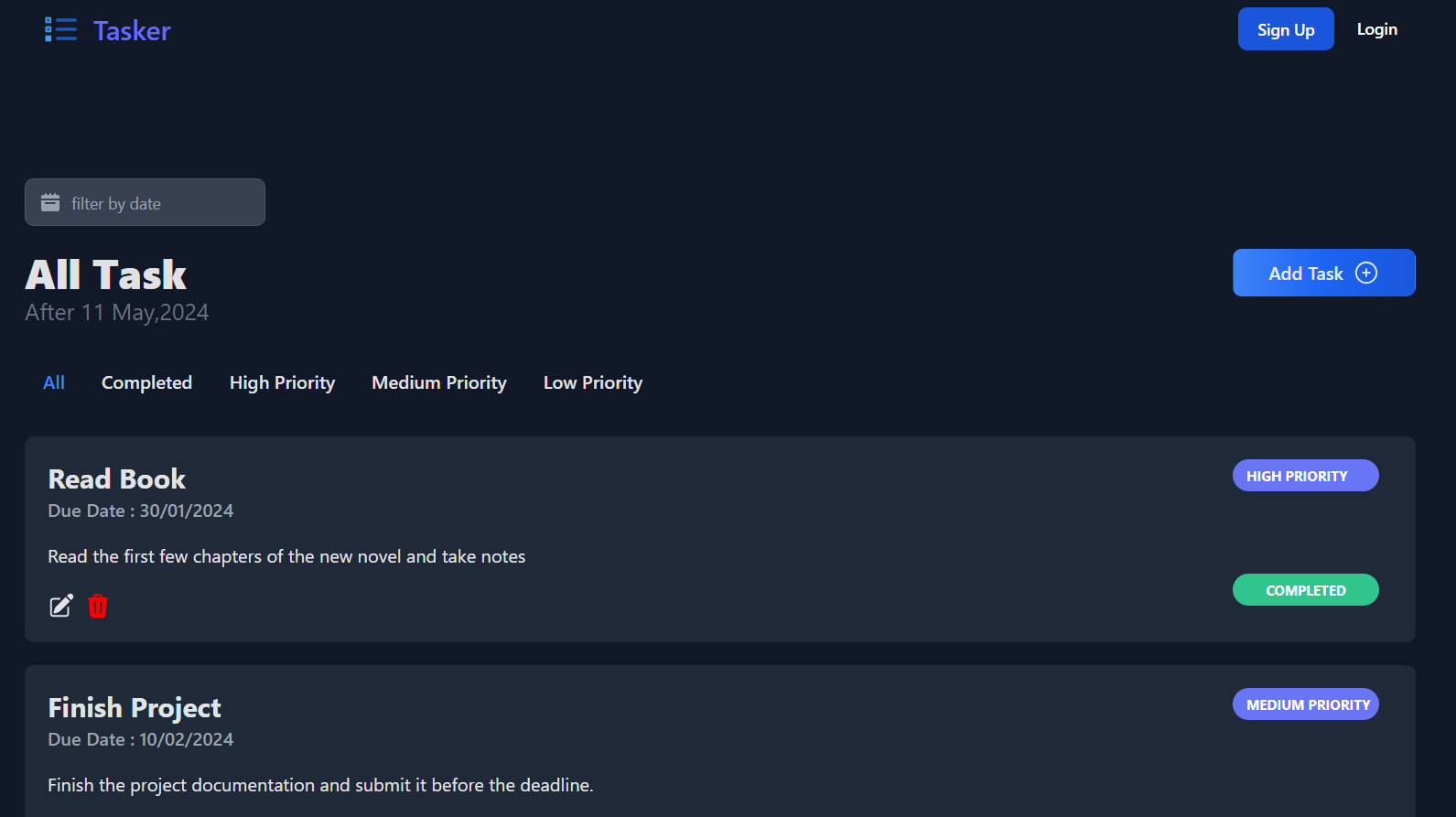
# Task Management Application

This project is a simple task management application designed to showcase full-stack development skills. The application allows users to create, edit, and delete tasks, prioritize tasks, set due dates, mark tasks as completed, and provides filtering and search functionality for tasks.

