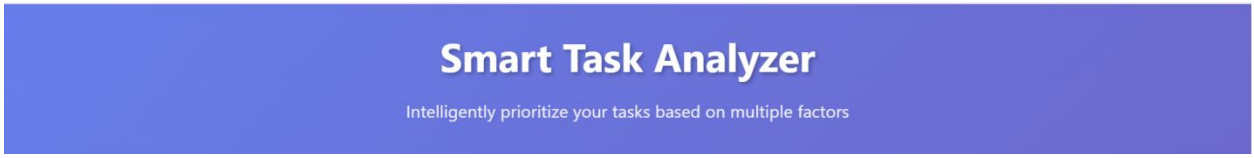
SMART TASK ANALYZER



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:

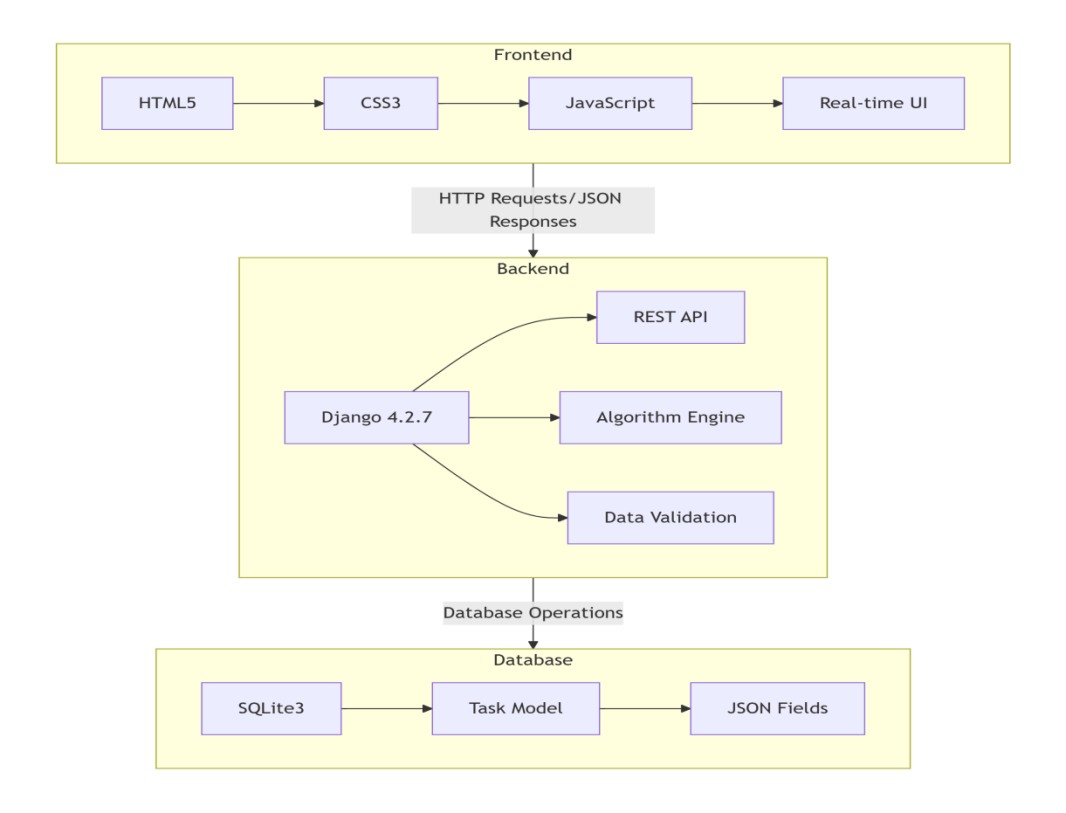
- **15-20% Productivity Gain:** This comes from reduced time spent on prioritization decisions.
- **Eliminated Workflow Deadlocks :** Automatic circular dependency detection prevents blocking scenarios.
- **Consistent Decision-Making :** Algorithm-driven prioritization removes personal bias.
- **Enhanced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

