

# SMART TASK ANALYZER

## Smart Task Analyzer

Intelligently prioritize your tasks based on multiple factors

**Name** : Bojja Naga Bhavya Charitha  
**Email** : bojjanagabhavyacharitha@gmail.com  
**Phone** : 9391923745  
**LinkedIn** : [bhavya](#)  
**Github** : [bhavya](#)  
**Position** : Full Stack Development Intern  
**Company** : Singularium  
**Video** : [link](#)  
**Date** : 27 Nov 2025

# 1. INTRODUCTION

- **The Smart Task Analyzer** is an intelligent prioritization system that turns subjective task management into data-driven optimization. It addresses the critical challenge of effective task sequencing that both professionals and teams face in today's fast-moving work environments.
- **Traditional task management** relies on manual prioritization, which leads to decision fatigue, inconsistent sequencing, and overlooked dependencies. This leads to wasted time, missed deadlines, and lower productivity. The Smart Task Analyzer overcomes these challenges with an advanced algorithmic approach that weighs multiple dimensions of every task to determine the best execution order.
- This system leverages computational logic to assess four crucial factors: **urgency, importance, effort, and dependencies**. By converting these aspects into quantitative scores, it provides objective prioritization, removing personal biases. As a result, the user always works on the most valuable tasks...
- Importantly, **the Smart Task Analyzer** has been enhanced with **smart abilities to resolve dependencies automatically, track progress, and adaptively select strategies**. The result is a seamless workflow in which users address execution aspects while the system resolves complex priority decisions.
- Engineered on top of modern web technologies with a responsive interface and robust architecture, this system embodies professional-grade software development while handling real-world productivity challenges.

## 2. Problem Statement & Vision

### The Problem:

Traditional task management approaches suffer from several critical limitations:

- **Manual Prioritization Burden** : Users waste significant time deciding what to work on next.
- **Subjectivity and Bias** : Personal preferences often override optimal task sequencing.
- **Dependency Blindness** : Circular dependencies and blocking tasks go undetected.
- **Inconsistent Decision-Making** : Lack of standardized prioritization leads to inefficient workflows.

### The Vision:

Transform task management from subjective guesswork to scientific optimization by creating an intelligent system that:

- **Automates Complex Prioritization** : Uses data-driven algorithms to determine optimal task sequences.
- **Provides Objective Insights** : Eliminates personal bias through consistent scoring methodology.
- **Prevents Workflow Deadlocks** : Automatically detects and resolves dependency conflicts.
- **Adapts to User Needs** : Offers multiple prioritization strategies for different work styles.

## **The Impact:**

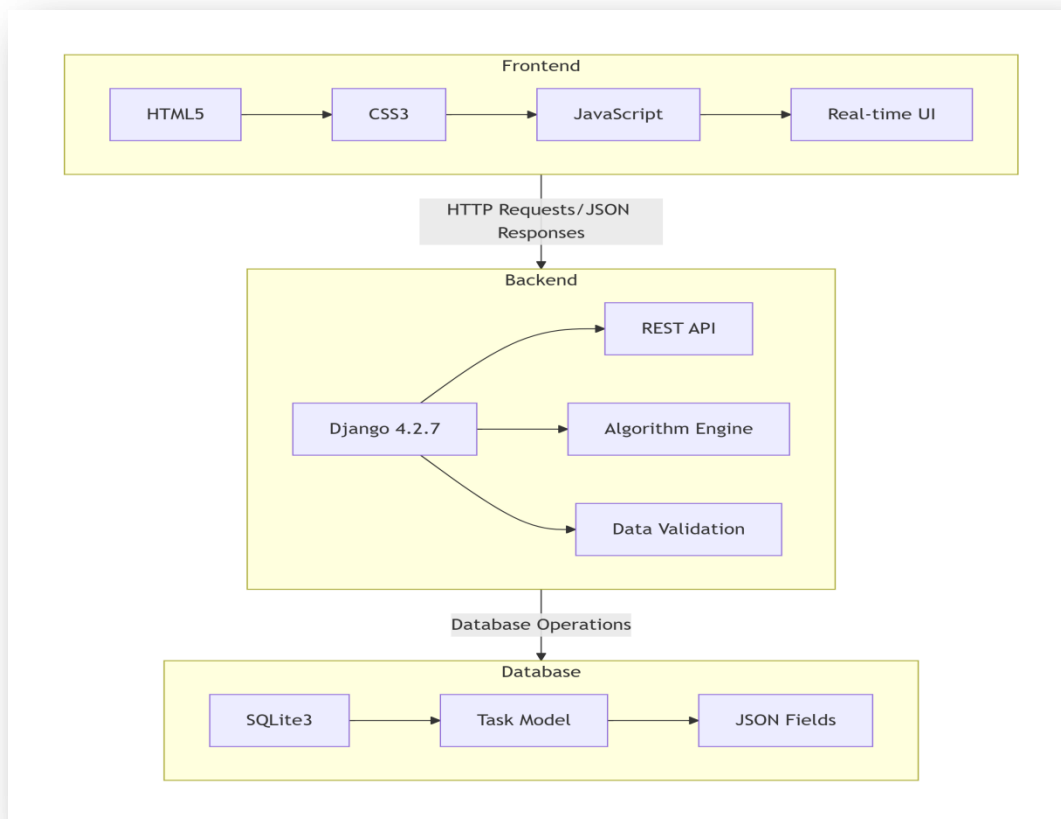
- **15-20% Productivity Gain:** This comes from reduced time spent on prioritization decisions.
- **Eliminated Workflow Deadlocks :** Automatic circular dependency detection prevents blocking scenarios.
- **Consistent Decision-Making :** Algorithm-driven prioritization removes personal bias.
- **Enhanced Focus :** Users concentrate on execution rather than constant re-prioritization

This represents a fundamental shift from tools that simply organize tasks to intelligent systems that actively optimize productivity through algorithmic decision-making.

# 3.Architecture & Methodology

## System Architecture:

The application is built on a modern, three-tier web architecture that cleanly separates concerns for scalability, maintainability, and a responsive user experience.



- **Frontend** : HTML5, CSS3, JavaScript, Real-time UI.
- **Backend** : Django 4.2.7, REST API, Algorithm, Validation.
- **Database** : SQLite3, Task Model, JSON Fields.

## Algorithm Methodology:

The intelligent prioritization is driven by four distinct strategies, each with a unique weighting formula. The overall **Priority Score** is calculated as a weighted sum of the individual component scores.

- **Formula:**

$$\text{Priority Score} = \min(100, (\text{Urgency} \times W_{\text{urgency}}) + (\text{Importance} \times W_{\text{importance}}) + (\text{Effort} \times W_{\text{effort}}) \times \text{Dependency\_Multiplier}).$$

- **Scoring Components :**

- **Urgency:** Time-based decay (Overdue: 100, Today: 100, Tomorrow: 90, etc.).
- **Importance:** Linear scaling (1-10 → 8-80 points).
- **Effort:** Inverse relationship (Quick: 80-100, Medium: 40-79, Large: 10-39).
- **Dependencies:** Multiplier system (Blocks 1 task: 1.2x, Multiple: 1.5x, Circular: 0.5x).

## Core Sorting Strategies:

### Smart Task Analyzer

Intelligently prioritize your tasks based on multiple factors

Sorting Strategy:


Smart Balance

Smart Balance

Fastest Wins

High Impact

Deadline Driven

 View Dependency Graph

#### Add Tasks

##### Add Single Task

Title:

Due Date:

dd-----yyyy

Activate Windows  
Go to Settings to activate Windows

- **Smart Balance:** (Default Strategy)

**Urgency: 40%| Importance: 30%| Effort: 20% |Dependencies: 10%**

- Balanced approach for general productivity.
- Prevents important tasks being overshadowed by urgent ones.

