Task Management Application Documentation

Overview

The Task Management Application is a web-based platform built using **Next.js** that allows users to manage their daily tasks efficiently. It provides features to create, view, edit, and delete tasks with persistence via **MongoDB**. The project demonstrates the power of Next.js as a full-stack framework and integrates server-side actions for backend logic.

Features

Core Features:

1. **Add Tasks**: Create tasks with details such as title, description, and due date.
2. **View Tasks**: View a list of all tasks in the system.
3. **Update Tasks**: Edit task details and mark them as complete/incomplete.
4. **Delete Tasks**: Remove unwanted tasks from the system.

Technologies Used

Frontend:

- **Next.js**: For server-rendered React pages and Server Actions.
- **TailwindCSS**: For building a responsive and clean UI.

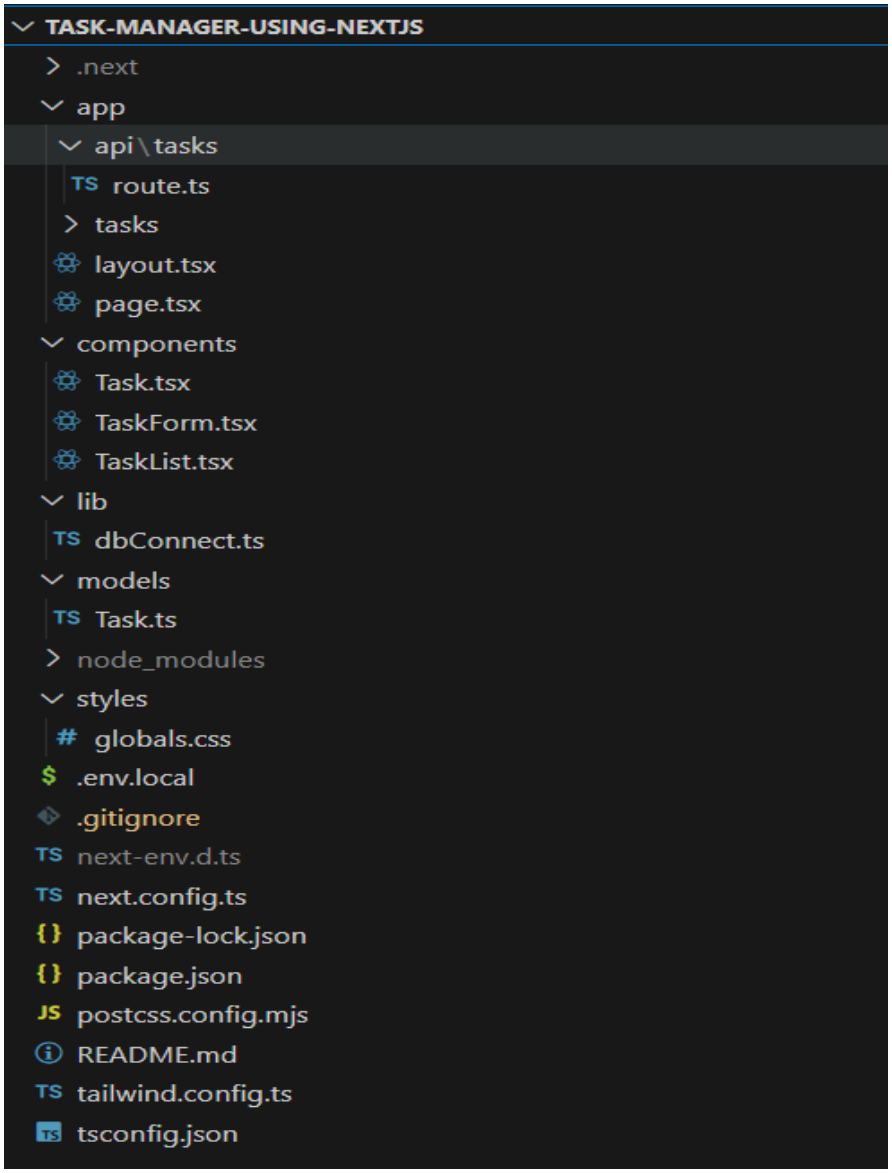
Backend:

- **Next.js API Routes**: Server-side API for task operations.
- **MongoDB**: Database for persistent data storage.

Deployment:

- **Vercel**: Hosting platform for the application.

Project Structure



Setup Instructions

Prerequisites:

- **Node.js** (version 14 or higher)
- **MongoDB** (local or cloud instance)
- **Vercel Account** (optional for deployment)

1. Clone the Repository

- git clone https://github.com/your-username/task-manager-using-nextjs.git
- cd task-manager-using-nextjs

2. Install Dependencies

npm install

3. Set Up Environment Variables

Create a **.env.local** file in the root directory and add the following:

MONGODB_URI=your-mongodb-connection-string

4. Run the Application Locally

npm run dev

Access the application at http://localhost:3000.

Deployment

1. Push the code to a GitHub repository.
2. Log in to **Vercel** and import the repository.
3. Set up the environment variables in Vercel's settings.
4. Deploy the project.

API Endpoints

Base URL: /api/tasks

1. **GET** /api/tasks

- *Fetch all tasks.*
    - Response:

```
[
 {
   "id": "task-id",
   "title": "Task Title",
   "description": "Task Description",
   "dueDate": "2025-02-15",
   "completed": false
```

```
  }
]
```

2. **POST /api/tasks**

- *Create a new task.*
    - Request:

```
{
  "title": "New Task",
  "description": "Task Description",
  "dueDate": "2025-02-15"
}
```

3. **PUT /api/tasks/:id**

- *Update a task.*
    - Request:

```
{
  "title": "Updated Title",
  "completed": true
}
```

4. **DELETE /api/tasks/:id**
    - *Delete a task.*

Key Components

1. **TaskList.tsx**:
    - Displays the list of tasks dynamically fetched from the backend.
2. **TaskForm.tsx**:

- Form to add or edit tasks, with validation logic.

3. **Task.tsx**:
    - Represents individual tasks with actions (create, edit, delete).