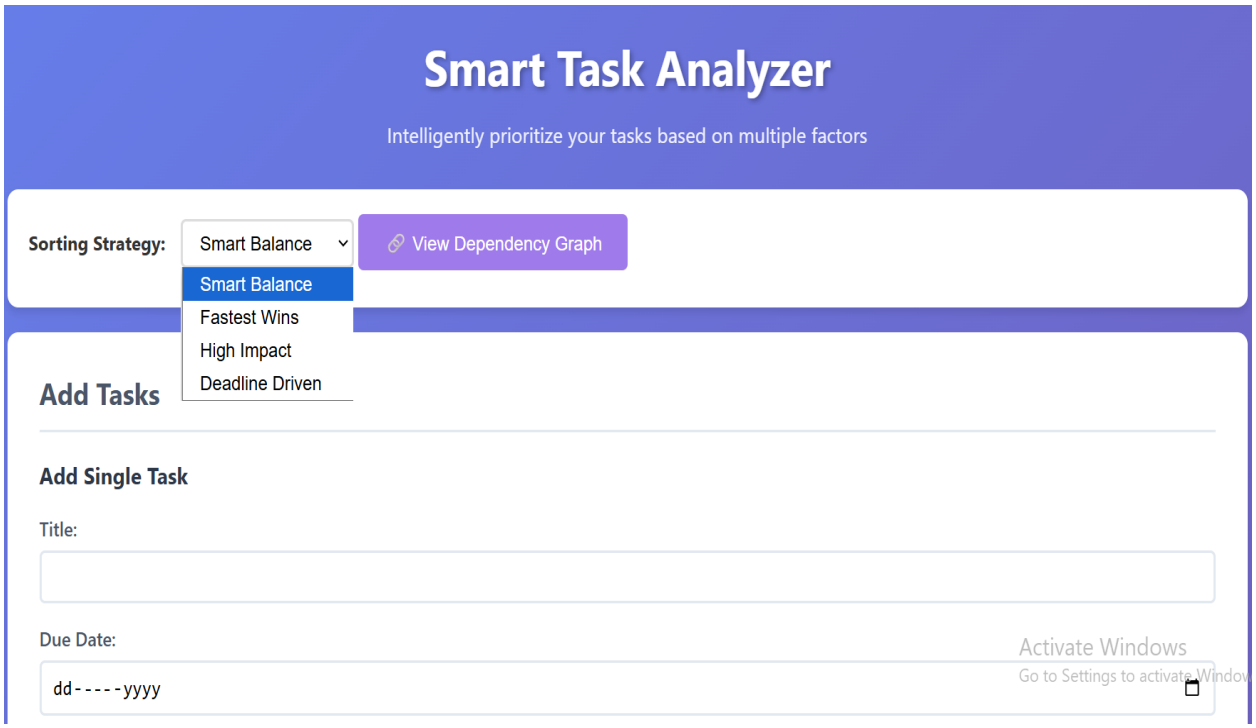
- **Formula:**

Priority Score = min (100 , (Urgency × W_urgency) + (Importance ×W_importance)+(Effort × W_effort) × 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 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.

```python

def calculate_priority_score(task, strategy="smart_balance"):
    weights = {
        "fastest_wins": {"urgency": 0.2, "importance": 0.2, "effort": 0.5, "dependency": 0.1},
        "high_impact": {"urgency": 0.1, "importance": 0.7, "effort": 0.1, "dependency": 0.1},
        "deadline_driven": {"urgency": 0.6, "importance": 0.2, "effort": 0.1, "dependency": 0.1},
        "smart_balance": {"urgency": 0.4, "importance": 0.3, "effort": 0.2, "dependency": 0.1}
    }

    w = weights.get(strategy, weights["smart_balance"])

    urgency = calculate_urgency_score(task)
    importance = calculate_importance_score(task)
    effort = calculate_effort_score(task)
    dependency = calculate_dependency_score(task)

    # Simple calculation that won't exceed 100
    final_score = (urgency * w["urgency"] +
                   importance * w["importance"] +
                   effort * w["effort"]) * dependency

    return min(100, round(final_score, 2))  # Never above 100
```

