AFOUR TECHNOLOGIES
An ACL Digital Company

Presents

Hackathon AFourathon 3.0 – 2023

BOOKOPIA

Library Management System

Documentation

By

Team Dev Rookies

Sugam Phirke
Akanksha Kulkarni
Bhushan Javalgekar
(PCCoE)

Purpose

The purpose of the library management system project is to create a web-based application that can help libraries automate their processes and improve efficiency. The intended users of the system are librarians, library staff, and library patrons.

The library management system project aims to address the challenges associated with manual library management processes. These challenges include:

- Manual book tracking: Libraries currently track books manually, which can be time-consuming and error-prone. The project aims to eliminate manual book tracking by automating the process of borrowing, returning, and managing inventory.
- Inefficient search and retrieval: Libraries often have large catalogs of books, which can make it difficult for patrons to find the books they are looking for. The project aims to improve the search and retrieval process by making it easier for patrons to find books by title, author, subject, or keyword.
- Lack of automation: Many library tasks are still performed manually, such as issuing due date reminders, generating reports, and updating book status. The project aims to automate these tasks to reduce manual effort and improve operational efficiency.
- Limited accessibility: Libraries are often not accessible to patrons who cannot visit the library in person. The project aims to provide easy access to library resources for patrons by enabling online book reservations, renewals, and access to digital resources.

Scope & No Scope

Scope:

- User management: The system will provide user management functionalities for librarians and patrons, such as registration, login, and profile management.
- Book management: The system will allow librarians to manage book records, including adding new books, updating book information, and tracking book availability.
- Borrowing and returning: The system will facilitate the borrowing and returning of books, including issuing due date reminders and managing late returns.
- Search and filtering: The project will provide search functionality to enable users to search for books based on various criteria, such as title, author, book id.
- Reservation and renewal: Patrons will be able to reserve books online and renew their borrowed books if eligible for renewal.

No Scope Dependencies from Third-Party Services or Libraries:

- Authentication and authorization: The project will implement its own user authentication and authorization mechanism.
- Payment gateway: Integration with a payment gateway for fines or fees related to book borrowing will not be included in the system.
- External APIs: The solution will not depend on any external APIs for book data, such as ISBN lookup or book recommendations.

Design

UX Design

The UX design for Bookopia was meticulously crafted to ensure an intuitive and user-friendly experience for librarians using the application. The following design principles and guidelines were followed to achieve this goal.

- Design Aesthetic

Bookopia features a modern and visually engaging design aesthetic. The user interface incorporates clean lines, subtle animations, and a harmonious color palette to create a visually pleasing experience. The typography choice focuses on legibility and readability, allowing users to easily navigate and consume information.

- User Flow

To enhance usability, a user-centric approach was adopted in designing the application's user flows. User flow diagrams were created to map out the journey of librarians as they interact with the system. These diagrams illustrate how users navigate between different screens, perform actions, and access relevant information. [Provide URL for UX flow diagrams].

- Interactive Design

Bookopia leverages interactive design principles to provide a responsive and engaging user experience. The application utilizes intuitive form inputs, clear and actionable buttons, and real-time validation to guide users through their interactions. Feedback mechanisms, such as success messages and error notifications, ensure that users are informed about the outcome of their actions.

- Accessibility and Inclusivity

Bookopia prioritizes accessibility and inclusivity to ensure that the application is usable by all users, including those with disabilities. The design adheres to WCAG 2.1 (Web Content Accessibility Guidelines) standards, incorporating features like appropriate color contrast, keyboard navigation support, and alternative text for images. Usability testing was conducted with individuals with diverse abilities to validate and refine the accessibility features.

- User Research and Testing

The design decisions in Bookopia were informed by user research and usability testing. User feedback and insights were collected throughout the development process to identify pain points and areas for improvement. Iterative testing and design iterations were performed to address usability issues and optimize the user experience.

By following these design principles and guidelines, Bookopia offers a visually appealing, intuitive, and accessible interface that enhances the productivity and satisfaction of librarians.

