Presents
Hackathon AFourathon 3.0 - 2023

Prepared by:
Pranjal Jain
Rajdeep Singh
Utkarsh Jain

AFour Digital Asset Design, Engineering and Support Guide

Solution

Solution Purpose

The website for the library tries to overcome problems in managing and using library resources. Students, teachers, and library personnel are its intended end users or consumers. The website seeks to improve user experience for all stakeholders by streamlining the borrowing process, offering simple access to catalogue information, and enhancing overall library management.

Scope and No Scope

Users can register on the library website, search and browse books, borrow and return books, update personal information, and receive notifications, among other key use cases. It is not anticipated that the website would have any scope dependencies on third-party services or libraries because it was developed using just internal development tools and resources.

Design

UX Design
UX Diagrams

URL:- https://www.figma.com/file/JKuLfsW4MEk8NNTsxE3XCt/Lib?type=design&node-id=0%3A1&mode=design&t=VbDf44MMQkhC0bzA-1

UX Design Principles/Guidelines/Standards

Simplicity and minimalism in design.
Consistency in layout and visual elements.
Clear and intuitive navigation.
Responsive design for seamless user experience on different devices.
Legible fonts and well-structured content.
Visual hierarchy to highlight important elements.
Interactive elements and immediate feedback.
User-centered design approach.
Accessibility considerations for all users.
Performance optimization for fast loading times.

Database Design

Database Schema: The database schema includes tables for students, books, and their relationships.

- Student Table: ID(Primary Key), Name, Email(Unique), Phone(Unique), Books
- Book Table: Code(primary Key), Title, Author, Description, Alloted , StCode

Database Design Principles/Guidelines/Standards

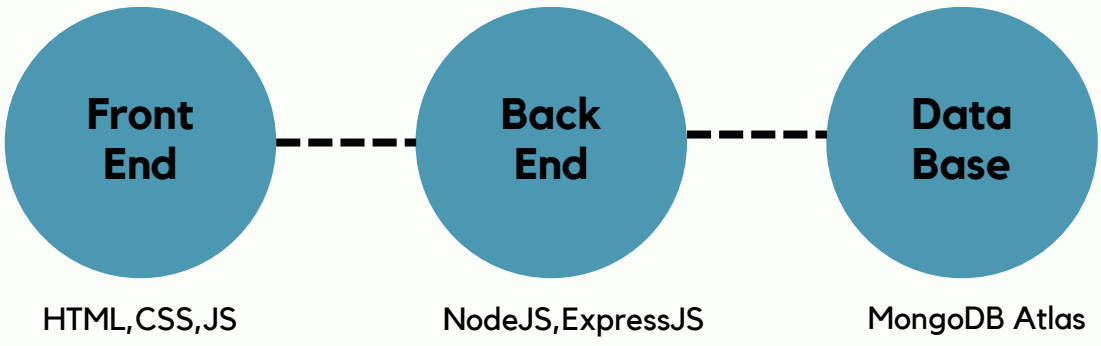
Denormalization: For increased effectiveness and fewer join operations, certain data is denormalized.
Indexing: In order to improve query performance, indexes are implemented on important fields.
Data Validation: To guarantee data consistency and integrity, appropriate validation criteria are performed.

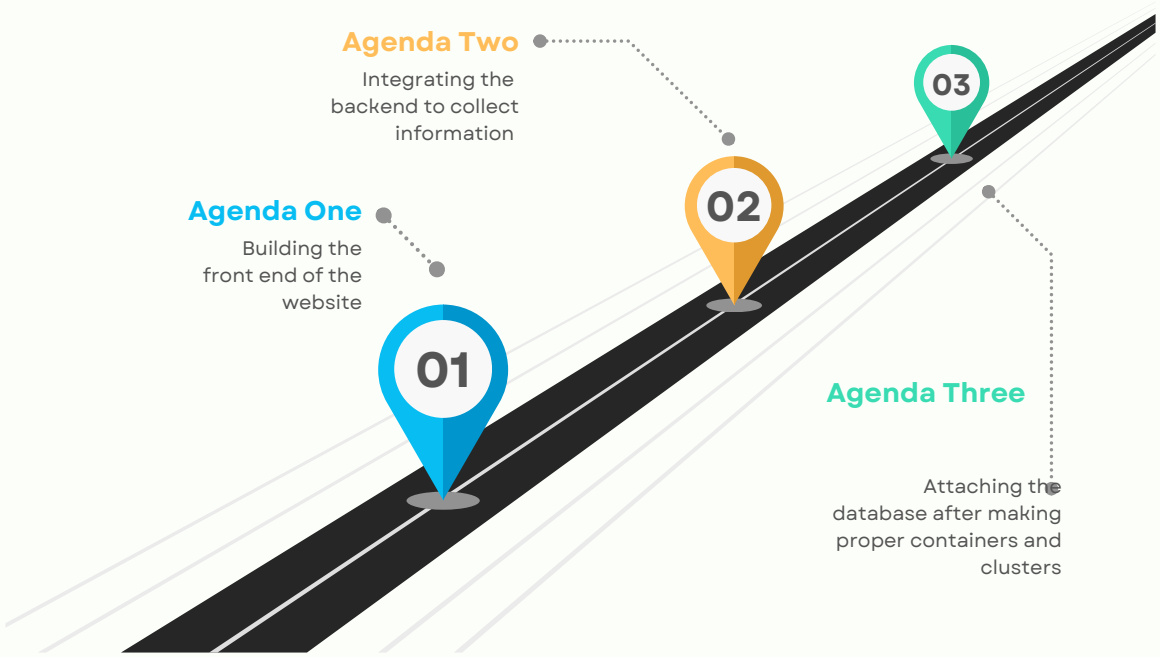
These procedures aid in quicker data retrieval and improved database performance.