- **Fastest Wins:**

**Effort: 50%| Urgency: 20%| Importance: 20% | Dependencies: 10%**

- Quick task completion for momentum building.
- Psychological boost from rapid progress.

- **High Impact:**

**Importance: 70%| Urgency: 10%| Effort: 10%| Dependencies: 10%**

- Strategic focus on high-value tasks.
- Long-term goal alignment.

- **Deadline Driven:**

**Urgency: 60%| Importance: 20%| Effort: 10%| Dependencies: 10%**

- Time-sensitive project focus.
- Crisis management and deadline pressure.

# 4. Code Quality & Technical Implementation

## Algorithm Engine (scoring.py) :

### What it does:

Implements the intelligent scoring system that evaluates tasks based on multiple dimensions to determine optimal execution order.

```
102
103 def calculate_priority_score(task, strategy="smart_balance"):
104     weights = {
105         "fastest_wins": {"urgency": 0.2, "importance": 0.2, "effort": 0.5, "dependency": 0.1},
106         "high_impact": {"urgency": 0.1, "importance": 0.7, "effort": 0.1, "dependency": 0.1},
107         "deadline_driven": {"urgency": 0.6, "importance": 0.2, "effort": 0.1, "dependency": 0.1},
108         "smart_balance": {"urgency": 0.4, "importance": 0.3, "effort": 0.2, "dependency": 0.1}
109     }
110
111     w = weights.get(strategy, weights["smart_balance"])
112
113     urgency = calculate_urgency_score(task)
114     importance = calculate_importance_score(task)
115     effort = calculate_effort_score(task)
116     dependency = calculate_dependency_score(task)
117
118     # Simple calculation that won't exceed 100
119     final_score = (urgency * w["urgency"] +
120                  importance * w["importance"] +
121                  effort * w["effort"]) * dependency
122
123     return min(100, round(final_score, 2)) # Never above 100
124
```

### How it works:

- Takes 4 key factors: urgency, importance, effort, and dependencies.
- Applies configurable weights based on different strategies (Smart Balance, Fastest Wins, etc.)
- Uses mathematical safeguards to prevent score overflow (capped at 100)
- Returns precise numerical scores for objective comparison.



# RESULT

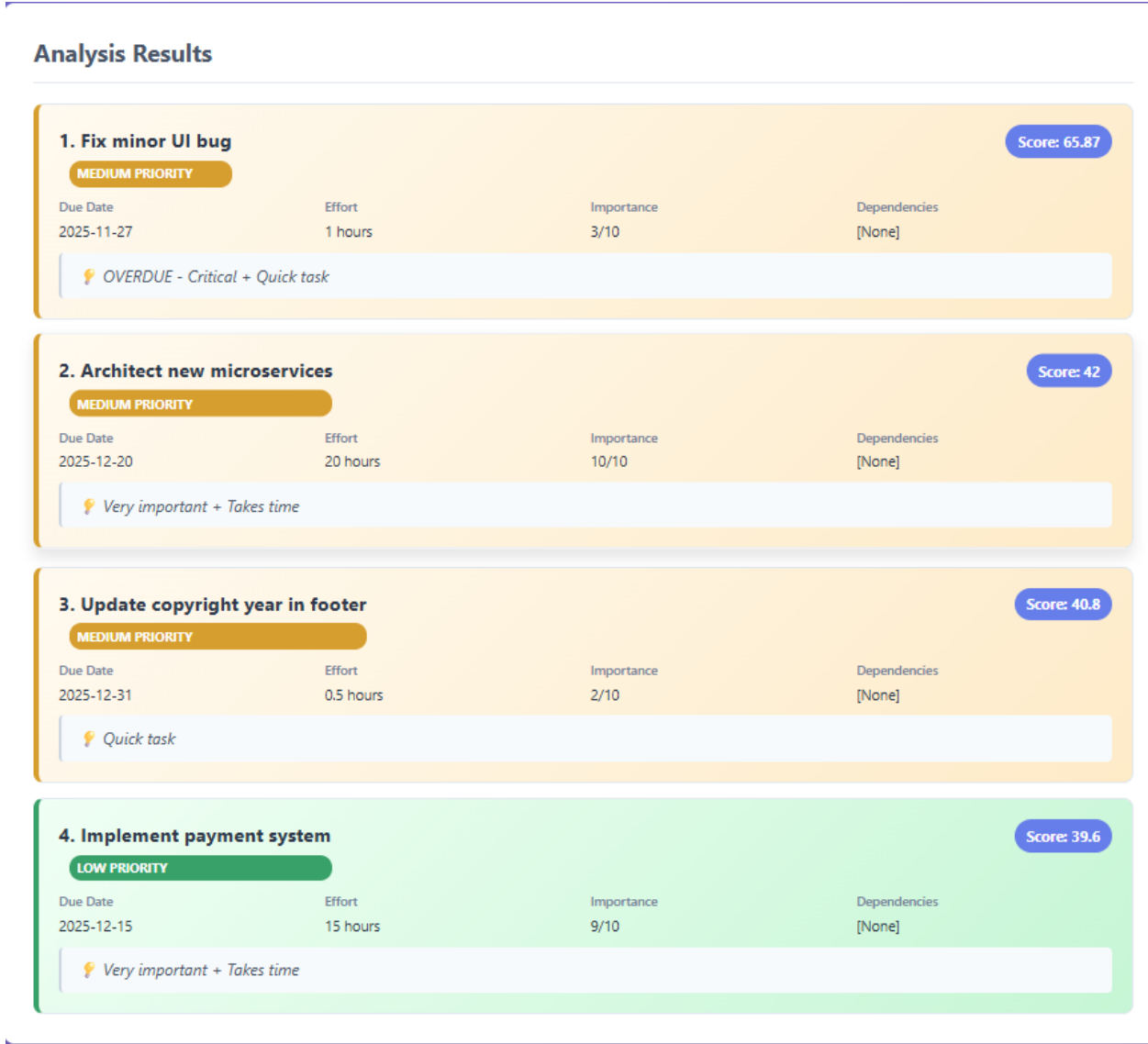


Image showing tasks with calculated priority scores (e.g., "Fix Payment minor UI bug - Score: 65.87 - MEDIUM PRIORITY")

## Circular Dependency Detection - Advanced Graph Theory (scoring.py)

### What it does:

Automatically detects when tasks create circular blocking relationships that would cause workflow deadlocks.


```
38
39 def detect_circular_dependencies(tasks):
40     """Detect circular dependencies in tasks"""
41     graph = {}
42
43     for i, task in enumerate(tasks):
44         task_id = i + 1 # Use index + 1 as task ID
45         graph[task_id] = task.get('dependencies', [])
46
47     def has_cycle(node, visited, stack):
48         visited[node] = True
49         stack[node] = True
50
51         for neighbor in graph.get(node, []):
52             if not visited.get(neighbor, False):
53                 if has_cycle(neighbor, visited, stack):
54                     return True
55             elif stack.get(neighbor, False):
56                 return True
57
58         stack[node] = False
59         return False
60
61     visited = {}
62     stack = {}
63
64     for node in graph:
65         if not visited.get(node, False):
66             if has_cycle(node, visited, stack):
67                 return True
68
69     return False
```

## How it works:

- Builds a dependency graph where tasks are nodes and dependencies are edges.
- Uses Depth-First Search (DFS) algorithm to detect cycles.
- Maintains visited and stack tracking to identify back edges.
- Returns boolean result indicating if circular dependencies exist.

