High-Level Design for a DSA Practice and Contest Website

This document outlines the high-level design of a web application designed for users to practice Data Structures and Algorithms (DSA) and participate in contests. The application will be built using the MERN stack (MongoDB, Express.js, React, and Node.js) and deployed in Docker containers. This document will cover the core architectural components, user interface and experience, data storage and management, question and contest management, deployment and containerization, and security and scalability aspects.

Website Architecture

The website architecture is based on a client-server model, with a React front-end and a Node.js backend. The front-end provides the user interface for browsing questions , adding problems, updating and deleting problems and participating in contests, and managing user profiles. The backend handles data storage, retrieval, authentication, contest management, and API interactions. MongoDB is used as the primary database for storing user data, questions, contests, and other relevant information.

The architecture leverages the advantages of the MERN stack, including scalability, flexibility, and ease of development. The use of RESTful APIs allows for efficient communication between the front-end and backend. A key aspect of the architecture is the implementation of a robust authentication system to secure user accounts and sensitive data.

React

Node.js

MongoDB

Express.js

User Authentication Flow

1

Registration

Users can create a new account by providing their basic information such as name, email, and password.

2

Login

Registered users can log in to the website using their email and password credentials.

3

Password Reset

Users who have forgotten their password can request a password reset link to securely update their login information.

4

Multi-Factor Authentication During Forgot Password

The website can optionally require users to provide an additional verification code sent to their mobile device to enhance security.

User Interface and User Experience

The user interface is designed to be user-friendly and intuitive, prioritizing ease of navigation and accessibility for all users. The design will adhere to best practices for user experience, emphasizing clarity, consistency, and a clean aesthetic. The site will feature a responsive design to ensure optimal viewing on various devices.

Key features of the user interface will include:

- A comprehensive question bank with clear explanations and code samples
- Interactive coding editors for practicing and submitting solutions
- Personalized dashboards for tracking progress and performance

Data Storage and Management

MongoDB, a NoSQL database, will be used to store all application data. This includes:

- User profiles, including usernames, passwords, and other personal information
- DSA questions, including problem statements, input and output formats, constraints, and sample test cases
- User solutions, including submitted code and execution results
- Contest data, such as contest schedules, leaderboards, and participant submissions

MongoDB's flexible schema, high scalability, and efficient querying capabilities make it a suitable choice for managing the dynamic data of this application. To ensure data integrity and security, proper indexing and data validation procedures will be implemented.

Problem Management

The question management system will allow for the creation, editing, and deletion of DSA questions. Each question will be categorized based on difficulty level and topic, making it easier for users to find relevant practice problems. The system will also include features for:

- Generating test cases for validating user solutions
- Storing and retrieving user submissions
- Evaluating user solutions based on correctness and efficiency

The contest management system will enable the creation and scheduling of timed contests. Users can register for contests, view leaderboards, and submit solutions. The system will automatically evaluate submissions and track contest rankings. The contest module will be designed to handle a large number of concurrent users and ensure smooth contest execution.

Deployment and Containerization

The application will be deployed using Docker containers, which provide a lightweight and portable way to package and run the application and its dependencies. Docker will enable consistent deployment across different environments and facilitate scaling as user traffic increases. The Docker setup will include:

- Dockerfile for defining the container image
- Docker Compose for orchestrating multiple containers (frontend, backend, database)
- Deployment scripts for automating the deployment process

Docker containers offer several advantages, including faster deployment, reduced resource consumption, and improved portability. This approach allows for a scalable and reliable deployment environment that can handle a growing user base.

1

Dockerfile

Define container image

2

Docker Compose

Orchestrate multiple containers

3

Deployment Scripts

Automate deployment process

Security and Scalability

The website will implement robust security measures to protect user data and prevent unauthorized access. This includes:

- Strong password encryption
- Secure authentication and authorization
- Regular security audits and vulnerability testing
- Implementation of security best practices and industry standards

To ensure scalability, the application architecture will be designed with horizontal scaling in mind. The backend and database can be scaled by adding more server instances, while the front-end can be scaled using a content delivery network (CDN). Load balancing will be implemented to distribute traffic evenly across multiple servers. The system will also incorporate monitoring and logging tools to track performance and identify potential bottlenecks.

Integrating AWS with Docker on an EC2 Instance

1

Launch an EC2 Instance

Start by provisioning an Amazon EC2 instance, which will serve as the hosting environment for your compiler application.

2

Install Docker

On the EC2 instance, install Docker to containerize your compiler application and its dependencies.

3

Configure AWS Integration

Set up the necessary AWS credentials and permissions to allow your Docker containers to access and leverage AWS services.

4

Build and Deploy Docker Image

Create a Docker image for your compiler application and deploy it to the EC2 instance, leveraging AWS services as needed.

Advantages and Disadvantages of MERN Stack

1

Flexibility

The MERN stack provides a flexible, component-based architecture that allows developers to easily swap out technologies as needed.

2

Full-Stack JavaScript

With JavaScript powering both the front-end and back-end, the MERN stack offers a consistent, streamlined development experience.

3

Large Community

The MERN stack has a large and active community, providing abundant resources, libraries, and tools to support development.

1

Steep Learning Curve

Mastering the MERN stack requires learning multiple technologies, which can be challenging for developers new to the ecosystem.

2

Performance Concerns

The reliance on JavaScript for both front-end and back-end can lead to performance issues, especially for computationally intensive applications.

3

Vendor Lock-in

Developers may become heavily invested in the MERN stack, making it difficult to migrate to alternative technologies in the future.

Conclusion and Next Steps

This high-level design provides a comprehensive overview of the architecture, features, and technology choices for the DSA practice and contest website. The proposed design leverages the strengths of the MERN stack, Docker containers, and MongoDB to create a secure, scalable, and user-friendly application. The next steps include:

- Detailed design of the user interface and user experience
- Implementation of the backend API and data storage
- Development and testing of the question and contest management features
- Deployment and containerization using Docker
- Ongoing security audits and performance optimization