Task Management Application Documentation

Task Management Application

Overview

The **Task Management Application** is a web-based platform built using **Next.js** that helps users manage their daily tasks efficiently. It provides functionality to create, view, edit, and delete tasks with persistence using **MongoDB**. This project demonstrates the power of **Next.js** as a full-stack framework and integrates server-side actions for backend logic. Additionally, features like **user authentication**, **JWT tokens**, and **session management** are implemented to ensure that users can only view their own tasks.

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 for the authenticated user.
3. **Update Tasks**: Edit task details and mark them as complete/incomplete.
4. **Delete Tasks**: Remove unwanted tasks from the system.
5. **User Authentication**: Secure login and registration using **JWT tokens** and **bcrypt** for password hashing.
6. **Responsive Design**: Built with **TailwindCSS** to ensure a mobile-friendly and clean UI.
7. **Toast Notifications**: User-friendly notifications are displayed using **React-Toastify**.

Technologies Used

Frontend:

- **Next.js**: Server-rendered React pages and Server Actions.
- **TailwindCSS**: For building a responsive and clean UI.
- **React-Toastify**: Beautiful notifications for user feedback.

Backend:

- **Next.js API Routes**: Server-side API for task operations.
- **MongoDB**: Persistent data storage.
- **JWT Tokens**: For user authentication and session management.
- **bcrypt**: For password hashing and security.

Deployment:

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

Project Structure

```
NEXTJS-TASK-MANAGER
├── .next
├── app
│   ├── api
│   │   └── auth
│   │       └── route.ts
│   │   └── tasks
│   │       └── route.ts
│   ├── login
│   │   └── page.tsx
│   ├── register
│   │   └── page.tsx
│   ├── layout.tsx
│   ├── page.tsx
├── components
│   ├── Footer.tsx
```

```
│   ├── Login.tsx
│   ├── Navbar.tsx
│   ├── Register.tsx
│   ├── Task.tsx
│   ├── TaskForm.tsx
│   └── TaskList.tsx
├── lib
│   └── dbConnect.ts
├── models
│   └── Task.ts
│   └── User.ts
├── node_modules
├── styles
│   └── globals.css
├── .env.local
├── .gitignore
├── draft.md
├── next-env.d.ts
├── next.config.ts
├── package-lock.json
├── package.json
├── postcss.config.mjs
├── tailwind.config.ts
└── tsconfig.json
```

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/Sayan-Rajak-Das/Task-Management-App-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
JWT_SECRET=your-secret-key
```

4. Run the Application Locally