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

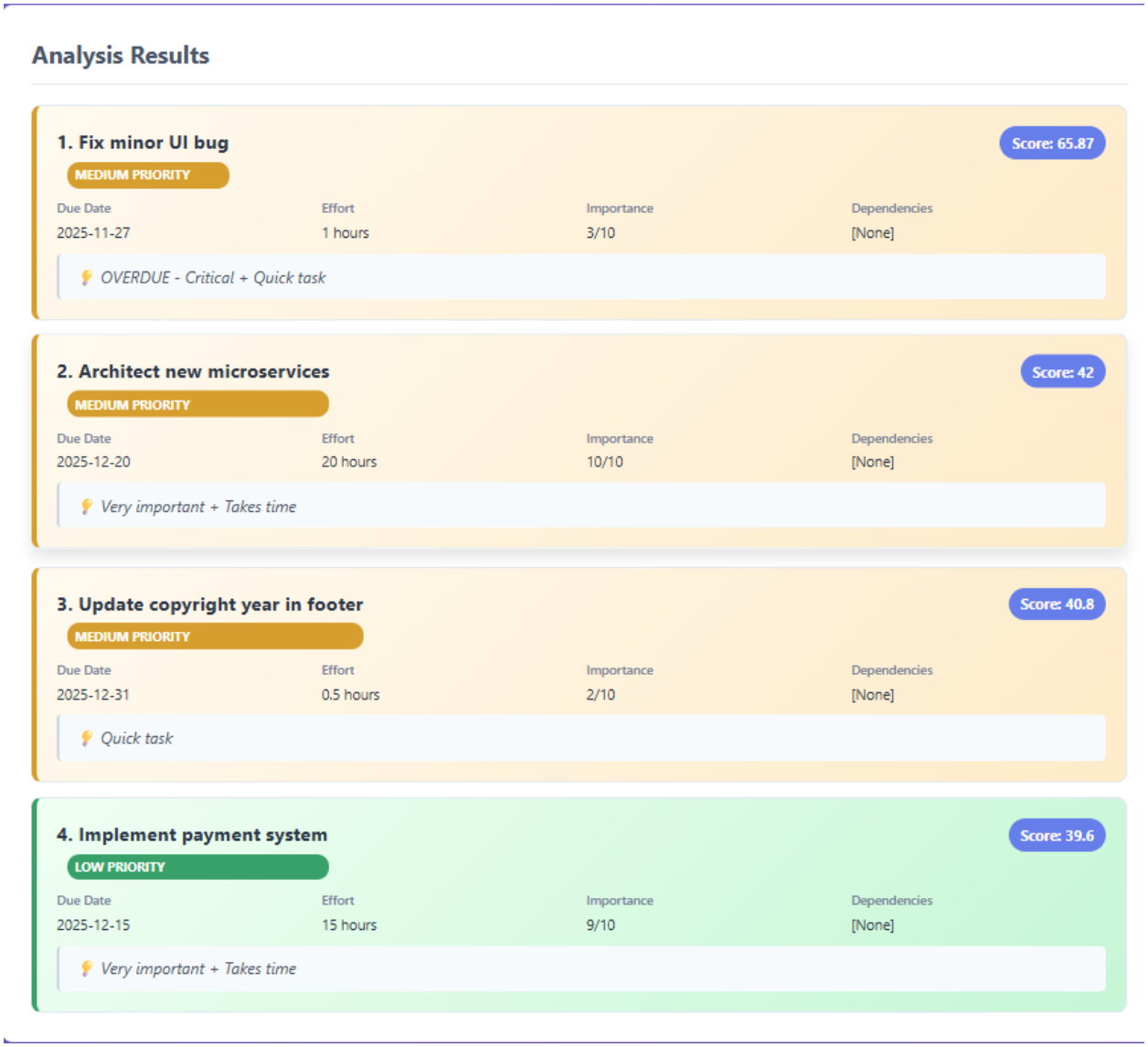


Image showing tasks with calculated priority scores (e.g., "Fix Payment minor UI bug - Score: 65.87 - MEDIUM PRIORITY")

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

What it does:

