

Smart Task Organizer - Final Report (front end)

1. Introduction

The Smart Task Organizer is a task management system designed to help users organize, track, and manage daily tasks efficiently.

The system provides full CRUD functionality, task sorting and filtering, automatic data persistence, and task exporting through a RESTful API.

The project follows clean architecture principles and applies software engineering best practices.

2. Functional Requirements (FR1–FR11)

FR1: The system shall allow the user to create a new task with title, description, deadline, and priority.

FR2: The system shall allow the user to edit an existing task.

FR3: The system shall allow the user to delete a task.

FR4: The system shall display a list of all tasks.

FR5: The system shall set the task status to (ToDo) by default.

FR6: The system shall allow the user to mark a task as “Completed”.

FR7: The system shall allow sorting tasks by deadline and priority.

FR8: The system shall allow filtering tasks to show completed, not-completed, and high-priority tasks.

FR9: The system shall save all tasks automatically when the system closes.

FR10: The system shall load all saved tasks automatically when the system starts.

FR11: The system shall allow the user to export all tasks to a text file.

3. Non-Functional Requirements

NFR1: The system shall respond to user actions within 2 seconds.

NFR2: The user interface shall be simple and easy to use.

NFR3: All data shall be stored locally in a readable format (JSON).

NFR4: The system code shall follow clean code rules and be analyzed using SonarQube.

NFR5: The system architecture shall be analyzed using JArchitect.

Project Overview

Smart Task Organizer is a system that helps users manage daily tasks efficiently through a user-friendly interface and RESTful API.

Functional Requirements

The system supports task creation, editing, deletion, sorting, filtering, saving, loading, exporting, and completion tracking.

Non-Functional Requirements

The system ensures fast response time, clean code, local storage, and usability.

Test Cases

Test Case ID: TC-01

Requirement: FR1

Objective: Create Task

Precondition: User logged in

Steps: Enter task details

Expected Result: Task created successfully

Test Case ID: TC-02

Requirement: FR2

Objective: Edit Task

Precondition: Task exists

Steps: Modify task

Expected Result: Task updated

REST API

POST /tasks
GET /tasks
PUT /tasks/{id}
DELETE /tasks/{id}
POST /export

Design Patterns

Singleton Pattern used for Task Storage
MVC Architecture for clean separation

Tools

SonarQube used for code quality
JArchitect used for architecture analysis

& Jira GitHub

Project hosted on GitHub with daily commits
Jira used for sprint and task tracking

Design Patterns

1 MVC Pattern

السبب في استخدامه:

، مما يسهل الصيانة والاختبار (Controller) عن البيانات (Model) عن المنطق (View) فصل الواجهة

Model: Task, User

Controller: taskController.js

View: HTML / CSS

2 Singleton Pattern

السبب في استخدامه:

ضمان وجود اتصال واحد مع قاعدة البيانات فقط.

إدارة موارد النظام بشكل مركزى وتجنب التكرار.

3 Factory Pattern

السبب في استخدامه:

إنشاء بطريقة منظمة وقابلة للتوسعة بسهولة (مثل Task Objects)

