application Name: Troy's eLibrary

This application package has been developed and submitted by Divij Karnani, a student of Troy High School, Fullerton, CA, for FBLA 2019 competition - Coding and Programming event

The application consists of 3 main components -

1. Database : SQLite

2. Server : Node.js, Express & Sequelize

3. Client : Vue.js & Vuetify

Project setup

1. Install Node.js 10.15.1 (Free download from Source: https://nodejs.org/en/download/)

2. Install SQLite Studio (Free download from Source: https://sqlitestudio.pl)

3. Unzip the package into a folder called elibrary or clone from git repository https://github.com/dkarnani/elibrary.git

4. Open a terminal window and run the following commands-

5. cd elibrary

6. npm install (This will install the required dependencies mentioned in package.json)

7. npm start dev - This will start the node server on http://localhost:3000

8. npm run serve - This will launch the vue-cli application on port 8080

Alternatively, the application can be opened in a browser using the URL - https://dkarnani.github.io/elibrary

Purpose

The main purpose of the application is to help manage the issuance of e-books to students. The user interface of the application is structured into 4 main pages –

1. Maintain eBooks

A user can search for existing books to edit or add new books from this page. There is validation for required fields and uniqueness of the book's ISBN number. This page also allows maintenance of book codes. From this page, a user can easily determine how many copies of a book are in stock and checked out, which student has the copies checked out with the checkout and due dates.

2. Maintain Students

A user can search for students to edit or add new students from this page. There is validation for required fields and uniqueness of the student ID number. This page also shows how many books have been issued to a student. There is also a link to navigate to the checkout / return page.

3. Check-out / Return eBooks

This page is used to issue and return books. A user can select the student by name or student ID and check the books that are issued to the student. Books can be easily issued to a student or returned back from the student.

4. Print Report for Issued eBooks

This page shows a table that contains books issued to the students. A PDF version of the report can be exported or printed.

Technical Design

1. Database – All the data used by this application is stored in the SQLite relational database. The data from the client side is dynamically saved in the database with each edit/new/delete action. The tables were designed based on the data objects that need to be stored.

Four tables were created →

a. Books – Contains the books master data records
b. BooksInventories – Contains the book codes associated with each book
c. Students – Contains students master data records
d. Transactions – Contains information on all the books issue and return transactions.

There were 5 additional views created for processing in the application →

a. BookInventoryWithStudent – Shows the associations between book codes and students
b. BooksInStock – Shows the number of available copies of each book
c. BooksWithCount – Shows the total number of copies of each book
d. StudentsWithCount – Shows the number of books that are due for each student
e. TransReport1 – Shows all the books that each student currently has

2. Server side – This was built using Node.js, Express.js, and Sequelize. The server receives the CRUD (Create, Retrieve, Update and Delete) operations from the UI and performs database operations

3. Client side- This was built using vue-js-cli framework and vuetify. The user interacts with the UI controls that perform error handling and make REST API calls to the server.

    a. The client-side webpage includes multiple interface options:
        i. a search bar for students/books
        ii. a form that allows adding new students/books
        iii. a table displaying the students/books
        iv. edit/delete icons allowing to change the data
        v. a drop-down menu for selecting a student to check out books
        vi. an option allowing to export the data of students and books to PDF form

The overall flow of the program goes as follows:

- When the student makes a change to the data on the webpage, it sends a REST API call to the server.
- When the server receives this call, the node.js and express.js files on the server perform the specified function and alter the database as requested by the action.

- This function then sends a response back to the client side, forcing the table on the webpage to refresh and display the updated data.