Links:

❖ Desktop view

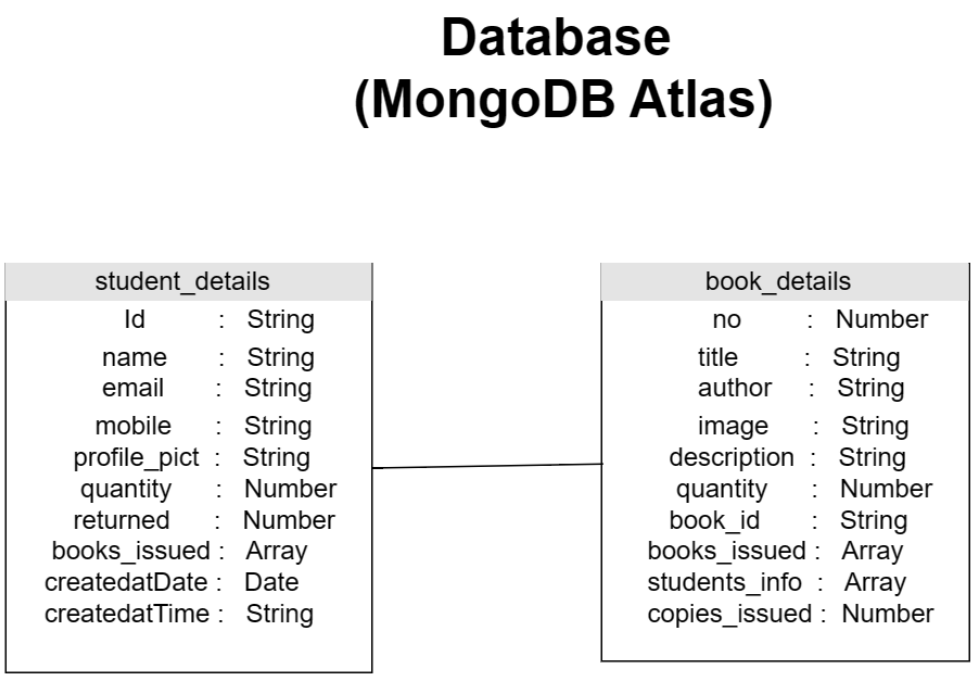
https://sugamphirke.com/Projects/LMS/img/bookopiaHomePage.png
https://sugamphirke.com/Projects/LMS/img/studentPage.png
https://sugamphirke.com/Projects/LMS/img/studentPage-studentModal.png
https://sugamphirke.com/Projects/LMS/img/studentPage-studentModal_addFunctionality.png
https://sugamphirke.com/Projects/LMS/img/studentPage_searchFunctionality.png
https://sugamphirke.com/Projects/LMS/img/noSearchFound.png
https://sugamphirke.com/Projects/LMS/img/studentPage-addStudentModal.png
https://sugamphirke.com/Projects/LMS/img/bookPage.png
https://sugamphirke.com/Projects/LMS/img/bookpage-bookModal.png
https://sugamphirke.com/Projects/LMS/img/bookpage-bookModal_addFunctionality.png
https://sugamphirke.com/Projects/LMS/img/bookpage-bookModal_studentList.png
https://sugamphirke.com/Projects/LMS/img/bookPage-addBookModal.png
https://sugamphirke.com/Projects/LMS/img/bookPage_searchFunctionality.png

❖ Mobile View

https://sugamphirke.com/Projects/LMS/img/mobileView-homePage.jpg
https://sugamphirke.com/Projects/LMS/img/mobileView-studentPage.jpg
https://sugamphirke.com/Projects/LMS/img/mobileView-studentPage_searchFunctionality.jpg
https://sugamphirke.com/Projects/LMS/img/mobileView-studentPage-addStudentModal.jpg
https://sugamphirke.com/Projects/LMS/img/mobileView-bookPage.jpg
https://sugamphirke.com/Projects/LMS/img/mobileView-bookpage-bookModal.jpg

https://sugamphirke.com/Projects/LMS/img/mobileView-bookpage-addFunctionality.jpg
https://sugamphirke.com/Projects/LMS/img/mobileView-bookpage-bookModal_studentList.png.jpg
https://sugamphirke.com/Projects/LMS/img/mobileView-bookPage-addBookModal.png.jpg
https://sugamphirke.com/Projects/LMS/img/mobileView-noSearchFound.jpg
https://sugamphirke.com/Projects/LMS/img/mobileView-studentPage-studentModal.jpg
https://sugamphirke.com/Projects/LMS/img/mobileView-studentPage-studentModal_addFunctionality.jpg
https://sugamphirke.com/Projects/LMS/img/mobileView-bookPage_searchFunctionality.jpg