```
npm run dev
```

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

Deployment

Steps:

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 on **Vercel**.

API Endpoints

Base URL: `/api/tasks`

1. GET /api/tasks

- **Fetch all tasks** for the logged-in user.

  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).

Conclusion

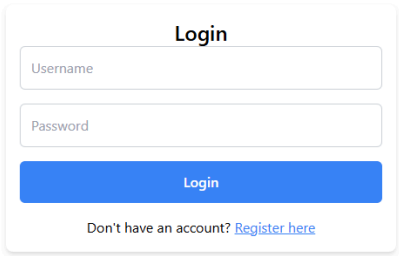
This **Task Management Application** demonstrates how to build a full-stack application with **Next.js** and **MongoDB**, covering all essential CRUD operations. With the integration of **Vercel**, it’s ready to be deployed with minimal configuration.

User Authentication & Security

- The application features **user authentication** using **JWT tokens**.
- Passwords are securely stored using **bcrypt** for hashing.
- **Session management** ensures that only the current user can view their own tasks and not those of others.

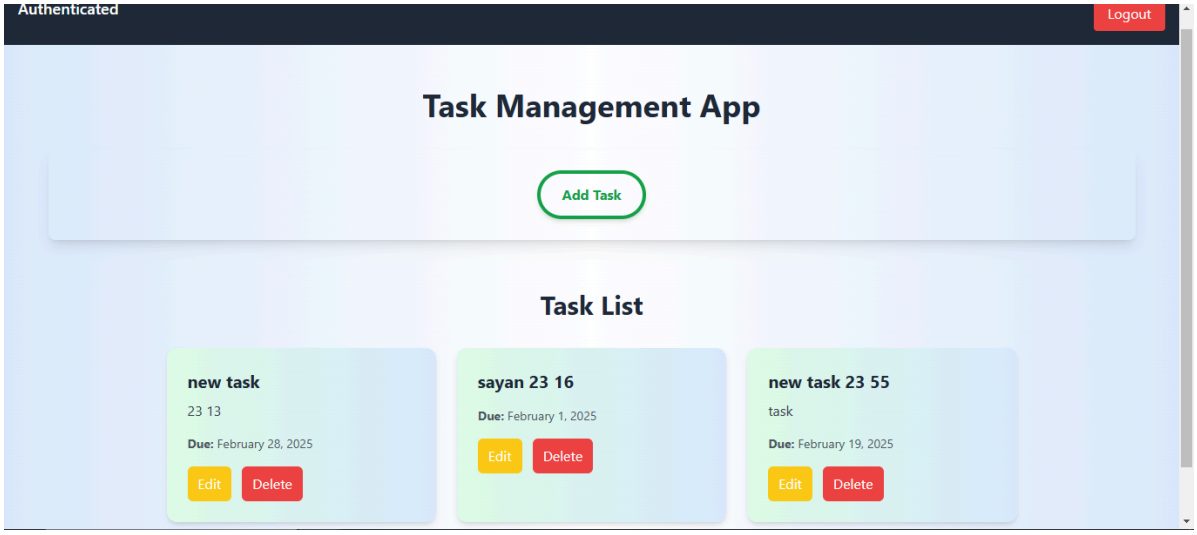
Screenshots

1. **Login Page:**

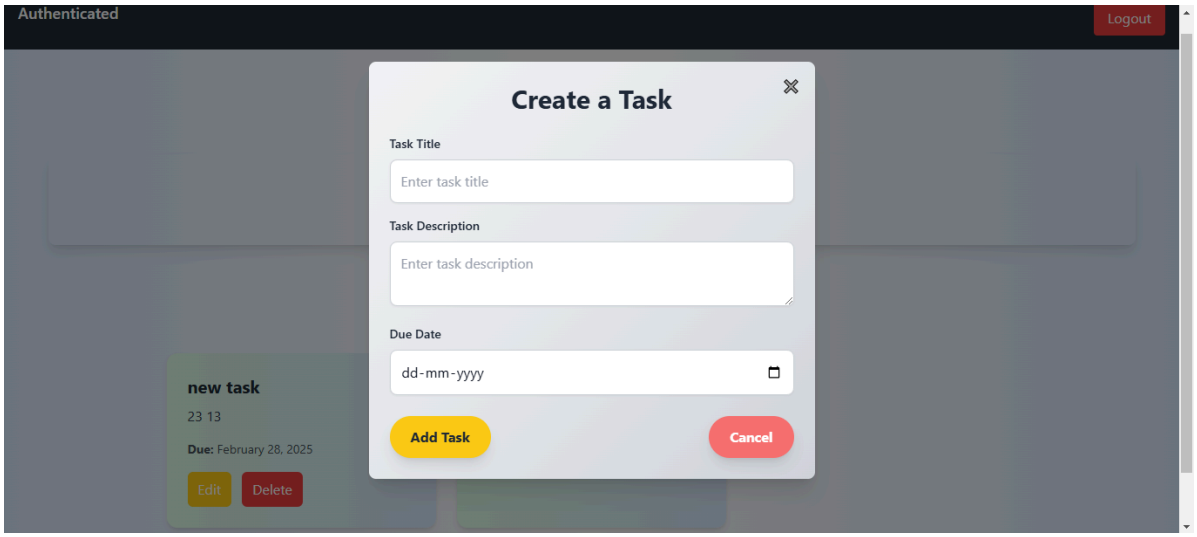


---

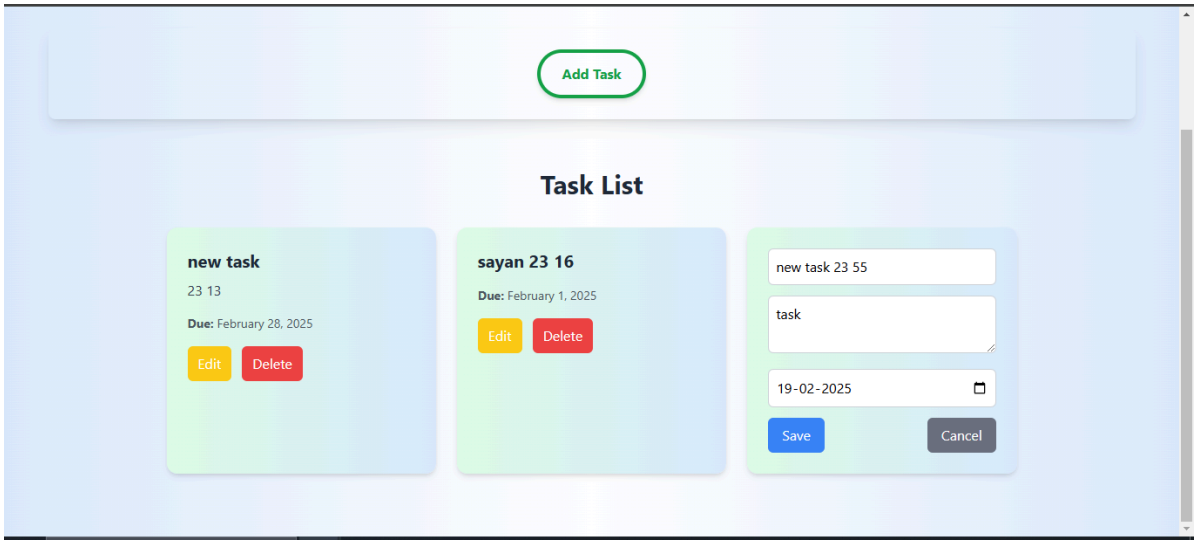
2. **Task List Page:**



3. **Add Task Form:**



4. **Edit Task:**



Links

- **GitHub Repository**: https://github.com/Sayan-Rajak-Das/Task-Management-App-Using-NextJs
- **Vercel Deployment**: https://task-management-app-using-next-js.vercel.app/