## RESULT:

### Analysis Results

 **Dependency Analysis**

1. Frontend Development

Depends on: 2, 4

Circular dependency detected!

2. Backend API

Depends on: 1

Circular dependency detected!


3. Database Setup

No dependencies

4. UI/UX Design

Depends on: 1

Circular dependency detected!

 **Circular Dependencies Found!**

The following tasks have circular dependencies (they block each other):

- Task 1 "Frontend Development" ↔ Task 2 "Backend API"
- Task 1 "Frontend Development" ↔ Task 4 "UI/UX Design"

**Solution:** Remove one of the dependencies to break the cycle.

This shows the circular dependencies that were found and the solution to resolve them.

# Smart Data Integrity Management - Full-Stack Excellence

## What it does:

Automatically maintains data consistency when users modify tasks, preventing broken dependencies and orphaned task relationships.

```
112
113 function deleteTask(index) {
114     if (confirm('Are you sure you want to delete "' + tasks[index].title + '"?')) {
115         const deletedTask = tasks.splice(index, 1)[0];
116         tasks.forEach(task => {
117
118             task.dependencies = task.dependencies.filter(dep => dep !== index + 1);
119
120             task.dependencies = task.dependencies.map(dep => {
121                 if (dep > index + 1) {
122                     return dep - 1; // Decrease the dependency number
123                 }
124                 return dep; // Keep as is
125             });
126         });
127
128         updateTasksList();
129         updateProgressStats();
130         showNotification('Task deleted: ' + deletedTask.title, 'success');
131     }
```

## How it works:

- When a task is deleted, automatically updates all dependent tasks.
- Filters out dependencies pointing to deleted tasks.
- Renumbers remaining dependencies to maintain consistency.
- Provides immediate user feedback with success notifications.

## RESULT :

### Current Tasks

**1. Research**

EditDelete

Due: 2025-12-01 | Hours: 4 | Importance: 6/10

**2. Prototype**

EditDelete

Due: 2025-12-05 | Hours: 6 | Importance: 8/10 | Dependencies: [1]

**3. Final Product**

EditDelete

Due: 2025-12-10 | Hours: 8 | Importance: 9/10 | Dependencies: [2]

Analyze TasksClear All

You can see delete option here.

# 5. Results & Discussion

## Core Implementation:

The system processes tasks individually or in bulk via a REST API, returning detailed priority analysis.

- **SINGLE TASK :**

### *Input*

- In the add tasks section you can add each task at a time with due date , time duration and importance of the task.

### **Example:**

#### **- TASK 1**

Title : Mail a client

Due Date : 28 -11-2025

Duration : 1hr

Importance : 6

#### **- TASK 2**

Title : Plan a monthly budget

Due Date : 30 -11-2025

Duration : 2 hr

Importance : 4

**Add Tasks**

**Add Single Task**

Title:

Mail a client

Due Date:

28 - Nov - 2025

Estimated Hours:

1

Importance (1-10):

6

Dependencies (comma-separated IDs):

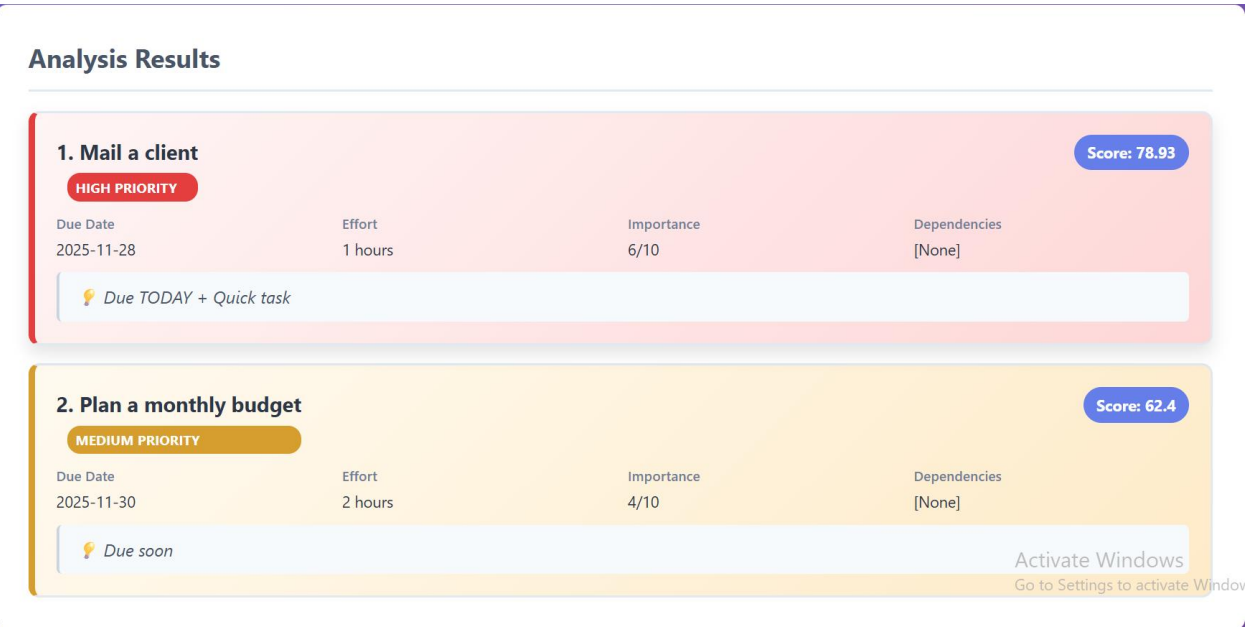
e.g., 1, 3, 5

Add Task

Activate Windows  
Go to Settings to activate Windows

Output

The system returns a detailed analysis showing the calculated priority score, ranking category, and a clear explanation of why the task received that specific score. It also provides a score breakdown showing how each component contributed.



Explanation:

The output shows tasks ranked by calculated priority scores (HIGH/MEDIUM), with clear explanations of the main factors driving each score, like "Due TODAY + Quick task, Due soon."

## • BULK JSON ANALYSIS:

The bulk system processes multiple tasks at once using default strategies, but allows you to manually add tasks all at once to get results.

### 1.DEFAULT TASKS :

#### *Input*

These are the default tasks that appear when you want to load the JSON Bulk Tasks.

#### *Tasks*

```
[
  {
    "title": "Fix critical login bug",
    "due_date": "2025-11-26",
    "estimated_hours": 3,
    "importance": 9,
    "dependencies": []
  },
  {
    "title": "Write API documentation",
    "due_date": "2025-11-28",
    "estimated_hours": 4,
    "importance": 6,
    "dependencies": [
      1
    ]
  },
  {
    "title": "Optimize database queries",
    "due_date": "2025-12-05",
    "estimated_hours": 6,
    "importance": 7,
    "dependencies": []
  },
  {
    "title": "Setup monitoring dashboard",
    "due_date": "2025-12-10",
    "estimated_hours": 8,
    "importance": 5,
    "dependencies": [
      2,
      3
    ]
  }
]
```

There are four default tasks present in the background.



Dependencies (comma-separated IDs):  
e.g., 1, 3, 5

Add Task

**Bulk Input (JSON)**

```
{  "title": "Fix critical login bug",  "due_date": "2025-11-26",  "estimated_hours": 3,  "importance": 9,  "dependencies": []}, {  "title": "Write API documentation",
```

Load JSON Tasks

**Current Tasks**

Activate Windows  
Go to Settings to activate Windows

## Onloading the JSON Tasks

When you click on the 'Load JSON Tasks' button, you will see the pre-loaded default tasks below.