- Database Design



Database Design Principles/Guidelines/Standards:

- Indexing

We implemented indexing techniques to enhance database performance. Indexes were created on frequently queried columns, such as student ID, book code, and book author. This speeds up data retrieval operations, enabling faster searching and filtering.

Careful consideration was given to the selection of appropriate indexing types, such as B-tree or hash indexes, based on the characteristics of the indexed data and query patterns. This optimization ensures efficient data access for various search and retrieval operations.

- Data Validation and Constraints

To maintain data consistency and integrity, we employed data validation and constraints at the database level. This includes enforcing entity relationships, unique key constraints, and data type validations. By setting up these constraints, we prevent the insertion of invalid or inconsistent data, ensuring the reliability of the system.

- Scalability and Replication

Bookopia is designed with scalability in mind. By leveraging cloud-based database solutions like MongoDB Atlas, we can easily scale our database infrastructure to handle increasing volumes of data and user traffic. Additionally, the use of database replication ensures high availability and fault tolerance, allowing for seamless and uninterrupted operations.

- Security and Access Controls

Data security is a top priority for Bookopia. We have implemented robust security measures, including authentication and authorization mechanisms, to control access to the database. User roles and permissions are defined to restrict unauthorized access and protect sensitive information.

By adhering to these database design principles, guidelines, and standards, we ensure efficient data management, optimal performance, and data integrity within the Bookopia library management system.

API Design

studentData.js:

GET /studentData

- Description: Get data of all the students.
- Parameters: None
- Response: Returns an array of student data.

GET /studentData/Available/:id

- Description: Get data of all the students who have not issued a book of a particular ID.

- Parameters:
    - id: The ID of the book.
- Response: Returns an array of student data.

POST /studentData

- Description: Post new student data to the database.
- Request Body: Student data to be added.
- Response: Returns the newly created student data.

PATCH /studentData

- Description: Edit student data present in the database.
- Request Body: Updated student data.
- Response: Returns the updated student data.

PATCH /studentData/BookInfo

- Description: Edit book data for a particular student.
- Request Body: Updated book data for the student.
- Response: Returns the updated student data.

DELETE /studentData

- Description: Delete a student from the database.
- Parameters: None
- Response: Returns a success message indicating the deletion was successful.

bookData.js:

GET /bookData

- Description: Get data of all the books.
- Parameters: None
- Response: Returns an array of book data.

GET /bookData/Available/:id

- Description: Get data of all the books not issued by a student of a particular ID.
- Parameters:
    - id: The ID of the student.
- Response: Returns an array of book data.

POST /bookData

- Description: Post new book data to the database.
- Request Body: Book data to be added.
- Response: Returns the newly created book data.