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

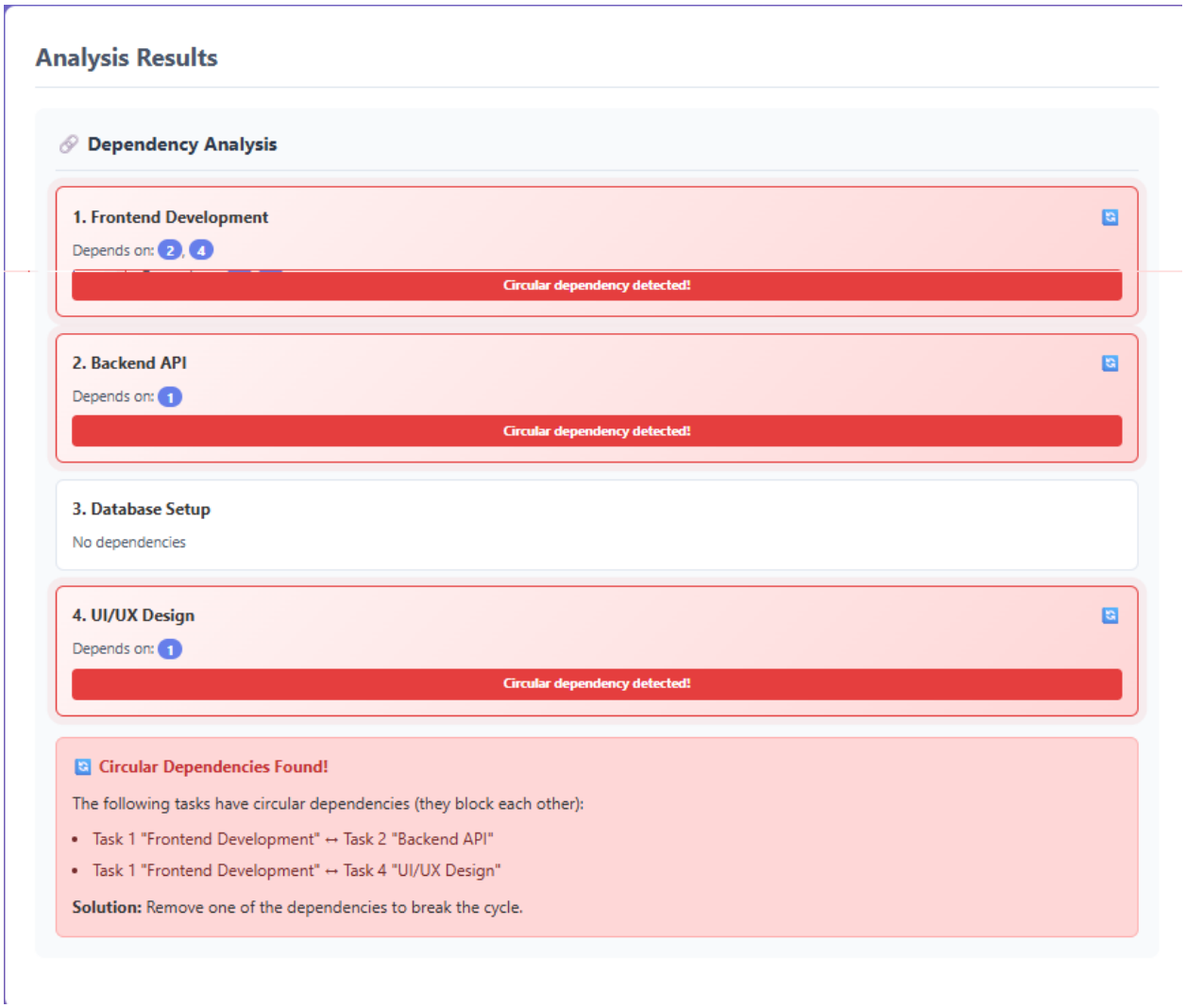
```python
def detect_circular_dependencies(tasks):
    """Detect circular dependencies in tasks"""
    graph = {}

    for i, task in enumerate(tasks):
        task_id = i + 1  # Use index + 1 as task ID
        graph[task_id] = task.get('dependencies', [])

    def has_cycle(node, visited, stack):
        visited[node] = True
        stack[node] = True

        for neighbor in graph.get(node, []):
            if not visited.get(neighbor, False):
                if has_cycle(neighbor, visited, stack):