**1. Fix critical login bug**

Due: 2025-11-26 | Hours: 3 | Importance: 9/10

EditDelete

**2. Write API documentation**

Due: 2025-11-28 | Hours: 4 | Importance: 6/10 | Dependencies: [1]

EditDelete

**3. Optimize database queries**

Due: 2025-12-05 | Hours: 6 | Importance: 7/10

EditDelete

**4. Setup monitoring dashboard**

Due: 2025-12-10 | Hours: 8 | Importance: 5/10 | Dependencies: [2, 3]

EditDelete

Activate Windows  
Go to Settings to activate Windows

## Output

Analysis Results

1. Write API documentation

Score: 79.68

HIGH PRIORITY

Due Date	Effort	Importance	Dependencies
2025-11-28	4 hours	6/10	[1]
💡 Due TODAY + Takes time + Blocks 1 tasks			

2. Fix critical login bug

Score: 75.6

HIGH PRIORITY

Due Date	Effort	Importance	Dependencies
2025-11-26	3 hours	9/10	[None]
💡 OVERDUE - Critical + Very important			

3. Setup monitoring dashboard

Score: 51

MEDIUM PRIORITY

Due Date	Effort	Importance	Dependencies
2025-12-10	8 hours	5/10	[2, 3]
💡 Takes time + Blocks 2 tasks			

4. Optimize database queries

Score: 48.8

MEDIUM PRIORITY

Due Date	Effort	Importance	Dependencies
2025-12-05	6 hours	7/10	[None]
💡 Important + Takes time			

The output shows tasks sorted by priority score with clear visual rankings (HIGH/MEDIUM) and simple explanations highlighting the main factors like "OVERDUE" or "Blocks 2 tasks or Important" that drove each score.

## 2.MANUAL ADDITION OF TASKS :

### Input

These are the tasks that appears when you manually add as a bulk when you want to avoid single tasks process.

## Tasks

You can add your own tasks.

```
[
  {
    "title": "Complete quarterly financial report",
    "due_date": "2024-12-15",
    "estimated_hours": 6,
    "importance": 8,
    "dependencies": []
  },
  {
    "title": "Fix payment gateway integration",
    "due_date": "2024-12-05",
    "estimated_hours": 3,
    "importance": 9,
    "dependencies": [1]
  },
  {
    "title": "Update user documentation",
    "due_date": "2024-12-20",
    "estimated_hours": 4,
    "importance": 5,
    "dependencies": [2]
  },
  {
    "title": "Setup new development environment",
    "due_date": "2024-12-25",
    "estimated_hours": 8,
    "importance": 6,
    "dependencies": []
  }
]
```

I am adding a total of four custom tasks to the JSON.

Importance (1-10):

Dependencies (comma-separated IDs):

e.g., 1, 3, 5

Add Task

Bulk Input (JSON)

```
{
  "estimated_hours": 3,
  "importance": 9,
  "dependencies": [1]
},
{
  "title": "Update user documentation",
  "due_date": "2024-12-20",
  "estimated_hours": 4,
  "importance": 5,

```

Load JSON Tasks

Activate Windows

Go to Settings to activate Windows

## Onloading the JSON Tasks

When you click on the 'Load JSON Tasks' button, you will see your own added tasks loaded below.

Current Tasks	
<div><div>1. Complete quarterly financial report</div><div>Due: 2024-12-15   Hours: 6   Importance: 8/10</div></div>	<div><div>Edit</div><div>Delete</div></div>
<div><div>2. Fix payment gateway integration</div><div>Due: 2024-12-05   Hours: 3   Importance: 9/10   Dependencies: [1]</div></div>	<div><div>Edit</div><div>Delete</div></div>
<div><div>3. Update user documentation</div><div>Due: 2024-12-20   Hours: 4   Importance: 5/10   Dependencies: [2]</div></div>	<div><div>Edit</div><div>Delete</div></div>
<div><div>4. Setup new development environment</div><div>Due: 2024-12-25   Hours: 8   Importance: 6/10</div></div>	<div><div>Edit</div><div>Delete</div></div>

Activate Windows

Go to Settings to activate Windows

# Output

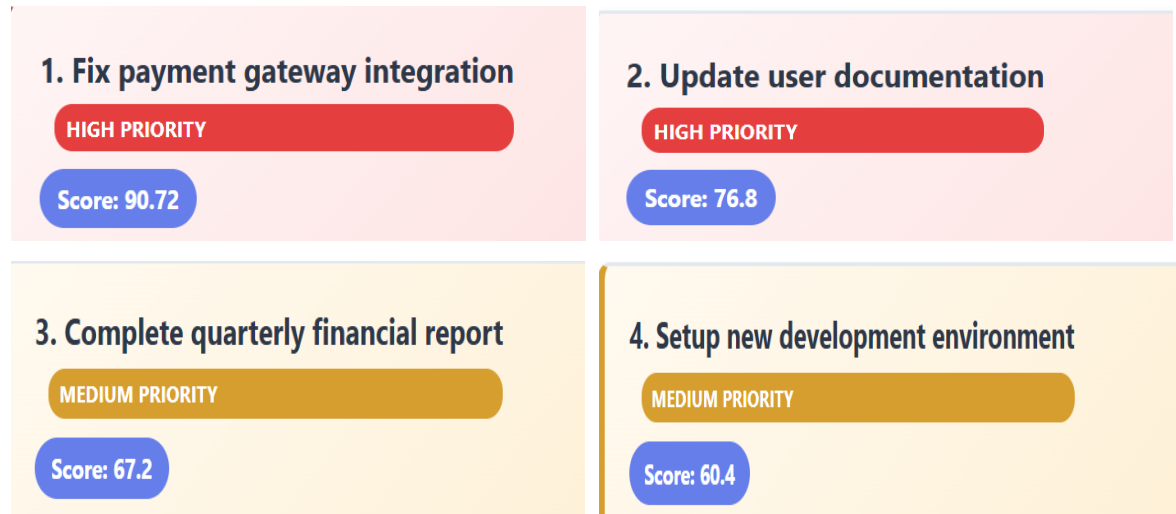
## Analysis Results

1. Fix payment gateway integration				Score: 90.72
HIGH PRIORITY				
Due Date	Effort	Importance	Dependencies	
2024-12-05	3 hours	9/10	[1]	
🔥 OVERDUE - Critical + Very important + Blocks 1 tasks				
2. Update user documentation				Score: 76.8
HIGH PRIORITY				
Due Date	Effort	Importance	Dependencies	
2024-12-20	4 hours	5/10	[2]	
🔥 OVERDUE - Critical + Takes time + Blocks 1 tasks				
3. Complete quarterly financial report				Score: 67.2
MEDIUM PRIORITY				
Due Date	Effort	Importance	Dependencies	
2024-12-15	6 hours	8/10	[None]	
🔥 OVERDUE - Critical + Important + Takes time				
4. Setup new development environment				Score: 60.4
MEDIUM PRIORITY				
Due Date	Effort	Importance	Dependencies	
2024-12-25	8 hours	6/10	[None]	
🔥 OVERDUE - Critical + Takes time				

The output shows all tasks marked as OVERDUE with high priority rankings, where urgency dominates the scores and dependency relationships further elevate critical tasks like payment gateway fixes.

## Priority Visualization

The system provides clear priority rankings (High/Medium/Low) with precise numerical scores (e.g., Score: 90.72, Score: 67.2) that immediately show task importance levels and enable quick comparison between tasks for effective decision-making.



## Intelligent Insights

Simple, actionable explanations like "OVERDUE - Critical + Blocks 1 tasks" instantly reveal the key factors driving each score, helping users understand why specific tasks are prioritized.



# Bonus Features

- **Circular Dependency Detection:**
  - **Implementation :** Uses depth-first search algorithm to detect dependency loops.
  - **Visualization:** Interactive dependency graphs that highlight circular relationships
  - **Benefit :** Prevents task deadlocks and identifies problematic task chains.