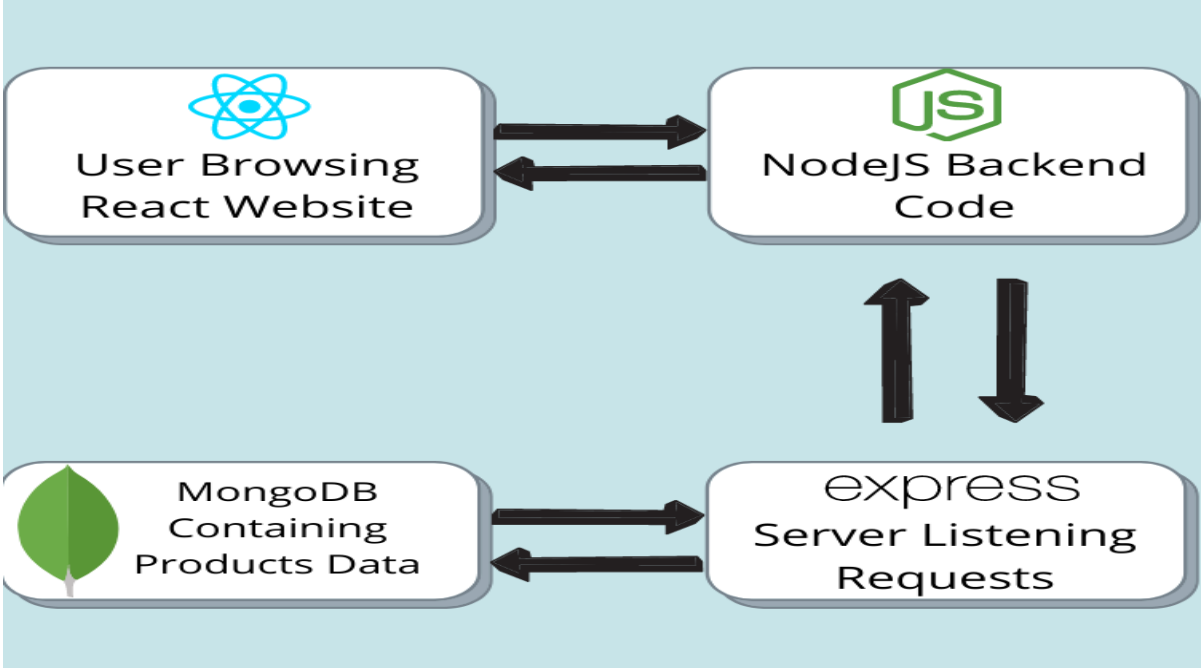
PATCH /bookData

- Description: Edit book data present in the database.
- Request Body: Updated book data.
- Response: Returns the updated book data.

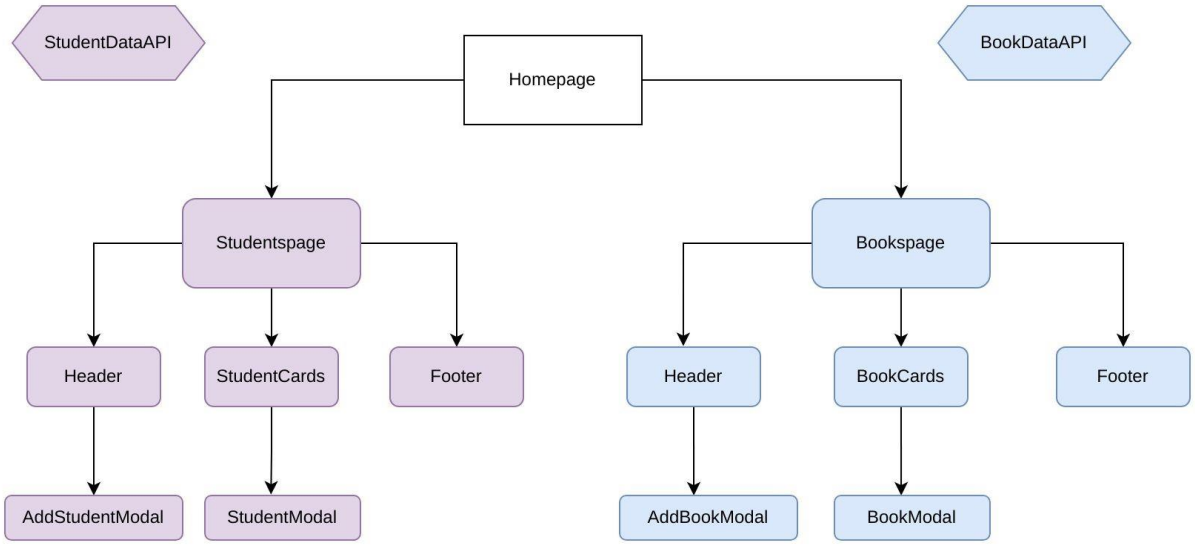
DELETE /bookData/:no

- Description: Delete a book from the database.
- Parameters:
  - no: The unique identifier of the book.
- Response: Returns a success message indicating the deletion was successful.