تسهيل إضافة أنواع جديدة من المهام أو الكائنات دون تغيير الكود الأساسي

Test Cases

Test Case ID	Requirement ID	Test Objective	Preconditions	Test Steps	Expected Result
TC-01	FR1	التحقق من إنشاء حساب جديد	المستخدم غير مسجل	فتح صفحة التسجيل 1. إدخال اسم مستخدم وكلمة مرور صحيحة 2. الضغط على Sign Up	يتم إنشاء الحساب بنجاح وتخزينه
TC-02	FR2	التحقق من تسجيل الدخول	الحساب موجود	إدخال بيانات صحيحة 1. إدخال بيانات Login 2. الضغط على	يتم الدخول للنظام بنجاح
TC-03	FR3	التحقق من رفض تسجيل دخول خاطئ	الحساب موجود	إدخال كلمة مرور خاطئة 1. إدخال كلمات السر خطأ	ظهور رسالة خطأ
TC-04	FR4	إضافة مهمة جديدة	المستخدم مسجل دخول	إدخال بيانات المهمة 1. إدخال بيانات المهمة Add Task 2. الضغط	تضاف المهمة لقائمة
TC-05	FR5	عرض جميع المهام	مهام موجودة	فتح Dashboard	ظهور جميع المهام
TC-06	FR6	تعديل حالة المهمة	مهمة موجودة	الضغط Complete Task	تحوّل الحالة إلى Completed
TC-07	FR7	حذف مهمة	مهمة موجودة	الضغط Delete Task	تحذف المهمة
TC-08	FR8	عرض الإحصائيات Dashboard	مهام متعددة	فتح Dashboard	ظهور الإحصائيات صحيحة
TC-09	FR9	تسجيل الخروج	المستخدم مسجل	الضغط Logout	يعود لصفحة الدخول
TC-10	FR10	حفظ المهام لكل مستخدم	أكثر من مستخدم	تسجيل الدخول بمستخدم آخر	ظهور مهام فقط
TC-11	FR11	تصدير المهام	مهام موجودة	الضغط Export Tasks	يتم تنزيل ملف المهام

Functional Requirements (FR1–FR11)

ID	Functional Requirement
FR1	النظام يسمح بإنشاء حساب
FR2	النظام يسمح بتسجيل الدخول
FR3	النظام يتحقق من صحة بيانات الدخول
FR4	المستخدم يستطيع إضافة مهمة
FR5	المستخدم يستطيع عرض المهام
FR6	المستخدم يستطيع إكمال المهمة
FR7	المستخدم يستطيع حذف مهمة
FR8	عرض الإحصائيات Dashboard
FR9	المستخدم يستطيع تسجيل الخروج
FR10	النظام يحفظ مهام كل مستخدم
FR11	المستخدم يستطيع تصدير المهام

Authentication

```
POST /api/auth/register  
POST /api/auth/login  
POST /api/auth/logout
```

Tasks CRUD

```
POST /api/tasks      → Create Task  
GET  /api/tasks      → Read Tasks  
PUT  /api/tasks/:id  → Update Task  
DELETE /api/tasks/:id → Delete Task
```

Dashboard

```
GET /api/dashboard
```

SonarQube:

للتأكد من جودة الكود والتنظيم (HTML/CSS/JS) تم تحليل كود الواجهة الأمامية.

للتأكد من خلوه من JavaScript/Backend Code Smells.

عبر قياس مدى سهولة تعديل الكود وإضافة مهام جديدة Maintainability تم التحقق من

التأكد من أن الدوال والฟناات ليست معقدة بشكل مفرط، مما يسهل اختبارها وصيانتها Complexity تم قياس

JArchitect:

تحليل هيكلية المشروع بالكامل، بما في ذلك Models (Task, User) وـ Controllers (taskController.js).

التأكد من أن الفناات والكائنات متراقبة بشكل صحيح ولا توجد تبعيات معقدة غير ضرورية.

لضمان قابلية التوسيع والصيانة MVC + Singleton + Factory Pattern تحسين تصميم المشروع حسب

تسجيل الدخول

لا تملك حساب؟ [إنشاء حساب](#)

مرحبا mary!

كيف حالك اليوم؟ هل أنت جاهز لتخطيم يومك وكتوبين مهامك؟

أضف مهمة جديدة

yyyy/mm/dd

عالية

قائمة المهام

العنوان	الوصف	الموعد النهائي	الأولوية	الحالة	أزرار
تسليم مشروع المختبر	ويب سايت لإدارة المهام اليومية	2025-12-28	High	Completed	<input checked="" style="width: 100px; height: 30px; border: 1px solid #ccc; border-radius: 5px; padding: 5px; font-size: 12px; color: #ccc; font-family: inherit; outline: none; background-color: #fff;" type="button" value="Complete"/> <input style="width: 100px; height: 30px; border: 1px solid #ccc; border-radius: 5px; padding: 5px; font-size: 12px; color: #ccc; font-family: inherit; outline: none; background-color: #fff;" type="button" value="Delete"/>

Dashboard

المستخدم الحالي : mary

عدد المهام الكلي : 1

عدد المهام المكتملة : 1