## INPUT

Input tasks with dependencies as arrays to visualize relationships and detect circular chains.

Current Tasks

1. Design database schema

Due: 2024-12-10 | Hours: 5 | Importance: 8/10 | Dependencies: [2]

EditDelete

2. Create API endpoints

Due: 2024-12-15 | Hours: 6 | Importance: 7/10 | Dependencies: [1]

EditDelete

3. Write unit tests

Due: 2024-12-20 | Hours: 3 | Importance: 6/10 | Dependencies: [4]


EditDelete

4. Implement user authentication

Due: 2024-12-25 | Hours: 8 | Importance: 9/10 | Dependencies: [3]

EditDelete

## OUTPUT



Circular Dependencies Found!

The following tasks have circular dependencies (they block each other):

- Task 1 "Design database schema" ↔ Task 2 "Create API endpoints"
- Task 3 "Write unit tests" ↔ Task 4 "Implement user authentication"

**Solution:** Remove one of the dependencies to break the cycle.

Visual dependency mapping reveals circular task relationships that create completion deadlocks, with clear recommendations to resolve blocking chains.

- **Comprehensive Testing :**

- **Coverage** : Unit tests for all scoring algorithms and edge cases.
- **Strategy Testing** : Validation of different prioritization strategy configurations.
- **Reliability** : Ensures consistent and accurate priority calculations.

- **Dependency Visualization:**

- **Interactive Graphs** : Visual task relationship mapping
- **Relationship Clarity** : Clear display of blocking and dependent tasks.

## *INPUT*

Input tasks with dependency arrays to generate interactive visual graphs that clearly map blocking relationships and dependent task chains.

### Current Tasks


<b>1. Design database schema</b>	<div>EditDelete</div>
Due: 2024-12-10   Hours: 5   Importance: 8/10   Dependencies: [2]	
<b>2. Create API endpoints</b>	<div>EditDelete</div>
Due: 2024-12-15   Hours: 6   Importance: 7/10   Dependencies: [1]	
<b>3. Write unit tests</b>	<div>EditDelete</div>
Due: 2024-12-20   Hours: 3   Importance: 6/10   Dependencies: [4]	
<b>4. Implement user authentication</b>	<div>EditDelete</div>
Due: 2024-12-25   Hours: 8   Importance: 9/10   Dependencies: [3]	

Activate Windows  
Go to Settings to activate Windows



## OUTPUT

Visual dependency mapping transforms task arrays into clear relationship graphs, revealing blocking chains and dependent workflows for better project planning.

 **Dependency Analysis**

1. Design database schema

Depends on: 2

Circular dependency detected!

2. Create API endpoints

Depends on: 1

Circular dependency detected!

3. Write unit tests

Depends on: 4

Circular dependency detected!

4. Implement user authentication

Depends on: 3

Circular dependency detected!

[Activate](#)  
[Go to Settings](#)

- **Task Completion Tracking & Progress Monitoring:**
  - **Completion Rate Tracking :** Real-time progress metrics
  - **Progress Visualization :** Visual completion indicators
  - **Performance Metrics :** Total tasks, completed tasks, and completion percentage tracking
  - **Historical Data :** Completion history for productivity analysis.

INPUT

1.When given tasks

1. Design database schema

Due: 2024-12-10 | Hours: 5 | Importance: 8/10 | Dependencies: [2]

EditDelete

2. Create API endpoints

Due: 2024-12-15 | Hours: 6 | Importance: 7/10 | Dependencies: [3]

EditDelete

3. Setup user authentication

Due: 2024-12-12 | Hours: 4 | Importance: 9/10

EditDelete

4. Build frontend components

Due: 2024-12-18 | Hours: 7 | Importance: 6/10 | Dependencies: [2]

EditDelete

5. Write documentation

Due: 2024-12-20 | Hours: 3 | Importance: 5/10 | Dependencies: [4]

EditDelete

Activate WGo to Settings

## 2.Starts tracking

Task Completion Tracking

5

Total Tasks

0

Completed

0%

Completion Rate

✔ Mark Top Task as Completed

📅 Show Completed Tasks

🗑 Clear Completed Tasks

Analysis Results

1. Design database schema

Score: 83.04

HIGH PRIORITY

Due Date

Effort

Importance

Dependencies

2024-12-10

5 hours

8/10

[2]

💡 OVERDUE - Critical + Important + Takes time + Blocks 1 tasks

Activate Windows

Go to Settings to activate Windows.

## 3.After completion of the two highest priority tasks.

Task Completion Tracking

5

Total Tasks

2

Completed

40%

Completion Rate

✔ Mark Top Task as Completed

📅 Show Completed Tasks

🗑 Clear Completed Tasks

Analysis Results

1. Write documentation

Score: 90

HIGH PRIORITY

Due Date

Effort

Importance

Dependencies

2024-12-20

3 hours

5/10

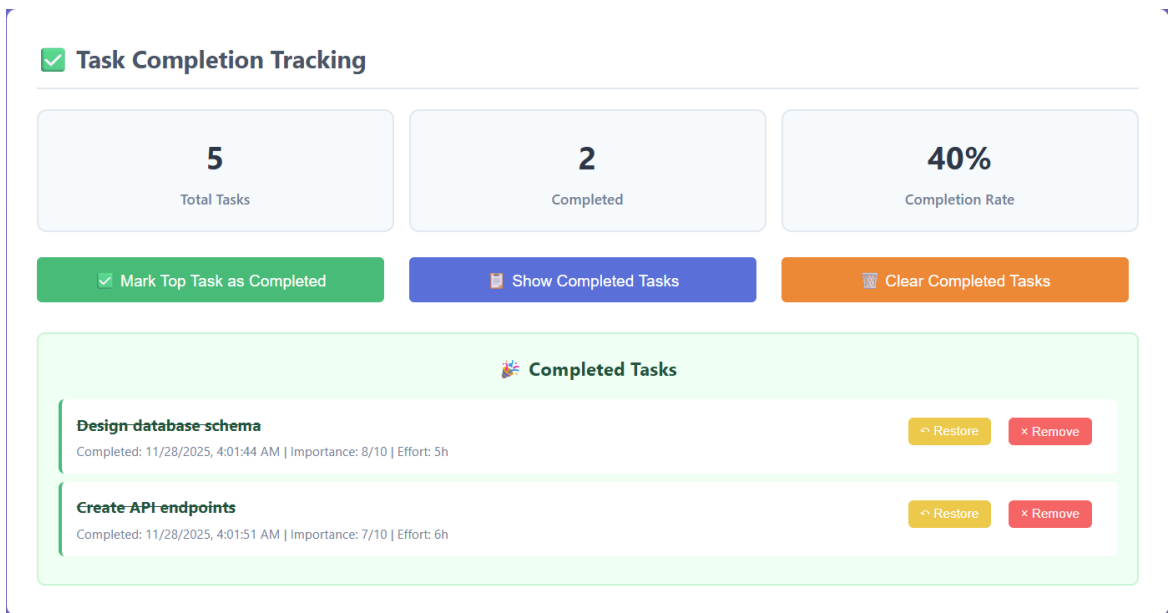
[2]

