**INTERNSHIP DOCUMENTATION**

**BACKEND – LIBRARY MANAGEMENT SYSTEM**

1. Detailed Structure of tables

There are 2 tables in my system:

One is Books and the Other one is User.

The Books table consist of fields:

1. \_id: The unique ID of the book
2. Book\_name: The name of the book.
3. Status: The status of the book i.e., Borrowed or Returned

The User table consist of fields:

1. \_id: The unique ID of the user
2. User\_name: The name of the user
3. Password: The password of the user which is hashed using an algorithm to keep it secure.
4. Books: The books which are with the user. It is referenced to the Books table.
5. Database Diagram

Books

\_id: Type: Object ID

BookName: Type: String

Status: [“Available”,”Borrowed”]

User

UserName: Type: String

\_id: The ID of the user

Password: Type: String

Books: Type: Object Id

1. APIs

BASE\_AUTH: <https://localhost:4000>

Authentication APIS

1. SignUp: POST : “BASE\_AUTH/signup”

Requirements: Account Type, User Name, Password, Confirm Password

Output: Success: True, Message: “User successfully created”

Errors: 1) If any of the above fields are not present, the user can’t signup.

2) Password and Confirm Password not matched.

3) If the user is already registered, he can’t sign up.

ii) Login: POST : “BASE\_AUTH/login”

Requirements: User Name, Password

Output: Success:True, Message: “User successfully logged in.”

Errors: 1) If any of the above fields are not present, the user can’t login.

2) If user is not signed up, he can’t log in.

3) If the details are not filled correctly, he can’t login.

LIBRARIAN APIS

BASE\_AUTH: <https://localhost:4000/librarian>

1. Add Book: POST: “BASE\_AUTH/addBook

Requirements: Book Name,

Output: Success:True, Message: “Book added successfully.”

Error: If the book name is not there, the book will not be added

1. View Books: GET: “BASE\_AUTH/viewBook

Requirements: N/A

Output: The names of all the books registered in system

Error: N/A

1. Update Book: POST: “BASE\_AUTH/updateBook

Requirements: The id of the book to be updated and the new book name of that book.

Output: The updated name of that book

Error: If the id is not valid, it will give an error.

1. Remove Book: DELETE: “BASE\_AUTH/removeBook”

Requirements: The id of the book to be removed

Output: Successfully deleted and the updated book records.

Error: If the id is not valid, it will give an error.

1. View All Members: GET: “BASE\_AUTH/viewAllMembers”

Requirements: N/A

Output: The names of all members

Error: N/A

1. Add Member: POST: “BASE\_AUTH/addMember”

Requirements: User Name and password for that member

Output: The user which is created

Error: The errors will be the same as signing up a user.

1. Update Member: POST: “BASE\_AUTH/updateMember”

Requirements: User Name of the member to be updated.

Output: The updated user

Error: If the user Name is not present, it will not be updated.

1. View Member: POST: “BASE\_AUTH/viewMember”

Requirements: The id of the member to be viewed.

Output: The details of the viewed member

Error: If the id is not there, he will not be able to view the member.

1. Remove Member: DELETE: “BASE\_AUTH/removeMember

Requirements: The id of the member to be removed.

Output: The updated users in the system.

Error: If the id is not there, he will not be able to delete the member.

MEMBER APIS:

BASE\_AUTH: <https://localhost:4000/member>

1. View Books: GET: “BASE\_AUTH/viewBooks”

Requirements: N/A

Output: The available books in the system.

Error: N/A

1. Show Books with User: GET: “BASE\_AUTH/showBooksWithUser”

Requirements: N/A

Output: The list of books with the user

Error: N/A

1. Borrow Book: POST: “BASE\_AUTH/borrowBook”

Requirements: The id of the book to be borrowed.

Output: The details of the book which is borrowed.

Error: If the book id is not valid, the book will not be borrowed.

1. Return Book: POST: “BASE\_AUTH/returnBook”

Requirements: The id of the book to be returned.

Output: The details of the book which is returned.

Error: If the book id is not valid, the book will not be returned.

1. Delete Account: DELETE: “BASE\_AUTH/deleteAccount”

Requirements: N/A

Output: “Successfully deleted”

Error: If the id is not present, the user will not be deleted.

1. Hosting Instructions
2. Signup or login on any backend deploying service such as Render.
3. Connect your account with your Github account.
4. Select New from the Navigation Bar and select the web service option.
5. Select Build and deploy from a Git repository.
6. Choose the repository which you want to deploy.
7. Then, fill in the details which are asked for.
8. Add any environment variables which you may have used in your project by selecting Advanced -> Add environment variables.
9. Click on deploy and your website will be live shortly after it.