Web Development Induction Program

Welcome to the Web Development Induction Program! As part of your journey in learning React and Next.js, you will have the opportunity to work on exciting projects. This document provides you with an overview of the tasks you can choose from and the instructions for starting your work.

Objective

The primary goal of this program is to familiarize you with frontend development concepts using React and Next.js and backend Technologies. You will work on either advanced tasks that include backend functionalities or beginner-friendly tasks that focus solely on frontend development.

Task Selection

Advanced Projects

If you feel confident and want to challenge yourself, you can choose one of the following advanced tasks. These tasks include backend functionalities and will provide valuable experience in full-stack development.

1. **Task 1: File Upload & Storage Platform**
    - Build a platform where users can upload PDF and image files, with unique links for access.
2. **Task 2: College Club Management Portal**
    - Create a dynamic website to manage and showcase college clubs, complete with club details, events, and member information.
3. **Task 3: Student Project Showcase and Portfolio Hub**
    - Develop a platform for students to showcase their projects and achievements, including profiles and project management.

Beginner-Friendly Tasks (Recommended for 1st year students)

If you're just starting out and want to focus on frontend skills, you can choose one of these simplified tasks:

- **Task A: College Clubs Directory**
    - Create a directory of college clubs, showcasing details such as names, descriptions, and images.
- **Task B: Event Countdown & Information Page**
    - Design a page displaying details of an upcoming event, complete with a countdown timer.

Instructions for Getting Started

1. **Choose Your Task**: Decide whether you want to undertake one of the advanced tasks or a beginner-friendly task. Remember, you are only required to complete **one task**.
2. **Set Up Your Environment**:
3. **Understand the Requirements**: Carefully read through the requirements for the task you've selected. Make sure you understand what is expected in terms of functionality, design, and deployment.
4. **Build Your Application**:
    - Start coding based on the requirements outlined for your chosen task.
    - Use React components to structure your application logically.
    - Keep your code organized and follow best practices for clean coding.
5. **Styling and Responsiveness**: Ensure your application is visually appealing and responsive on different devices. Utilize CSS modules or libraries like Tailwind CSS for styling.
6. **Deployment**:
    - Once you are satisfied with your project, deploy it on platforms like Vercel or Netlify.
    - Follow the respective instructions on these platforms for deployment.
7. **GitHub Submission**: Push your code to a GitHub repository, ensuring your code is well-documented. Share both the GitHub repository link and the live deployment link for submission.

Important Notes

- **Partial Completion**: It's acceptable to submit partially completed projects .
- **Best Practices**: Follow best practices for coding and design throughout the project.
- **Deadline**: Be mindful of submission deadlines and ensure your work is completed on time.

Final Reminder

This is a fantastic opportunity to develop your skills and gain hands-on experience in web development. Embrace the challenge, and don't hesitate to reach out for help if you encounter any issues. Good luck, and enjoy building your project!

Task 1: File Upload & Storage Platform

Objective:

Create a Next.js website where users can upload PDF and image files. Each uploaded file should be stored in a local folder, and the application should generate a unique link for viewing or downloading the file.

Requirements:

1. **Folder Structure**:
    - In the root of the project, create a folder named `Database`.
    - Organize uploaded files in date-based folders (e.g., `/Database/YYYY-MM-DD/`), each containing files uploaded on that date.
2. **Backend**:
    - Set up an API route in Next.js to handle file uploads.
    - Each uploaded file should be saved with a unique identifier to avoid conflicts and ensure that each file has a unique URL.
    - Store file metadata (e.g., original filename, upload date, unique link) in a JSON file or a simple database like SQLite.
3. **Frontend**:
    - Design a clean, user-friendly interface where users can:
        - Select PDF or image files for upload.
        - Submit files through an upload button.
        - Receive a unique link to access the uploaded file once the upload is successful.
    - Display a list of recently uploaded files with their access links.
4. **Error Handling**:
    - Validate file types to allow only PDFs and images (JPG, PNG).
    - Set a file size limit and display appropriate error messages for invalid inputs.
5. **Deployment**:
    - Deploy the website on Vercel or Netlify.
    - Push code to GitHub, keeping the `Database` folder in `.gitignore` to ensure that uploaded files are not included in the repository.
6. **Submission**:
    - Provide the GitHub repository link and the live site link.
    - Partial completion is allowed, as long as core functionality (upload, save, and unique link generation) is present.

