Library Management System

Assignment_ID: assignment_category_0008

Website Theme

A well-renowned school needs a Library Management system. They need a website that allows them to add books of different categories to their collection, and update book information. Books can be borrowed or returned, and the system keeps track of it. They wrote down their necessary features and provided them to you. Please thoroughly read the instructions below.

Key Rules:

- Include a minimum of 18 notable GitHub commits on the client side
- Include a minimum of 8 notable GitHub commits on the server side
- Add a meaningful readme.md file with the name of your website and a live site URL. Include a minimum of five bullet points to feature your website. Write your selected category's name here.
- Make it responsive for all devices. You need to make it responsive for mobile, tablet and desktop views.
- After reloading the page of a private route, the user should not be redirected to the login page.
- Use the Environment variable to hide the Firebase config keys and Mongodb credentials.
- Don’t use any Lorem ipsum text; you can not use the default alert to show any error or success message.

Main Requirements

1. Make sure your design and website idea is unique. First, finalise your idea (what type of website you want to build). Then google the site design or visit `ThemeForest` to get your website idea. [You can visit this blog to collect free resources for your website]. However, your website `can not be related to your previous assignments or any demo project displayed in the course or our conceptual sessions`

1. The home page will have a navbar, banner/slider, footer and at least 4 types of book categories having the following information: `relevant image` and `category-name`, `relevant button`.

2. The navbar will have a website name with the logo, Home, Add Book, All Books, Borrowed Books, and Login. Your website will have these routes. Except for the Home route, other routes will be private.

3. After successfully performing any CRUD - Create, Read, Update, Delete operation, please, show a relevant toast/sweet alert. (Don't use the browser alert() function. otherwise, marks will be reduced)

4. Add two extra sections to the home page in addition to the sections mentioned above.

5. Create an `Add Book` page where there will be a form for the user to add a book. The form will have:
   - Image
   - Name
   - Quantity of the book [must be numeric value]
   - Author Name
   - Category (for example - Novel, Thriller, History, Drama, Sci-Fi, etc.)
   - Short description
   - Rating [must] (number input, 1-5)
   - Add button
   - Some contents/texts about the book (could be static)

This will be a private/protected route.

6. Book categories are simply the topics of the books. You can choose from [here]. Clicking a category will redirect the user to the page having books based on that category(at least 4 books will be there). Each book will have:

   - Image
   - Name

- Author Name
- Category
- Rating

[render the numerical rating using React Rating package or any relevant package]

- Details button

7. Clicking the Details button will take the user to the book details route. Each route should display detailed information about the book. What you will include in the detailed information is entirely up to you but make sure it is relevant. Make sure to implement the `Borrow` button.

- Clicking the `Borrow` button, a modal will pop up. The modal will have a form requiring a return date and a Submit button. Email and Name fields will be filled by the currently logged-in user's email and displayName. By clicking the Submit button, the quantity of that specific book will be reduced by 1. Also, the book will be added to the `Borrowed Books`. If the quantity reduces to 0, disable the `Borrow` button. The quantity can't be negative.

[Explore MongoDB $inc operator. Here is some clue]

- On the `Borrowed Books` page a user will only see the books that he/she has borrowed(filter borrowed books based on the email of the logged-in user), each book card will have -

    - Image
    - Name
    - Category
    - Borrowed Date
    - Return Date
    - Return Button

- Clicking the "Return button" will increase that specific book quantity by 1, and remove the book card from the "Borrowed Books" page.

[Explore MongoDB $inc operator. Here is some clue]

8. The detailed route will be a private/protected route. Please make sure that if the user is not logged in, the private route redirects to the login page.

9. In the All Books route, all the books will be shown with the required information.
   - Image
   - Name
   - Author Name
   - Category
   - Rating [use React Rating package or any relevant package]
   - Update button

10. Each book will have an Update Button. Clicking on the `Update button` will redirect the user to a form page where the form will have the following fields:

   - Image
   - Name
   - Author Name
   - Category (must use dropdown menu)
   - Rating
   - Submit button

   Clicking the Submit button will update the information. This will be a private/protected route.

11. You Must implement Email and password-based Authentication. This means you will have to implement the Registration and the login page. Users should be able to toggle between Login and Registration view.

> `**Note**:` Do not enforce the `forget or reset password feature` and the `email verification method`, as it will inconvenience the examiner. If you want, you can add email verification/forget the password after receiving the assignment result.

On the Registration page, display errors when:

   The password

   - is less than 6 characters
   - don't have a capital letter
   - don't have a special character

On the Login page, display errors when:

- The password doesn't match
- email doesn't match

You can take the error message from Firebase. You can show the error below the input fields or via alert/toast. If you use alert/toast, do not implement the browser alert.

12. Also, implement at least `one extra login` which could be (Facebook, GitHub, google, etc).

13. Once logged in, the user name, profile picture and the logout button should appear on the navbar. If the user clicks on the logout button, make sure to log him/her out.