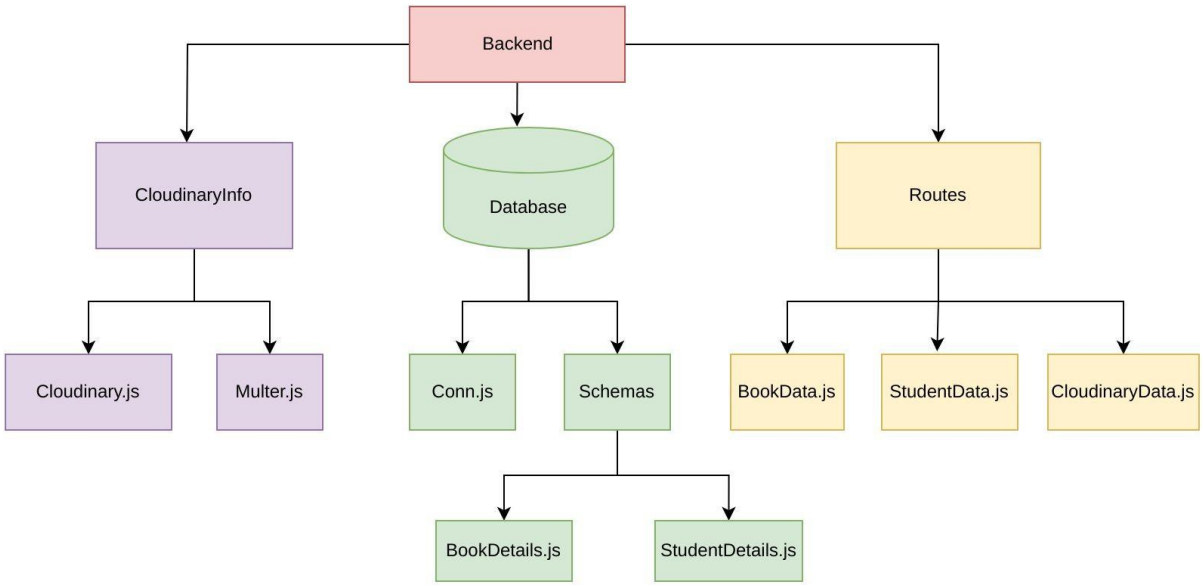
Technology Architecture Design



Frontend



Backend



MEARN Stack: The MEARN stack is a combination of technologies used for building web applications. It consists of MongoDB (a NoSQL database), Express.js (a web application framework for Node.js), React.js (a JavaScript library for building user interfaces), and Node.js (a JavaScript runtime). This stack allows for efficient development of full-stack JavaScript applications, from the server-side to the client-side.

MongoDB Atlas: MongoDB Atlas is a fully managed cloud database service provided by MongoDB. It allows you to host your MongoDB databases in the cloud, providing scalability, reliability, and easy management of your data. With Atlas, you can focus on your application development without worrying about database infrastructure setup and maintenance.

Cloudinary: Cloudinary is a cloud-based media management platform that simplifies the process of storing, optimizing, and delivering images and other media assets in your web applications. It provides features like dynamic image resizing, image transformations, video management, and seamless integration with various development frameworks.

Multer: Multer is a middleware for handling file uploads in Node.js applications. It simplifies the process of accepting files from client requests and storing them on the server. Multer works seamlessly with frameworks like Express.js and provides easy configuration options for managing file size limits, file types, and storage destinations.

Visual Studio Code: Visual Studio Code (VS Code) is a popular source code editor that provides a powerful and customizable development environment. It supports a wide range of programming languages and offers features like code autocompletion, debugging, version control integration, and a rich extension ecosystem. VS Code enhances developer productivity and is widely used for web application development.

Postman: Postman is an API development and testing tool that simplifies the process of designing, documenting, and testing APIs. It allows developers to send requests, analyze responses, and debug APIs easily. Postman provides a user-friendly interface, supports various request methods, and offers collaboration features to streamline the API development workflow.

Vercel: Vercel is a cloud platform that offers a serverless deployment environment for web applications. It simplifies the process of deploying and hosting static websites, single-page applications, and serverless functions. With Vercel, you can easily deploy your applications, benefit from automatic scalability, and leverage built-in features like global CDN and edge caching.