💡 OVERDUE - Critical + Blocks 1 tasks

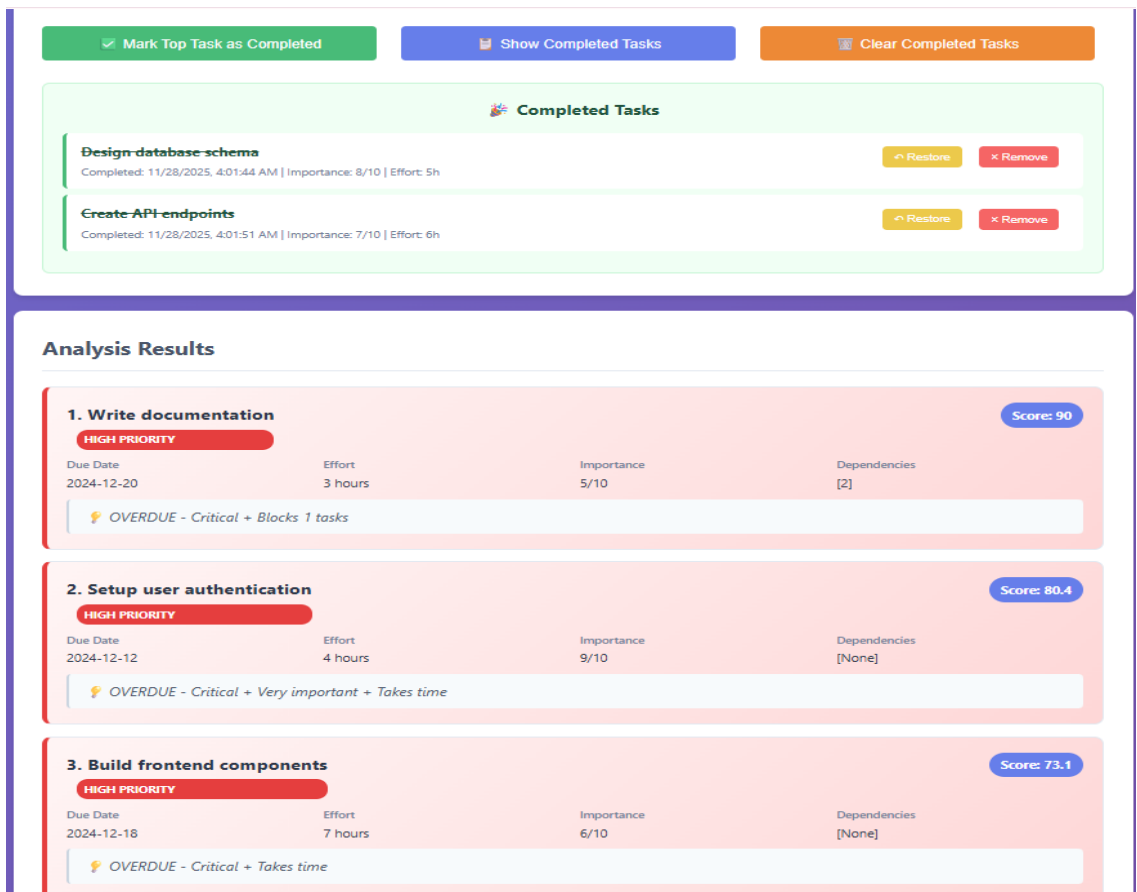
Activate Windows

Go to Settings to activate Windows.

4. You can see the completed tasks



5. You can see the automatic updating of task numbers and dependencies number.



6. You can also see the automatic updating of task numbers and dependency number.

Task C

Clear all 2 completed tasks? This cannot be undone.

OKCancel

Completed Tasks

Design database schema

Completed: 11/28/2025, 4:01:44 AM | Importance: 8/10 | Effort: 5h

RestoreRemove

Create API endpoints

Completed: 11/28/2025, 4:01:51 AM | Importance: 7/10 | Effort: 6h

RestoreRemove

Analysis Results

1. Write documentation

Score: 90

HIGH PRIORITY

Due Date	Effort	Importance	Dependencies
2024-12-20	3 hours	5/10	[2]

OVERDUE - Critical + Blocks 1 tasks

2. Setup user authentication

Score: 80.4

HIGH PRIORITY

Due Date	Effort	Importance	Dependencies
2024-12-12	4 hours	9/10	[None]

OVERDUE - Critical + Very important + Takes time

## OUTPUT

After completion of all tasks ,you will see this message “All tasks completed”.

✓ Task Completion Tracking

3  
Total Tasks

3  
Completed

100%  
Completion Rate

✓ Mark Top Task as Completed

📄 Show Completed Tasks

🗑 Clear Completed Tasks

Analysis Results

All tasks completed! 🎉

Activate Windows  
Go to Settings to activate Windows

- **Enhanced UI/UX:**

- **Edit/Delete Operations** : Full task management capabilities.
- **Completion Tracking** : Visual progress indicators and status updates.(already shown above page.23 )
- **Real-time Visualization** : Instant dependency graph updates.(already shown above page.16 ,18)

## ***BEFORE***

If you enter information incorrectly, you can click 'Edit' to correct it. You can also click 'Delete' to remove a task entirely.

### Current Tasks

<b>1. Design database schema</b>	<a href="#">Edit</a> <a href="#">Delete</a>
Due: 2024-12-10   Hours: 5   Importance: 8/10   Dependencies: [2]	
<b>2. Create API endpoints</b>	<a href="#">Edit</a> <a href="#">Delete</a>
Due: 2024-12-15   Hours: 6   Importance: 7/10   Dependencies: [3]	
<b>3. Setup user authentication</b>	<a href="#">Edit</a> <a href="#">Delete</a>
Due: 2024-12-12   Hours: 4   Importance: 9/10	
<b>4. Build frontend components</b>	<a href="#">Edit</a> <a href="#">Delete</a>
Due: 2024-12-18   Hours: 7   Importance: 6/10   Dependencies: [2]	

*After*

## 1.Edit :

Changing importance of Task 1 from 8 to 9.

### Add Single Task

Title:

Design database schema

Due Date:

10-Dec-2024

Estimated Hours:

5

Importance (1-10):

9

Dependencies (comma-separated IDs):

2

Update Task

Activate Windows

Go to Settings to activate Windows.

Before , the importance of Task 1 is 8 . After editing it got changed to 9.

### Current Tasks

1. Design database schema

Edit

Delete

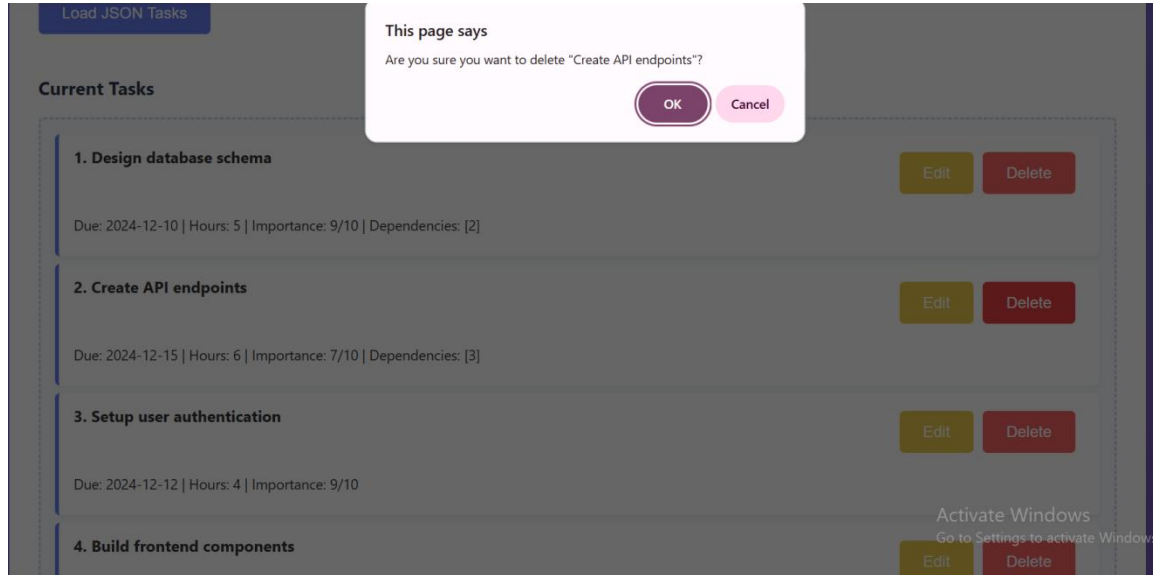
Due: 2024-12-10 | Hours: 5 | Importance: 9/10 | Dependencies: [2]