                    return True
            elif stack.get(neighbor, False):
                return True

        stack[node] = False
        return False

    visited = {}
    stack = {}

    for node in graph:
        if not visited.get(node, False):
            if has_cycle(node, visited, stack):
                return True

    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:

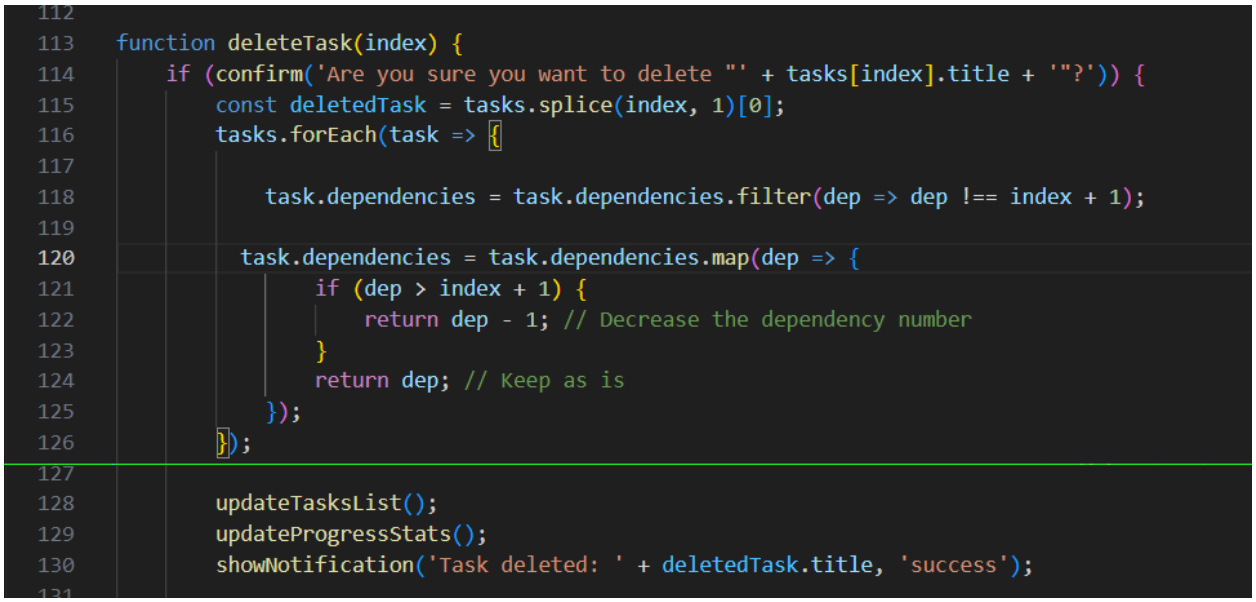


This shows the circular dependencies that were found and the solution to resolve them.

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.

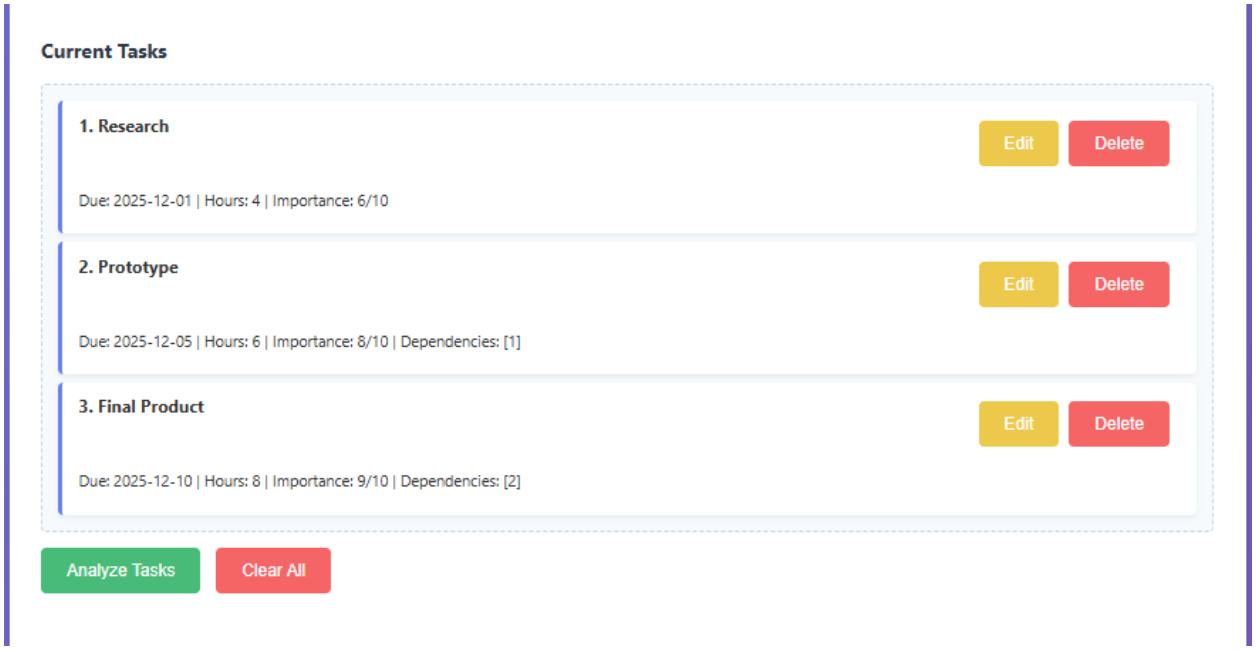