Render: Render is a cloud platform that provides hassle-free deployment and scaling for web applications. It supports various programming languages and frameworks, including Node.js, Python, Ruby, and more. Render simplifies the deployment process with its intuitive interface, automatic SSL, and seamless scaling capabilities, allowing developers to focus on building their applications.

Axios: Axios is a popular JavaScript library used for making HTTP requests from the browser or Node.js. It provides a simple and elegant API for handling asynchronous data fetching and supports features like request and response interception, request cancellation, and automatic JSON parsing. Axios is widely used in front-end and back-end applications for communicating with APIs.

ESLint: ESLint is a linting tool for JavaScript that helps enforce coding style consistency and identify potential errors or bugs. It provides configurable rules and plugins to catch common programming mistakes and enforce best practices. ESLint integrates seamlessly with various editors and build systems, allowing developers to maintain high-quality code and improve code readability.

SCSS/Sass: SCSS (Sassy CSS) is a superset of CSS that offers additional features and enhancements. It provides a more concise and powerful syntax with features like variables, mixins, nesting, and modular imports. SCSS/Sass makes CSS code more maintainable, reusable, and easier to write, ultimately improving the development workflow and organization of stylesheets.

In Bookopia, we follow several codebase good practices to ensure a clean and maintainable project structure. We adhere to the BEM (Block, Element, Modifier) model for writing CSS, promoting modularity, readability, and reducing specificity conflicts. Our project maintains a single global CSS file, ensuring consistency and minimizing style conflicts. We structure our codebase with a proper hierarchy in the folder structure, organizing files based on functionality or features, improving collaboration and scalability. We embrace modularity, creating independent and reusable code components that enhance maintainability and testability. Following the 5-1 Sass architecture, we structure our Sass (SCSS) files into distinct directories, promoting organization and making it easier to manage stylesheets. By incorporating these codebase good practices, we establish a solid foundation for a clean, scalable, and maintainable project structure in Bookopia.

Development

GitHub Link

Bookopia was developed using an Agile process, which allowed for iterative and incremental development, ensuring flexibility and adaptability throughout the project lifecycle. Agile principles such as customer collaboration, continuous delivery, and adaptive planning were key drivers in the development approach. The project team, consisting of three developers from Dev Rookies, embraced Agile methodologies such as Scrum or Kanban to manage the development process effectively. Regular stand-up meetings, sprint planning sessions, and retrospectives were conducted to foster collaboration, track progress, and address any challenges or changes in requirements. This Agile approach promoted frequent communication, quick feedback loops, and

a focus on delivering value to the end-users, resulting in a dynamic and responsive development process.

Adopting an Agile process for the development of Bookopia brought several benefits to the project. Firstly, it allowed for early and continuous customer involvement, enabling the team to gather valuable feedback and make necessary adjustments throughout the development journey. This iterative approach facilitated the timely identification and resolution of issues, resulting in a higher quality end product. Secondly, Agile methodologies facilitated better project visibility and transparency, providing stakeholders with regular progress updates and opportunities for course correction. It also enabled the team to prioritize and deliver features incrementally, ensuring that the most valuable functionalities were delivered early. Lastly, the Agile process encouraged a collaborative and empowered team environment, fostering better communication, creativity, and innovation among the developers. By embracing Agile, the Bookopia project team was able to effectively navigate evolving requirements, improve time-to-market, and deliver a robust and user-centric library management system. Following structure shows the hierarchy for the folders and files in the project:

1) Frontend :

1) api : All apis are global and stored here.
    a) BookDataApi.jsx
    b) StudentDataApi.jsx
2) components :
    a) Header.jsx
    b) Footer.jsx
    c) BookModal.jsx
    d) StudentModal.jsx
    e) AddBookModal.jsx
    f) AddStudentModal.jsx
    g) BookCards.jsx
    h) StudentCards.jsx
3) public:
    a) Book.json
    b) Cat.json
    c) loading.json
4) routes:
    a) HomePage.jsx
    b) BooksPage.jsx
    c) StudentsPage.jsx
5) sass:
    a) pages:
        1) _books-page.scss

