Task Management Application

Overview

This project is a Task Management Application built using a Python backend (Flask API) and a React frontend. It allows users to manage tasks by performing CRUD operations: viewing, adding, editing, and deleting tasks. The application uses an SQLite database for storage.

Features

- View Tasks: Display a list of tasks.
- Add Tasks: Create new tasks with a title, description, and completion status.
- Edit Tasks: Update task details and completion status.
- Delete Tasks: Remove tasks from the list.

Technologies Used

- Backend: Flask, Flask-RESTful, Flask-CORS, Flask-SQLAlchemy
- Frontend: React, JavaScript, HTML, CSS
- Database: SQLite

Setup Instructions

Backend Setup

1. Clone the repository:
   git clone <repository_url>
   cd <repository_folder>

2. Set up a virtual environment:
   python -m venv venv
   source venv/bin/activate # For MacOS/Linux
   venv\Scripts\activate   # For Windows

3. Install required Python packages:
   pip install flask flask-restful flask-sqlalchemy flask-cors

4. Run the backend server:
   python create_db.py

5. Run the backend server:
   python app.py
   The backend server will run on http://localhost:5000.

Frontend Setup

1. Navigate to the frontend folder:
   cd frontend

2. Install the dependencies:
   npm install

3. Start the React development server:
   npm start
   The frontend will run on http://localhost:3000.

Database Schema

The database consists of a single table named tasks:

Column Name	Data Type	Description
id	Integer	Primary key, unique ID for tasks.
title	String	The title of the task.
description	Text	A detailed description of the task.
completed	Boolean	Indicates whether the task is completed.

API Endpoints

GET /tasks

Retrieve a list of all tasks.

Response:

[{ 'id': 1, 'title': 'Sample Task', 'description': 'This is a task description.', 'completed': false }]

GET /tasks/{id}

Retrieve a specific task by its ID.

Response:

{ 'id': 1, 'title': 'Sample Task', 'description': 'This is a task description.', 'completed': false }

POST /tasks

Create a new task.

Request Body:

{ 'title': 'New Task', 'description': 'Description of the task', 'completed': false }

Response:

{ 'id': 2, 'title': 'New Task', 'description': 'Description of the task', 'completed': false }

PUT /tasks/{id}

Update an existing task by its ID.

Request Body:

{ 'title': 'Updated Task', 'description': 'Updated description', 'completed': true }

Response:

{ 'id': 1, 'title': 'Updated Task', 'description': 'Updated description', 'completed': true }

DELETE /tasks/{id}

Delete a task by its ID.

Response:

{ 'message': 'Task deleted successfully.' }