

Task Management System – Technical Documentation

1. Overview

A full-stack web application to manage personal tasks.

Tech Stack:

- **Frontend:** React
- **Backend:** Node.js + Express
- **Database:** MongoDB
- **Authentication:** JSON Web Tokens (JWT)

Core Capabilities for Authenticated Users:

- Register and log in
 - Create, view, update, and delete tasks
 - Mark tasks as completed
 - View real-time feedback and task updates
-

2. Authentication System

Registration

- **Endpoint:** `POST /register`
- **Required Fields:**
 - `username`: 3–30 characters
 - `password`: Minimum 8 characters
- **Password Storage:** Hashed using bcrypt

- **On Success:** A JWT token is returned
- **Validation Errors:**
 - Username too short/long
 - Password too short
 - Username already exists

Login


- **Endpoint:** `POST /login`
- **Required Fields:** username, password
- **Success:** Returns JWT token
- **Failure Cases:**
 - Invalid credentials
 - Nonexistent user
- **After Login:** User's tasks are loaded automatically

Logout

- the logout button will be displayed only after login ,its where u can (view tasks ,create etc..)
- UI is cleared (no residual data)
- User is redirected to the login view if clicked on button

3. Task Management

Task Creation

- **Endpoint:** `POST /tasks` ( JWT Required)
- **Fields:**

- **title** (required)
- **description** (optional)
- **priority**: One of **low**, **medium**, **high** (default: medium)
- **categoryId**: one of work, hacked (in task list ,if work chosen it will be colored with red if hacked chosen it will be with black)
- **dueDate**: Date in the future (required)
- **Validations:**
 - Title must not be empty
 - Due date must be today or in the future
 - Priority must be one of the allowed values
 - Category must exist (if provided)
- **Failures:**
 - Missing title or invalid due date → 400
 - Invalid priority or category → 400

Task Listing

- **Endpoint:** **GET** `/tasks` (🔒 JWT Required)
- **Returns:** All tasks belonging to the authenticated user

Marking Task as Complete

- **Endpoint:** **PATCH** `/tasks/:id`
- **Effect:** Updates **completed** status to **true**

Task Deletion

- **Endpoint:** **DELETE** `/tasks/:id`

- **Behavior:** Requires confirmation on frontend; deletes task permanently
-

5. 🎨 Frontend (React)

Auth Section

- Login/Register forms with real-time validation
- Password field: hidden input with min length validation
- Errors shown immediately on input blur or form submit

Task Form

- Controlled form containing:
 - **Title** (text)
 - **Description** (textarea)
 - **Priority** (dropdown: low, medium, high)
 - **Category** (dropdown populated from API)
 - **Due Date** (date picker - minimum date = today)

Task List

- Tasks rendered as **cards**:
 - Title, description, due date, priority, category
 - Completion checkbox
 - Delete button with confirmation prompt

Additional UX Features:

-  Success and error messages displayed after all major actions

- 🔄 Tasks list auto-refreshes after task creation or deletion
- 🗑️ Logout clears all data and redirects to login
- 🟢 Completed tasks shown with **green background**
- 🔴 Overdue tasks show **red indicator**
- ⌚ Loading spinner shown during async operations

6. 🔄 API Overview (with Expected Behaviors)

Method	Endpoint	Description	Auth Required	Success Response	Failure Scenarios
POST	<code>/register</code>	Create new user	❌ No	JWT token	Username taken, invalid input
POST	<code>/login</code>	Authenticate user	❌ No	JWT token	Wrong credentials, user not found
GET	<code>/tasks</code>	List user's tasks	✅ Yes	Array of tasks	Invalid/missing token
POST	<code>/tasks</code>	Create a new task	✅ Yes	Task object	Missing/invalid input, past due date
PATCH	<code>/tasks/:id</code>	Update (mark complete)	✅ Yes	Updated task	Task not found, unauthorized
DELETE	<code>/tasks/:id</code>	Delete task	✅ Yes	Deletion confirmation	Task not found, unauthorized