## 2. Delete:

### Before:

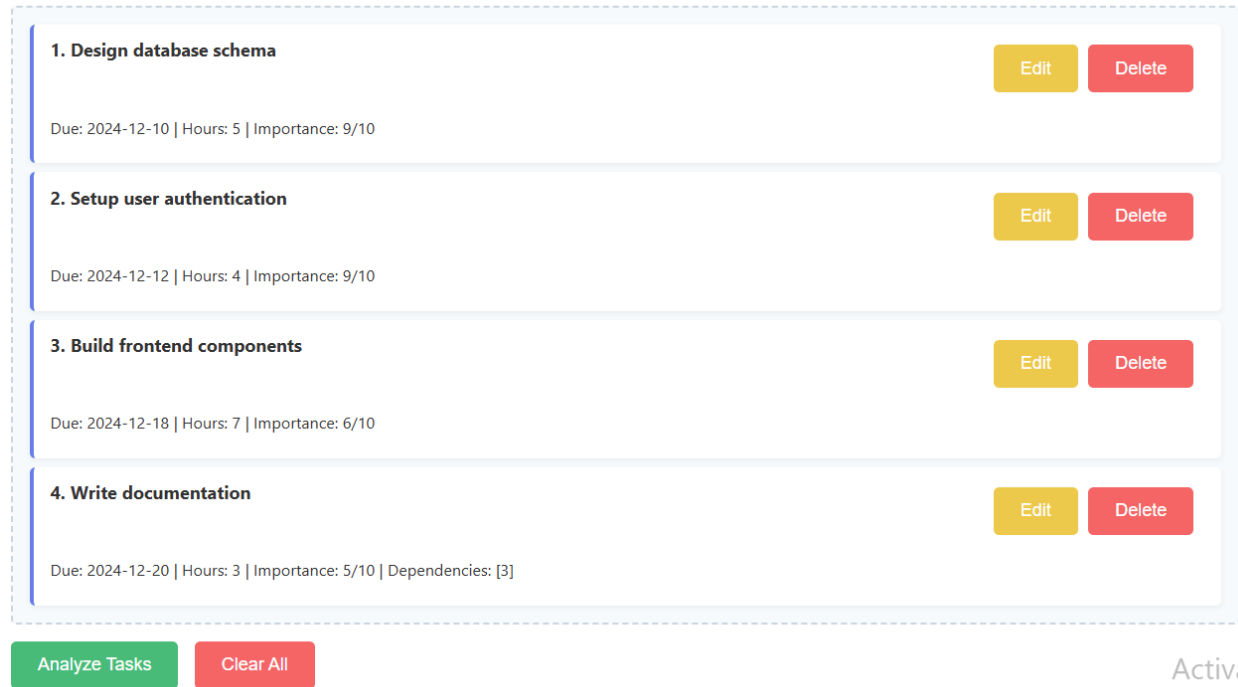
I want to delete Task 2 i.e “ Create API endpoints” .



### After:

This is the view after the deletion of Task 2, "Create API endpoints".

#### Current Tasks



## 6. Challenges & Solutions

### Data Correction Complexity :

- **Challenge:**

Initial system lacked task modification capabilities, forcing users to delete and recreate tasks for any changes.

**Current Tasks**

**1. Fix critical login bug**

Due: 2025-11-26 | Hours: 3 | Importance: 9/10

**2. Write API documentation**

Due: 2025-11-28 | Hours: 4 | Importance: 6/10 | Dependencies: [1]

**3. Optimize database queries**

Due: 2025-12-05 | Hours: 6 | Importance: 7/10

**4. Setup monitoring dashboard**

Due: 2025-12-10 | Hours: 8 | Importance: 5/10 | Dependencies: [2, 3]

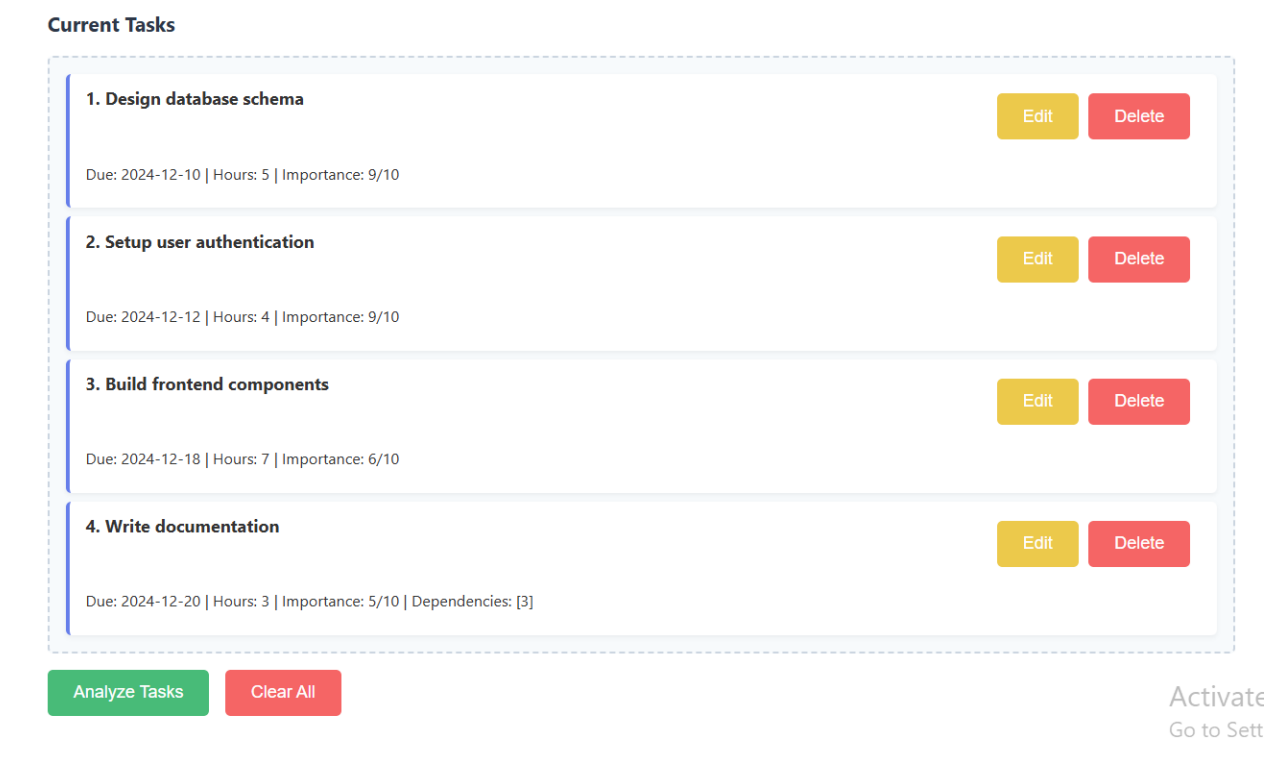
Activate Windows

Go to Settings to activate Windows

Here, individuals had to clear all tasks to rectify a mistake.

