Swami Keshvanand Institute of Technology,

Management & Gramothan, Jaipur

PROJECT KIT

Title of the Project

BibloTech

Abstract

The E-Library System is an advanced, user-friendly platform designed to enhance the accessibility and management of instructional materials and job-related resources. It enables users, including students, educators, and administrators, to interact seamlessly through features like material uploads, downloads, a Q&A forum, and job application management. The system is equipped with a responsive interface, robust backend services, cloud-based storage, and secure authentication, ensuring reliability, scalability, and efficiency. This project leverages technologies like ReactJS, Springboot and cloud to deliver a comprehensive and innovative solution.

Objectives

The E-Library System aims to provide a digital solution for managing educational resources, job opportunities, and collaborative discussions. The key objectives include:

- Simplifying access to instructional materials for students and educators.
- Providing a centralized platform for job listings and applications.
- Facilitating a Q&A forum for collaborative knowledge sharing.
- Ensuring secure data management and seamless user interactions.

Generic Keywords -

- E-Library
- Educational Platform
- Digital Repository

Specific Technology -

- ReactJS
- SpringBoot
- SQl/NoSQL
- Cloud - AWS S3
- JWT Authentication
- REST APIs

Key Features -

Feature	Description
Responsive User Interface	Designed using ReactJS for an engaging user experience.
Secure Authentication	Implemented JWT tokens for robust security.
Cloud Integration	AWS S3 for secure file storage and management.
Interactive Q&A Forum	Enables knowledge sharing and resolution of queries.
Admin Dashboard	Tools for managing content, users, and resources efficiently.

Functional Components -

Component	Functionality
User Authentication	Register, log in, and manage profiles securely.
Book Management	Issue and return books with subscription plans.
Digital Resource Access	24/7 access to e-books and question banks.
Subscription Management	Handle subscription plans and payments.
Admin Dashboard	Monitor and manage user activities and data.

Functionality -

The E-Library System offers the following functionalities:

- User registration, login, and authentication.
- Material uploads and categorized searches.
- Job listings with application submission and tracking.
- Interactive Q&A forum for collaborative discussions.
- Admin dashboard for management tasks.
- Notifications for important events and updates.

Functional Requirements -

1. Hardware Requirements

1. Client System: Minimum 4GB RAM, Dual-core processor, 500GB storage.
2. Server System: Minimum 8GB RAM, Quad-core processor, 1TB storage.
3. Internet Connectivity: Stable connection for accessing cloud services.

2. Software Requirements

1. Operating System: Windows 10 or later, Ubuntu 20.04 or later.
2. Development Tools: Visual Studio Code, Postman.
3. Backend: SpringBoot
4. Frontend: ReactJS.
5. Database: SQL, No SQL
6. Cloud Services: AWS S3, AWS EC2.

3. Manpower Requirements

1. Frontend Developer: To design and develop the user interface using ReactJS.
2. Backend Developer: To implement the server-side logic using SpringBoot.
3. Database Administrator: To manage and optimize database collections.
4. Cloud Specialist: To configure and maintain AWS services.

Non-Functional Requirements-

1. Performance Requirements

- The system should handle up to 100 simultaneous users without performance degradation.
- Response time for any action should not exceed 2 seconds under normal load.

- The system should process file uploads/downloads within 5 seconds for files up to 100 MB.

2. Scalability

- The system should support scaling to accommodate increased user demand (e.g., 500 users in peak times).
- The database should be designed to handle growth in data size by 10% annually.

3. Reliability and Availability

- The system should maintain an uptime of 99.9% during operational hours.
- Backup systems should restore data within 30 minutes in case of system failure.

4. Usability

- The interface should be intuitive and require no more than 1 hour of training for a new user.
- Provide multilingual support if users are from diverse language backgrounds.

5. Security

- All user data should be encrypted using AES-256 encryption.
- Implement role-based access control (RBAC) to restrict user actions based on roles (e.g., admin, librarian, user).
- The system should automatically log out inactive users after 10 minutes of inactivity.

6. Maintainability

- Codebase should follow standard coding practices and include clear documentation for future developers.
- Software updates and patches should be deployed with zero downtime.

7. Portability

- The system should be accessible on various platforms, including desktops, tablets, and mobile devices.
- Ensure compatibility with major web browsers like Chrome, Firefox, Edge, and Safari.

8. Compliance

- Adhere to relevant standards, such as GDPR for data protection if dealing with user data in the EU.
- Follow accessibility standards like WCAG 2.1 to make the system usable for people with disabilities.

9. Efficiency

- Optimize resource utilization to ensure the system runs efficiently on hardware with minimal specifications.
- Avoid memory leaks and unnecessary background processes.

10. Support and Serviceability

- Provide 24/7 technical support for critical issues.
- Include detailed troubleshooting guides and FAQs within the system.

Expected Outcomes

1. Increased accessibility to educational resources for students and teachers.
2. Enhanced job application process with structured listings and notifications.
3. Efficient management of data with secure storage and retrieval mechanisms.
4. Improved collaboration and engagement among users through interactive features.

Guidelines

To ensure the success of the E-Library System, the following guidelines should be adhered to:

1. Ensure modular development to simplify debugging and scalability.
2. Follow RESTful API design principles for seamless backend communication.
3. Utilize cloud services like AWS S3 for secure and efficient file storage.
4. Adhere to user authentication standards using JWT tokens.
5. Regularly test all features to ensure functionality and robustness.

6. Secure sensitive data using encryption and authentication techniques.
7. Optimize database queries to maintain performance efficiency.

References:

[1]ReactJS Official Documentation: https://react.dev
[2]JWT Authentication Guide: https://jwt.io/introduction
[3]SpringDB Official Documentation: https:/springdb.com
[4] AWS Documentation: https://aws.amazon.com/documentation