**Task Management Application Documentation**

**Front-end Implementation:**

1. **Landing Page Design:**
   * The landing page displays a list of tasks using a clean and intuitive layout to enhance user experience.
2. **Frameworks and Libraries:**
   * Utilized React.js for building the user interface due to its component-based architecture and efficiency.
   * Used normal CSS for styling and ensuring responsive design to maintain compatibility across various devices.
   * Implemented Redux Toolkit for state management to handle complex state logic efficiently.
3. **Adding New Tasks:**
   * Implemented a form with fields for title, description, and due date, allowing users to input and submit new tasks.
4. **Viewing Task Details:**
   * Included a detailed view modal or page to display complete information about a selected task.
5. **Editing Tasks:**
   * Enabled inline editing or a separate edit page/modal for modifying task details.
6. **Deleting Tasks:**
   * Added delete buttons with confirmation dialogs to safely remove tasks.
7. **Responsive Design:**
   * Ensured the layout adjusts seamlessly for both desktop and mobile devices using CSS media queries.

**Back-end Implementation:**

1. **Server-side Technology:**
   * Choose Node.js with Express.js for its asynchronous capabilities and strong support for RESTful APIs.
2. **RESTful API Design:**
   * Developed endpoints to handle CRUD operations:
     + Retrieve all tasks: GET /tasks
     + Create a new task: POST /tasks
     + Retrieve a task by ID: GET /tasks/:id
     + Update a task: PUT /tasks/:id
     + Delete a task: DELETE /tasks/:id
3. **Retrieving Tasks:**
   * Implemented the GET /tasks endpoint to fetch all tasks from the database.
4. **Creating Tasks:**
   * Structured the POST /tasks endpoint to accept and store new task data in MongoDB.
5. **Retrieving Single Task:**
   * Designed the GET /tasks/:id endpoint to fetch specific task details using the task ID.
6. **Updating Tasks:**
   * Used the PUT /tasks/:id endpoint to update existing task details based on the provided task ID.
7. **Deleting Tasks:**
   * Implemented the DELETE /tasks/:id endpoint to remove tasks from the database using the task ID.

**Database and Connection:**

1. **Database Technology:**
   * Used MongoDB for storing task data due to its flexibility and scalability.
   * Utilized Mongoose for connecting Node.js/Express with MongoDB, providing a straightforward schema-based solution.

**General Implementation:**

1. **Version Control:**
   * Managed the project using Git for version control to track changes and collaborate effectively.
2. **Clean Code Practices:**
   * Ensured the code is clean and well-structured with appropriate comments for better readability and maintainability.
3. **Error Handling:**
   * Implemented comprehensive error handling on both the front-end and back-end to manage and display errors gracefully.
4. **README.md File:**
   * Included detailed instructions in the README.md file on how to set up and run the application locally.
5. **Deployment:**
   * Deployed the application on a hosting platform such as Vercel for public access.

**Additional Notes and Explanations:**

1. **Design Choices:**
   * Chose a minimalist design for better user experience and ease of use.
   * Opted for React.js due to its flexibility and performance benefits.
   * Utilized Redux Toolkit for efficient state management.
2. **Challenges Faced:**
   * Managing state in React with Redux while ensuring responsiveness across different devices was challenging.
   * Implementing secure and efficient API endpoints required careful planning and testing.
3. **Future Improvements:**
   * Plan to implement user authentication and authorization to secure the application.
   * Add features like task categorization, tagging, and priority settings.
   * Improve the UI/UX based on user feedback and testing.
   * Make the website SEO-friendly by using Next.js to enable server-side rendering (SSR), static site generation (SSG), and incremental static regeneration (ISR).
4. **Additional Notes:**
   * Regularly updated the project repository and maintained thorough documentation.
   * Followed best practices for both front-end and back-end development to ensure scalability and maintainability.