```
function deleteTask(index) {
    if (confirm('Are you sure you want to delete "' + tasks[index].title + '"?')) {
        const deletedTask = tasks.splice(index, 1)[0];
        tasks.forEach(task => {

            task.dependencies = task.dependencies.filter(dep => dep !== index + 1);

          task.dependencies = task.dependencies.map(dep => {
                if (dep > index + 1) {
                    return dep - 1; // Decrease the dependency number
                }
                return dep; // Keep as is
            });
        });

        updateTasksList();
        updateProgressStats();
        showNotification('Task deleted: ' + deletedTask.title, 'success');
```

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 :



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

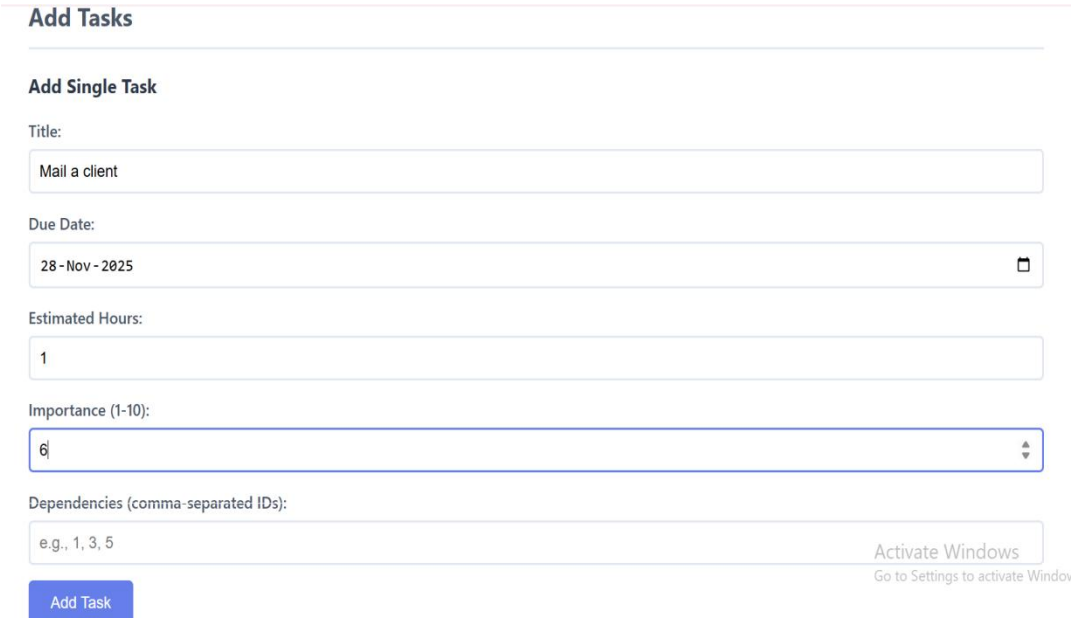
- TASK 2

Title : Plan a monthly budget

Due Date : 30 -11-2025

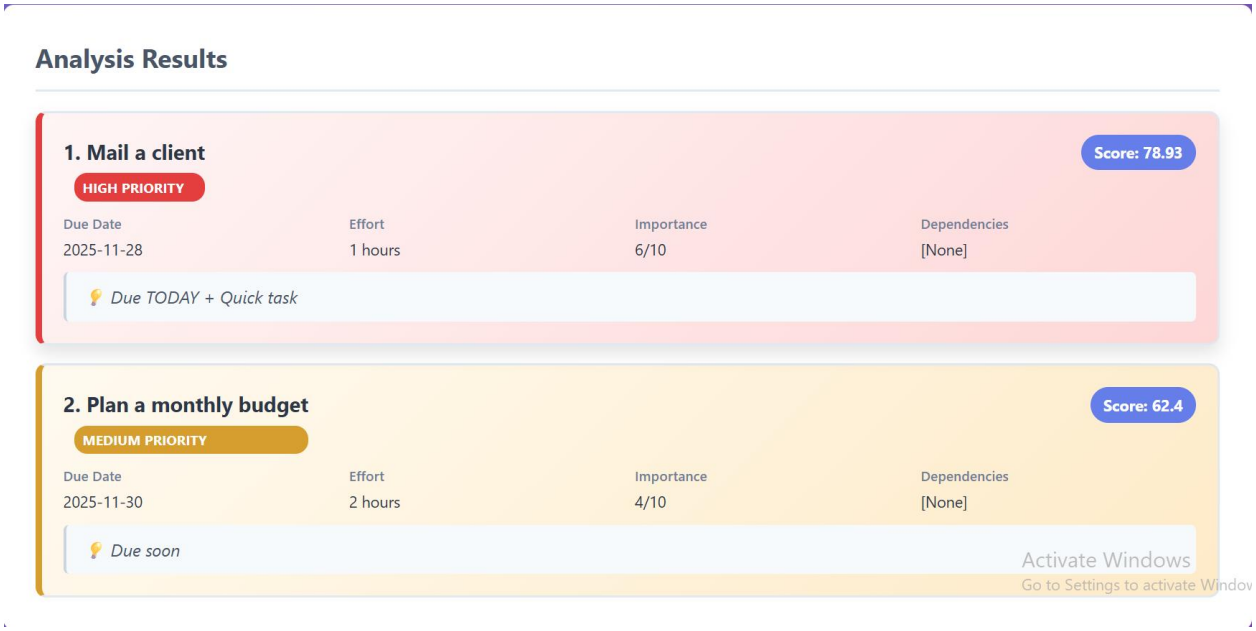
Duration : 2 hr

Importance : 4



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 manual add tasks all at once to get results.

1.DEFAULT TASKS :

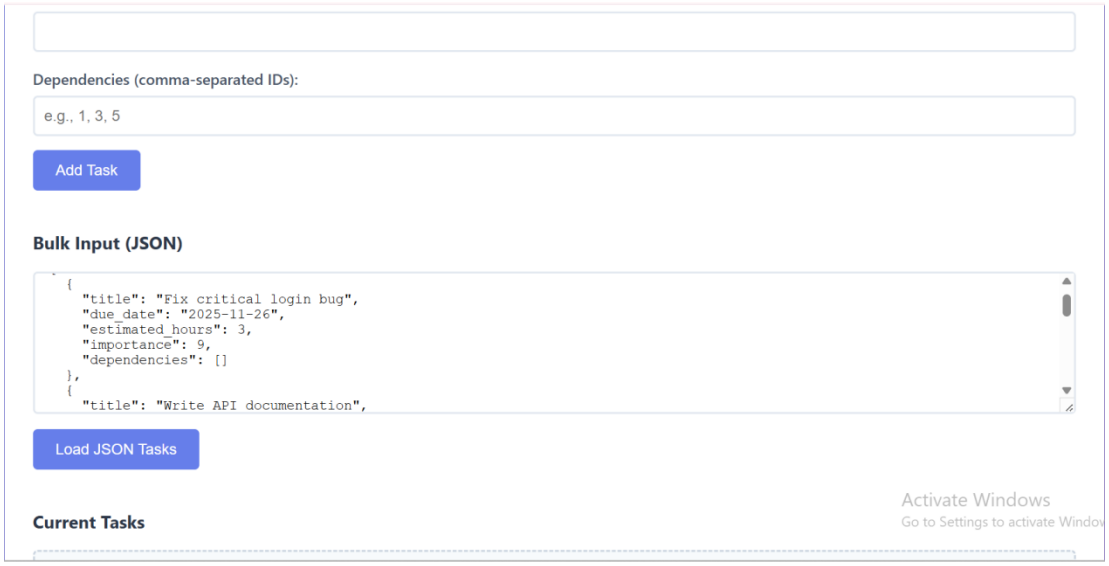
Input

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

Tasks

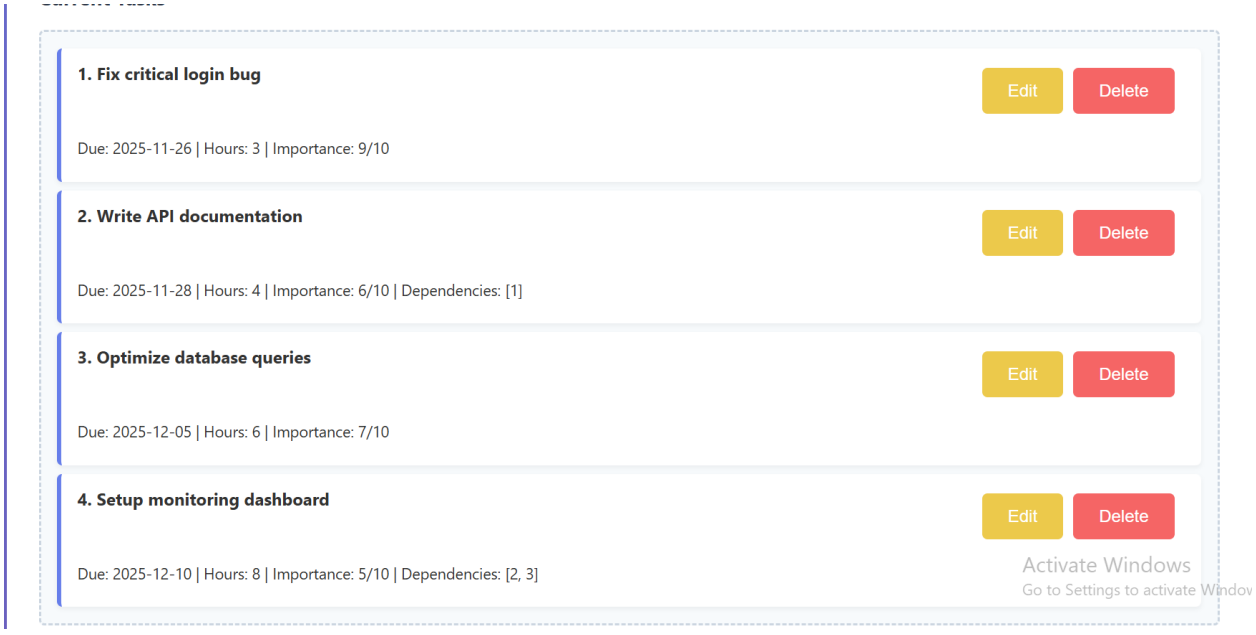
```
[
  {
    "title": "Fix critical login bug",
    "due_date": "2025-11-26",
    "estimated_hours": 3,
    "importance": 9,
    "dependencies": []
  },
  {
    "title": "Write API documentation",
    "due_date": "2025-11-28",
    "estimated_hours": 4,
    "importance": 6,
    "dependencies": [
      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.