2) _home-page.scss
3) _students-page.scss

b) layout:
1) _footer.scss
2) _form.scss
3) _grid.scss
4) _header.scss
5) _modal.scss
6) _modal.scss
7) _pagination.scss
8) _widgets.scss

c) components:
1) _book-cards.scss
2) _book-modal.scss
3) _buttons.scss
4) _notifications.scss
5) _student-cards.scss
6) _student-modal.scss

d) base:
1) _animations.scss
2) _base.scss
3) _typography.scss
4) _utilities.scss

e) abstracts:
1) _functions.scss
2) _mixins.scss
3) _variables.scss

2) Backend :

1) cloudinaryInfo:
   a) multer.js
   b) cloudinary.js
2) database:
   a) conn.js
   b) schemas :
      1) BookDetails.js
      2) StudentDetails.js
3) routes:
   a) bookData.js

b) StudentData.js
c) cloudinaryData.js

Branching strategy:

Branching is based on Frontend and Backend. Frontend is branched with respect to the view (components, apis, public, routes, sass styling). Backend is branched based on routes(all apis), database(connection and schemas of database) and cloudinaryInfo which is used for image upload on cloudinary.

Production Deployment

Infrastructure

Steps for setting the project on developers machine, locally:

1. Fork the repo and clone it.
2. Switch to development branch for running in development mode.
3. Make sure you have node Node.js installed in your system.
4. Create a file named ".env" in both the folders.
5. Add the following in Backend ".env" file:

```
MONGO_URL = "<Paste Your Localhost MongoDB Atlas URL here> "
PORT = 3001
CLOUD_NAME = "<Enter your cloudinary cloud name>"
API_KEY = "<Enter your cloudinary api key>"
API_SECRET = "<Enter your api_secret key>"
```

6. Add the following in Frontend ".env" file:

```
VITE_BASE_URL = "http://localhost:3001/"
```

7. Run (from the root) cd Frontend && npm install.
8. Run (from the root) cd Backend && npm install.
9. Open two terminal windows (one for running Server and other for the UI).
10. On Backend terminal, run npm run dev to start the server. It will run on port 3001.
11. On Frontend terminal, Run npm run dev to start the UI Frontend. It will run on port 5173.
12. Go to http://localhost:5173 to see the application running.

Application

Frontend: Vercel, PORT = 5173

Backend: Render, PORT = 3001

```
MONGO_URL =
"mongodb+srv://system:pccoe@cluster.qk2mvko.mongodb.net/CollegeLibraryManagement?retryWrites=true&w=majority"
PORT = 3001
CLOUD_NAME = "dkqrs8ic2"
API_KEY = "999676294331738"
API_SECRET = "67Ba2tQlzO3DIc7oBuMSR9da1Ow"
```

Sensitive Data

Scalability Configurations

- Monitoring and Auto-Scaling: Set up of monitoring tools to track application performance, resource utilization, and user traffic. Implement auto-scaling capabilities to automatically adjust the number of application instances based on predefined thresholds. This ensures that the application can scale up or down dynamically based on demand.
- Database Scaling: Evaluating the scalability of the database infrastructure.
- Optimization and Performance Tuning: Continuously monitor and optimize the application's performance by identifying bottlenecks, optimizing queries, and improving code efficiency. Regularly test the application's scalability under various load conditions to identify any limitations or areas for improvement.
- The codebase is developed such that the small parts of the hierarchical structure can be broken down into modular parts, ensuring the smooth scaling of the project further.

Security Configurations

- Set up SSL/TLS certificates to enable encrypted communication between the server and clients, securing data transmission. View Security Certificate
- Implemented proper input validation and output encoding techniques to mitigate common security vulnerabilities, such as SQL injection and cross-site scripting (XSS).
- Regularly updated and patched the application and underlying software dependencies to address security vulnerabilities.
- Implemented secure coding practices, such as input sanitization and secure session management, to prevent security breaches.

Credentials

BU/IT Point of Contact: projects@sugamphirke.com

Marketing Support

Contact Team: sugam.phirke20@pccoepune.org, akanksha.kulkarni20@pccoepune.org, bhushan.javalgekar20@pccoepune.org

---

AFour TECHNOLOGIES
An ACL Digital Company

AFour Technologies Pvt. Ltd. Office 501, 5th Floor, Sterling Tower,
Pan Card Club Road, Baner, Pune 411045, India