Task 2: College Club Management Portal

Objective:

Create a dynamic website to manage and showcase college clubs, displaying details like club posters, activities, upcoming events, images, and member information. This task includes creating both backend and frontend with a user-friendly UI.

Requirements:

1. **Backend**:
    - **Data Storage**:
        - Store club information, including club name, description, activities, and member profiles in a MongoDB database.
    - **API Endpoints**:
        - Create API endpoints for:
            - Adding a new club, including uploading its main poster, description, and club-specific details.
            - Updating or deleting club information.
            - Fetching all club information for display on the frontend.
        - Separate endpoints for uploading event details, club images, and member data (name, role, profile image).
    - **File Uploads**:
        - Integrate file storage for club posters and event images (using local storage or a cloud solution like AWS S3 or Firebase).
2. **Frontend**:
    - **Homepage**:
        - Display a list of all clubs with a main poster image, club name, and a short description.
    - **Club Detail Page**:
        - Create a dedicated page for each club with:
            - Club description, activities, and achievements.
            - List of upcoming events with event descriptions, dates, and images.
            - A gallery section for images related to the club's events.
            - Member profiles with names, roles, and profile pictures.
    - **Dynamic Rendering**:
        - Use API calls to dynamically fetch and render club data.
3. **Design**:
    - Ensure the design is visually appealing and responsive across devices.
    - Implement a search and filter option to make it easy to find specific clubs or events.
4. **Additional Features**:
    - Add a login for club admins to manage their club information.
    - Include form validation to ensure all required fields are completed before submitting.
5. **Deployment**:
    - Deploy the application on Vercel or Netlify and link the backend with the frontend.
    - Push the source code to GitHub.
6. **Submission**:
    - Provide GitHub repository link and the live site link.
    - Partial submissions are acceptable, with core functionality implemented (adding clubs, viewing clubs, basic CRUD operations).

Task 3: Student Project Showcase and Portfolio Hub

Objective:

Develop a platform where students can upload and showcase their projects, achievements, and portfolios. This site will act as a digital portfolio for students and serve as a place to showcase their work and accomplishments.

Requirements:

1. **Backend**:
    - **Project and Portfolio Management**:
        - Set up API routes to add, update, and delete projects and portfolio entries.
        - Each project entry should include:
            - Project title, description, images, technologies used, links to GitHub, live demo, and any awards or recognitions.
        - A separate endpoint for managing student portfolios (skills, personal information, and experience).
    - **Tags and Categories**:
        - Implement categories (e.g., Web Development, AI, Robotics) for filtering projects.
        - Use tags for technologies (e.g., React, Python, Java) to enable search functionality.
2. **Frontend**:
    - **Project Gallery**:
        - Design a gallery view of student projects with search and filter options.
        - Include a detailed project view page, showing project images, descriptions, technologies, and links.
    - **Student Profiles**:
        - Create a profile page for each student, displaying:
            - Profile picture, name, department, skills, and experience.
            - A showcase of their top projects, awards, and recognitions.
    - **Achievements Section**:
        - A highlight section to display awards, competitions, and any significant recognitions.
3. **Additional Features**:
    - **Like and Comment Functionality**:
        - Allow users to like and comment on projects.
    - **Admin Controls**:
        - Admin interface for reviewing and approving project submissions to ensure quality control.
    - **Responsive Design**:
        - Ensure the application is mobile-friendly and has a modern, intuitive interface.
4. **Technical Details**:
    - Use the MERN stack (MongoDB, Express, React, Node.js) with a Next.js frontend.
    - Use a cloud storage solution (AWS S3 or Firebase) for uploading and storing images and files.
    - Implement user authentication to allow only registered students to upload and manage their projects and profiles.
5. **Deployment**:
    - Deploy the application on Vercel, Netlify, or a similar platform.
    - Push all code to GitHub, ensuring it is well-documented.
6. **Submission**:
    - Provide GitHub repository link and live deployment link.
    - Partial submissions are allowed, as long as basic functionality (project uploads, student profile creation, and gallery view) is complete.