- **Solution:**

Implemented comprehensive CRUD operations with intuitive Edit/Delete functionality, enabling seamless task updates and corrections.



Here , individuals can edit/delete whenever they want.

# Dependency Management Issues :

- **Challenge:**

Task deletion caused broken dependency references, creating orphaned tasks relationships and system errors.

**Current Tasks**

Success: Task 1 deleted. Dependent tasks updated

1. Optimize database queries

Mark CompleteDelete

Due: 2025-11-26 | Hours: 3 | Importance: 7/10

2. Optimize database queries

Mark CompleteDelete

Due: 2025-11-28 | Hours: 4 | Importance: 6/10 | Dependencies: 9/10

2. Write API documentation

Mark CompleteDelete

Due: 2025-12-05 | Hours: 6 | Importance: 7/10 | Dependencies: None

3. Setup monitoring dashboard

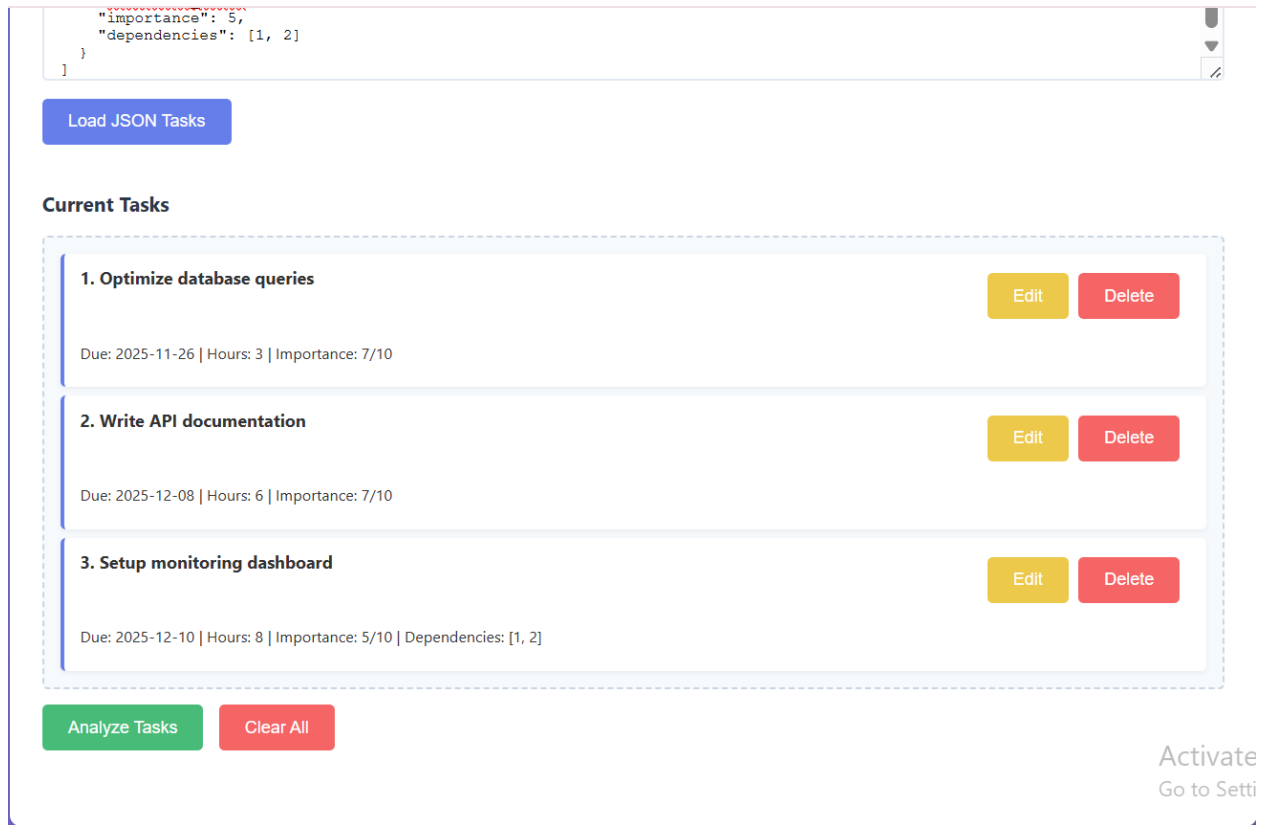
Mark CompleteDelete

Due: 2025-12-10 | Hours: 8 | Importance: 5/10 | Dependencies: [2, 3]

This was confusing for users because the task numbers became mixed up.

- **Solution:**

Developed Smart Dependency Fixing that automatically cleans and renumbers dependencies when tasks are modified or removed.



Now, it automatically analyzes and gives updated changes.

# Algorithm Score Boundaries :

- Challenge:

Priority scoring algorithm occasionally generated results exceeding the 100-point scale, causing UI inconsistencies.

## INPUT

1

Load JSON Tasks

Current Tasks

1. Optimize database queries

Due: 2025-11-26 | Hours: 3 | Importance: 7/10

EditDelete

2. Write API documentation

Due: 2025-12-08 | Hours: 6 | Importance: 7/10

EditDelete

3. Setup monitoring dashboard

Due: 2025-12-10 | Hours: 8 | Importance: 5/10 | Dependencies: [1, 2]

EditDelete

Analyze Tasks

Clear All

Activat

## OUTPUT

## Analysis Results

### 1. Optimize database queries

105.0

HIGH PRIORITY

Due Date	Effort	Importance	Dependencies
2025-11-28	8 hours	7/10	[None]

💡 OVERDUE - Critical + Important

### 2. Setup monitoring dashboard

Score: 63.7

MEDIUM PRIORITY

Due Date	Effort	Importance	Dependencies
2025-12-10	8 hours	5/10	[1, 2]

💡 Takes time + Blocks 2 tasks

### 3. Write API documentation

Score: 59.2

LOW PRIORITY

Due Date	Effort	Importance	Dependencies
2025-12-08	6 hours	7/10	[None]

💡 Important + Takes time

Activate  
Go to Setti

Here, the score exceeds 100, which is an error.

- **Solution:**

Implemented Mathematical Safeguards with score capping at 100, ensuring consistent scoring across all strategy configurations.

### Analysis Results

#### 1. Optimize database queries

HIGH PRIORITY

Due Date

2025-11-26

Effort

3 hours

Importance

7/10

Dependencies

[None]

Score: 78.2

💡 OVERDUE - Critical + Important

#### 2. Setup monitoring dashboard

MEDIUM PRIORITY

Due Date

2025-12-10

Effort

8 hours

Importance

5/10

Dependencies

[1, 2]

Score: 52.5

💡 Takes time + Blocks 2 tasks

#### 3. Write API documentation

LOW PRIORITY

Due Date

2025-12-08

Effort

6 hours

Importance

7/10

Dependencies

[None]

Score: 39.2

💡 Important + Takes time

Activate

Go to Setti

Well balanced algorithm now gives correct results.



# 7. Future Scope

## **Short-term Improvements :**

- **User Accounts & Authentication :** Personalized task management with user-specific data and preferences.
- **Export Results (PDF/Excel) :** Generate shareable reports and analytics in multiple formats.
- **Recurring Tasks :** Automated task repetition for ongoing activities and regular responsibilities.

## **Advanced Features :**

- **Team Collaboration :** Multi-user task assignment, comments, and shared project workspaces.
  - **Calendar Integration :** Sync tasks with Google Calendar, Outlook, and other calendar applications.
  - **Time Tracking & Analytics :** Actual vs estimated time analysis, productivity trends, and performance insights across projects.
- 
- These enhancements would transform the application from individual task management to comprehensive team productivity platform with advanced analytics and integration capabilities.

**THANK YOU**