Link to Live demo : <https://task-management-app-two-psi.vercel.app>

Github repo : <https://github.com/shubham691438/task-management-app>

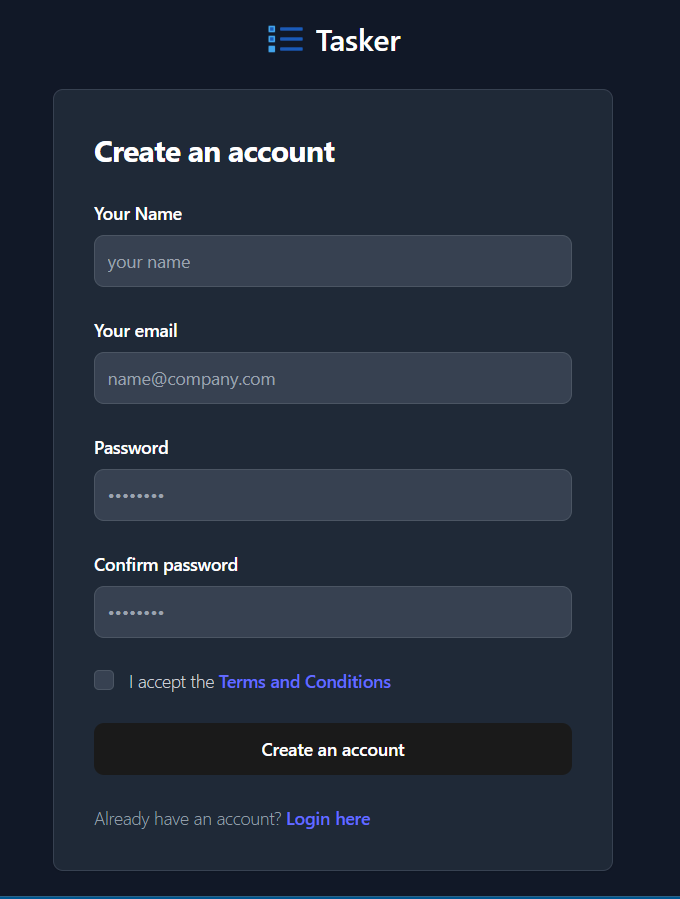
Link to Drive: <https://drive.google.com/file/d/1K8t-vCgq2GaR02eEHEcq0Fwfa2SGuM_k/view?usp=sharing>



## Features

### Frontend

* User-friendly and intuitive interface developed React and Tailwind
* User registration and login functionality
* Task prioritization (e.g., high, medium, low).
* Setting due dates for tasks.
* Marking tasks as completed.
* Filtering and search functionality for tasks.



### Backend

* Build using nodejs , express and Mongodb
* CRUD operations for tasks (Create, Read, Update, Delete)
* Secure user authentication and authorization.
* RESTful API endpoints for CRUD operations on tasks.

# API Documentation

## Task Management API

### Base URL

The base URL for accessing the Task Management API is /api/task.

### Endpoints

#### 1. Create Task

* **Endpoint**: /create
* **Method**: POST
* **Description**: Creates a new task.
* **Request Body**:
  + user (string): User ID associated with the task.
  + taskName (string): Name of the task.
  + dueDate (date): Due date of the task.
  + description (string): Description of the task.
  + priority (string): Priority level of the task (e.g., high, medium, low).
  + completed (boolean): Indicates whether the task is completed.
* **Response**:
  + task (object): Created task object.
* **Status Codes**:
  + 201: Task created successfully.
  + 400: Error in task creation.

#### 2. Get Tasks

* **Endpoint**: /get
* **Method**: GET
* **Description**: Retrieves all tasks.
* **Response**:
  + tasks (array): Array of task objects.
* **Status Codes**:
  + 200: Successful retrieval of tasks.
  + 400: Error in retrieving tasks.

#### 3. Delete Task

* **Endpoint**: /delete/:\_id
* **Method**: DELETE
* **Description**: Deletes a task by ID.
* **Request Parameters**:
  + \_id (string): ID of the task to be deleted.
* **Response**:
  + task (object): Deleted task object.
* **Status Codes**:
  + 200: Task deleted successfully.
  + 400: Error in deleting task.

#### 4. Update Task

* **Endpoint**: /update/:\_id
* **Method**: PUT
* **Description**: Updates a task by ID.
* **Request Parameters**:
  + \_id (string): ID of the task to be updated.
* **Request Body**:
  + user (string): Updated user ID associated with the task.
  + taskName (string): Updated name of the task.
  + dueDate (date): Updated due date of the task.
  + description (string): Updated description of the task.
  + priority (string): Updated priority level of the task.
  + completed (boolean): Updated completion status of the task.
* **Response**:
  + task (object): Updated task object.
* **Status Codes**:
  + 200: Task updated successfully.
  + 400: Error in updating task.

## User Signup/Login API

### Base URL

The base URL for accessing the Authentication API is /api/use.

### Endpoints

#### 1. User Registration

* **Endpoint**: /register
* **Method**: POST
* **Description**: Registers a new user.
* **Request Body**:
  + name (string): Name of the user.
  + email (string): Email address of the user.
  + password (string): Password of the user.
* **Response**:
  + userId (string): ID of the registered user.
  + name (string): Name of the registered user.
  + email (string): Email address of the registered user.
  + token (string): JWT token for authentication.
* **Status Codes**:
  + 200: User registered successfully.
  + 400: Error in user registration.

#### 2. User Login

* **Endpoint**: /login
* **Method**: POST
* **Description**: Logs in an existing user.
* **Request Body**:
  + email (string): Email address of the user.
  + password (string): Password of the user.
* **Response**:
  + userId (string): ID of the logged-in user.
  + name (string): Name of the logged-in user.
  + email (string): Email address of the logged-in user.
  + token (string): JWT token for authentication.
* **Status Codes**:
  + 200: User logged in successfully.
  + 400: Error in user login.