Beginner - Friendly React/Next.js Tasks for 1st-Year Students

These tasks are designed for 1st-year students who are just getting started with React and Next.js. Both tasks are frontend-only, focusing on essential skills like component structure, state management, and responsive design. Follow the instructions carefully to complete these tasks, and feel free to add any creative touches!

Choose Your Task Level:

- You can opt to tackle one of the more advanced tasks listed in the **Top 3 Projects** section if you feel confident. These projects involve both frontend and backend work, are slightly more complex, and provide great hands-on experience with React/Next.js.
- If you're not ready for the advanced projects, you can start with the **Simplified Frontend-Only Tasks** below. These are designed to help you build a strong foundation in web development without backend requirements.

Task A: College Clubs Directory

Objective:

Build a simple directory showcasing different college clubs. This will allow students to practice creating and displaying lists, working with props, and structuring components in React/Next.js.

Requirements:

1. **Directory Layout**:
    - Display a list of college clubs, each as a card component.
    - Each club card should include:
        - Club name.
        - Club description.
        - An image or icon representing the club.
    - Provide a short list of 4-6 example clubs (e.g., Coding Club, Robotics Club, Drama Club, Sports Club).
2. **Component Structure**:
    - **ClubCard Component**:
        - Create a reusable `ClubCard` component that accepts `name`, `description`, and `image` as props.
    - **Main Page**:
        - Map over an array of club data and render multiple `ClubCard` components to display the clubs.
3. **Styling**:
    - Use CSS modules or inline styling for a clean and simple layout.
    - Style the cards with basic hover effects and a responsive design that displays cards in a grid on desktop and a single-column layout on mobile.
4. **Additional Features** (Optional):
    - Add a search bar to filter clubs by name.
    - Use conditional rendering to display a "No clubs found" message if the search yields no results.
5. **Deployment**:
    - Deploy the project on Vercel or Netlify and share the live link.
    - Push the code to a GitHub repository.

Task B: Event Countdown & Information Page

Objective:

Create a simple event information page that displays a countdown timer for an upcoming event. This task introduces students to working with state and effects in React for interactivity.

Requirements:

1. **Event Page Structure**:
    - Create a page with:
        - An event title (e.g., "Freshers' Party 2024").
        - Event date and time.
        - Brief description of the event.
2. **Countdown Timer**:
    - Display a countdown timer that updates in real-time to show days, hours, minutes, and seconds remaining until the event.
    - Use React's `useState` and `useEffect` hooks to calculate and update the countdown.
3. **Styling**:
    - Design the page with attractive visuals for the countdown timer and event details.
    - Ensure the design is responsive, with the timer and event information stacked vertically on smaller screens.
4. **Additional Features** (Optional):
    - Add a "More Info" button that toggles visibility of extra event details (e.g., event location, dress code).
    - Include a placeholder image banner for the event.
5. **Deployment**:
    - Deploy the project on Vercel or Netlify and provide the live link.
    - Push the code to a GitHub repository.

Instructions and Evaluation:

- **Deployment**: Each task must be deployed on a hosting platform for easy access.
- **Submission**: Submit GitHub and live deployment links before the deadline and should be made public so that we can see all codes.
- **Grading**: Partial marks are available based on the functionality level completed.
- **Guidelines**: Follow best practices for clean code, structured folders, responsive design, and proper documentation.

These tasks are designed to assess both frontend and backend skills while encouraging creativity and problem-solving in web development.