

## Knowledge Sharing Forum -

### Objective

Develop a Knowledge Sharing Forum application where users can ask questions, provide answers, and engage in discussions through nested replies. The application should also allow tagging and categorizing questions to improve discoverability.

### Requirements

#### Frontend

1. Library/Framework: React or Nextjs
2. Styling: Tailwind CSS or Material-UI for responsive and interactive design.
3. Functionalities:
  - Homepage:
    - Display a list of questions with their titles, tags, number of answers, and timestamps.
    - Include a search bar to filter questions by title or tags.
    - Pagination to manage the list of questions.
  - Question Detail Page:
    - Display the full question with details such as the author, timestamp, tags, and description.
    - Show all answers, each with nested replies.
    - Provide a form for adding new answers or replying to existing ones.
  - Post Question Page:
    - A form for users to post a new question with fields for title, description, and tags/categories (e.g., dropdown or multiselect).
  - Responsive Design: Ensure the app is mobile-friendly.

#### Backend

1. Framework: Node.js with NestJS or Express.js.
2. Database: MongoDB for data storage.
3. API Endpoints:
  - Questions:
    - POST /questions: Create a new question with title, description, and tags.
    - GET /questions: Fetch a paginated list of questions with optional filters (e.g., tags, search keyword).
    - GET /questions/:id: Fetch a single question by its ID along with its answers and replies.
  - Answers:
    - POST /questions/:id/answers: Add an answer to a question.
  - Replies:
    - POST /answers/:id/replies: Add a nested reply to an answer.
4. Tags/Categories:
  - Store tags/categories in a separate collection for reusability and management.

- Enable multi-tagging for questions.
5. Authentication:
- Implement JWT-based authentication to identify users.
  - Restrict certain actions (e.g., posting questions or answers) to authenticated users only.

## Features to Evaluate

### 1. Frontend Skills:

- Clear and responsive UI/UX.
- Efficient state management for nested replies and form handling.
- Implementing routing and dynamic pages using Next.js.

### 2. Backend Skills:

- Well-structured APIs with proper request validation (e.g., using class-validator in NestJS).
- Efficient MongoDB queries, especially for nested data (e.g., populate in Mongoose).
- Proper error handling and status codes.

### 3. Database Design:

- A clean schema for questions, answers, and replies.
- Indexed fields for optimized search and filtering.

### 4. Optional Advanced Features:

- Implement sorting options (e.g., latest questions or most answered questions).
- Markdown support for question and answer content.

## Deliverables

### - Frontend:

- A responsive application with the required pages (home, question detail, and post question).

### - Backend:

- RESTful APIs with clear documentation (e.g., using Swagger or Postman collections).

### - Database:

- A well-defined schema stored in MongoDB.

### - Deployment:

- Frontend hosted on Vercel or Netlify.
- Backend hosted on a cloud provider (e.g. Vercel).