API Design

The project uses APIs for a number of services, including user registration, book management, student administration, and authentication.
API Design and Development Principles: The project adheres to industry-recognized standards
Supporting Documents: The project offers API Swagger documentation that lists the request/response formats, parameters, and API endpoints.
Overall, the project exhibits an API architecture that has been carefully planned and built, ensuring scalability, maintainability, and adherence to coding standards.

Technology Architecture Design





Deployment architecture: The project's containerized design uses Docker containers for specific parts like the database and backend server.
For scalability and high availability, the project uses clustering techniques, which enable many instances of the backend server to process incoming requests.
Load balancing: To ensure optimal resource utilisation and increased performance, incoming traffic is distributed among the clustered instances using a load balancer.

Development

Source Code/Git

Front End Code Repository: This repository has the code for the project's front end, including the HTML, CSS, and JavaScript files in charge of the client-side functionality and user interface.

Back End Code Repository: This repository stores the code for the project's back end, which includes server-side code, database connections, and API implementation utilising Node.js, Express, and MongoDB, among other tools.The project is organised generally into various repositories, each of which serves a particular function in the growth and organisation of the project's codebase.

Branching strategy

The project follows a Git-based branching strategy, specifically the feature branch workflow. It promotes collaboration and isolation of code changes by creating feature branches for each new development task. This allows for easier code reviews, version control, and seamless integration of features into the main branch.

Unit Testing Method

The project's unit testing methodology takes a collaborative approach. The unit test cases are created by the peer developer or the team lead, and the developer turns them into unit test code. When all unit test cases pass, the developer develops the application code to guarantee the project's codebase is resilient and of high quality.

Testing

Test Scope

Functional testing, integration testing, performance testing, and security testing are all part of the project's testing scope. The testing plan includes validating API integration, system performance under load, user registration, book and student management operations, and data security. In-depth test plans will be created to cover these topics and guarantee the system's quality.

BVT Test Cases

User Registration: Check whether a user can properly register with accurate information.
Book Creation: A book's ability to be created and saved in the database should be confirmed.
Student Creation: Check whether a student can be generated and saved in the database by doing this step.
Book Allotment: Check to see if a student can receive a book.
These BVT test cases assure the system's initial stability and usefulness and cover the fundamental functions of the system.

Test Automation Objective & Scope

Test automation aims to increase test efficiency, increase test coverage, and minimise manual work. Functional testing, regression testing, and API testing are all included in the scope of test automation. The project's goal is to maximise testing efforts while ensuring dependable and consistent software quality by utilising test automation.

Test Strategy / Test Plan

Test Strategy: To assure the functioning and dependability of the system, the project uses a thorough test strategy that comprises unit testing, integration testing, and end-to-end testing.
Test Plan: For each testing phase, the project includes a thorough test plan document that lists the test scenarios, test cases, test data, and anticipated outcomes.
Overall, the project shows a clear testing strategy that ensures complete test coverage and adherence to quality standards.

DevOps

Dev Integration Environment Set up

CI/CD Setup: Using technologies like Jenkins or GitLab CI/CD, the project uses a CI/CD (Continuous Integration/Continuous Deployment) methodology. Build pipelines must be set up, automated testing must be integrated, and code must be deployed to the Dev integration environment.
These actions guarantee that the project's development environment will experience smooth deployment procedures.

Production Deployment

Infrastructure

1. Set up a cloud hosting provider like Render.com.
2. Create a virtual machine or container instance.
3. Install Node.js and MongoDB on the instance.
4. Clone the project repository and install dependencies.
5. Configure environment variables for database connection and other settings.
6. Start the server and ensure it's accessible through the assigned IP or domain.
7. Set up a reverse proxy or load balancer for scalability.
8. Configure domain settings and SSL/TLS certificates for secure communication.
9. Monitor the infrastructure for performance and security. These instructions ensure a robust and scalable infrastructure for our project deployment.

Application

1. Set up a hosting environment with the necessary dependencies.
2. Clone the project repository from the designated source.
3. Install project dependencies using package manager.
4. Configure environment variables for database connection and other settings.
5. Build and compile the project.
6. Start the server and ensure it is running on the designated port.
7. Access the application through the specified URL. These instructions provide a clear path for deploying and configuring the application, ensuring a seamless setup process.

Scalability Configurations

1. Horizontal scaling: By deploying numerous application instances and load balancing the requests, the project is built to accommodate rising traffic.
2. Cloud Infrastructure: The project is set up on Render.com, a cloud platform that makes it simple to add or remove resources as needed.
3. Scalability of the database: The project makes use of a scalable database solution, such as MongoDB Atlas, which enables simple scaling by modifying the database resources.

With these configuration decisions, the project is guaranteed to be able to effectively handle increasing user needs while maintaining top performance.

Emergency Maintenance

Emergency Maintenance Details:-

Pranjal Jain:-7792069000
Email:-21ucs155@lnmiit.ac.in

Rajdeep Singh:-8949088349
Email:-21ucs165@lnmiit.ac.in

Utkarsh Jain:-7425094343
Email:-21uec137@lnmiit.ac.in

Github-Link

https://github.com/Pranjal-2310/Library_Website

Email us for more details of the project!

library.modern123@gmail.com

Team Afrofuturism Signing off

AFOUR TECHNOLOGIES
An ACL Digital Company

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