TASK MANAGEMENT APP DESIGN DOCUMENT

1. OVERVIEW

This document outlines the design of a Task Management Application built with a React frontend, Spring Boot backend, and MongoDB Atlas database. The app allows users to create, view, and delete tasks through a web interface.

Base URL: http://localhost:3000

2. COMPONENTS

2.1 TaskList Page

The TaskList Page displays all tasks fetched from the backend and serves as the landing page.

- Route: TaskList at /tasks
- Uses API: GET /api/tasks
- Contains Elements:
  - List display with data-testid="task-list"
  - Checkbox with data-testid="task-complete"
  - Add button with data-testid="add-task-btn"
- Description: Displays a list of tasks with checkboxes to mark completion.

Navigates-to TaskForm with button data-testid="add-task-btn"

2.2 TaskForm Page

The TaskForm Page allows users to create new tasks.

- Route: TaskForm at /add-task
- Uses API: POST /api/tasks
- Contains Elements:
  - Title input with data-testid="task-title"
  - Description input with data-testid="task-desc"
  - Submit button with data-testid="submit-task"
- Description: Provides a form to input task details and submit them.

Navigates-to TaskList with button data-testid="submit-task" (on successful submission)

2.3 DeleteTasks Page

The DeleteTasks Page enables users to delete existing tasks.

- Route: DeleteTasks at /delete-tasks
- Uses API: GET /api/tasks (to fetch tasks)
- Uses API: DELETE /api/tasks/ (to delete a task)
- Contains Elements:
  - Task list with data-testid="delete-task-list"

    - Delete button with data-testid="delete-task-" (dynamic per task)
    - Back button with data-testid="back-to-tasks"
- Description: Shows a list of tasks with delete options.

Navigates-to TaskList with button data-testid="back-to-tasks"

3. API ENDPOINTS

3.1 Task Retrieval

- API: GET /api/tasks
    - Description: Fetches all tasks from the database.
    - Response: JSON array of tasks.

3.2 Task Creation

- API: POST /api/tasks
    - Description: Creates a new task.
    - Request Body: {"title": "string", "description": "string", "completed": boolean}
    - Response: Created task JSON.

3.3 Task Update

- API: PUT /api/tasks/
    - Description: Updates a task's completion status.
    - Request Body: {"title": "string", "description": "string", "completed": boolean}
    - Response: Updated task JSON.

3.4 Task Deletion

- API: DELETE /api/tasks/
    - Description: Deletes a task by ID.
    - Response: 200 OK or 404 Not Found.

4. NAVIGATION

- TaskList navigates-to TaskForm with button data-testid="add-task-btn".
- TaskForm navigates-to TaskList with button data-testid="submit-task" (on success).
- DeleteTasks navigates-to TaskList with button data-testid="back-to-tasks".
- Navigation links at the top:
    - TaskList at /tasks with data-testid="nav-tasks".
    - TaskForm at /add-task with data-testid="nav-add-task".
    - DeleteTasks at /delete-tasks with data-testid="nav-delete-tasks".

5. AUTHENTICATION

- No login required for this version.
- Future enhancement: Requires Login for all pages with credentials ("admin", "password123").

6. DATA MODEL

Tasks are stored in MongoDB Atlas in the `tasks` collection:

- Title: String (required)
- Description: String (optional)
- Completed: Boolean (default: false)
- Sample: {"title": "Write code", "description": "Finish React app", "completed": false}

7. DEPLOYMENT

- Backend: Spring Boot at http://localhost:8080
- Frontend: React at http://localhost:3000
- Database: MongoDB Atlas cloud instance

8. NOTES

- The TaskList Page is the first page users see upon loading the app.
- All components use styled CSS for a presentable UI.