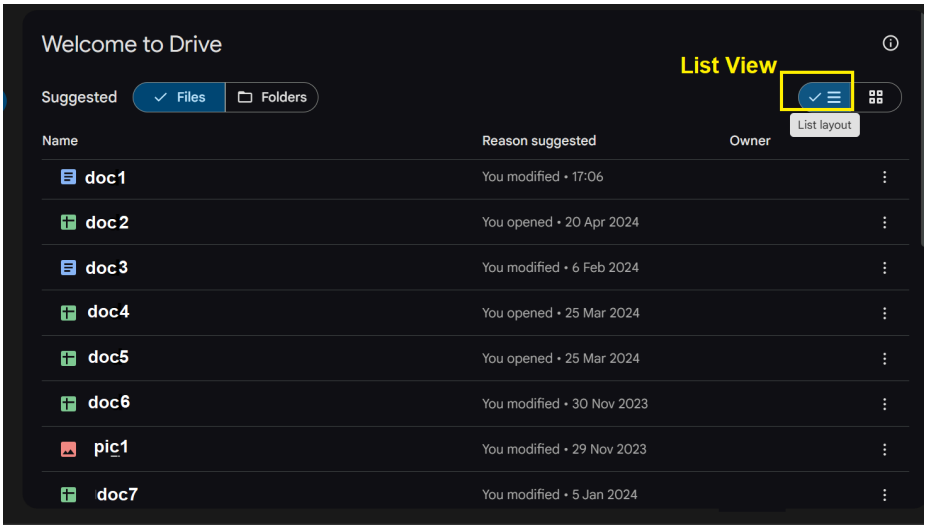
14. Add a 404 page (not found page)

Challenge Requirements

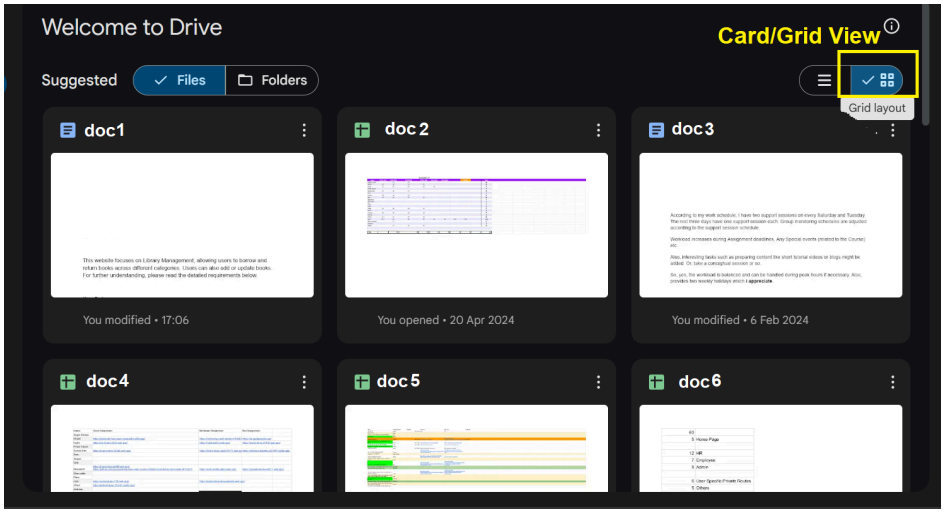
1. Use the JWT token for doing the CRUD operations in the "All Books" and `Add Book` routes.

2. There will also be a Filter button named **“Show available books”** on the **All Books** page. By clicking this button, only available books (Quantity > 0) will be shown.

3. There will be a dropdown menu on the **All Books** page. *Card View and Table View* will be the options in the menu. By selecting this menu, the view will be changed. For example, if someone selects Table view, then the card will disappear and the books will be shown in a Table. If Card View is selected, books will be simply shown as cards. [Include an image for example]



List View



Grid View

4. Don't allow borrowing a book twice for a single user. But after returning a book, the user can borrow that specific book again.

5. Implementing a dark/light theme toggle for the home page. It's optional to implement the theme toggle for the entire website.

Optional Requirements

1. Allow borrowing a maximum of 3 books per person. If someone tries to borrow more than 3 books, show a toast alert.

2. You can add a "librarian" role to a specific email/password registered user. By using that account, the user can navigate the "All Books" route, and perform all the adding or updating operations. But a normal user, not having a "librarian" role, can't do these operations. If you can do this, please, provide the email and password of the account which has the "librarian" role in the README.md file.

3. Use Axios interceptors for handling network requests.
4. Use Swiper JS for the banner and slider.
5. Use the React Hook Form for handling any form.
6. Use ReactToPDF or any relevant package to make the PDF version of the "Read" page of a book.

Additional information

1. You cannot load the data from a .json file. The data must be stored in the database and you must use data from the database.
2. You can use a local or host image anywhere or use pictures from the internet. And it's ok if you have the image URL, but the image link doesn't work.
3. You can use vanilla CSS or any CSS library.
4. Try to host your site on Firebase (Netlify hosting will need some extra configurations)
5. Host your server-side application on Vercel. If needed, you can host somewhere else as well.
6. Make Sure you deploy server-side and client-side on the first day. If you have any issues with hosting or GitHub push, please join the "Github and deploy" related support session.

What to submit

- Your assignment ID/variant
- Your client-side code GitHub repository
- Your server-side code GitHub repository
- Your live website link

Deadline

Read the Text Instructions.

No Pain, No Gain:

- The most beautiful moments in life come after going through hardships and challenges.