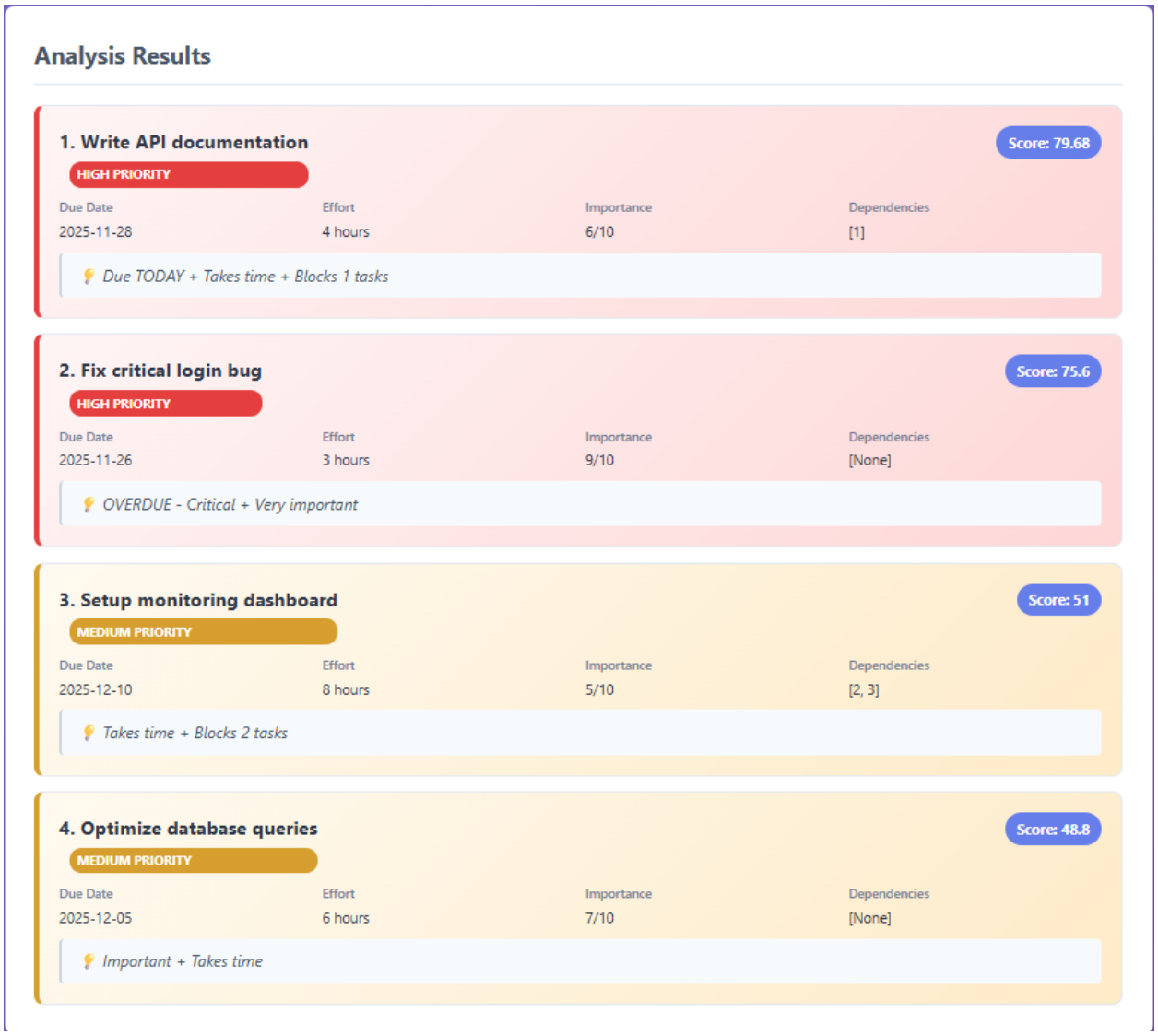


Onloading the JSON Tasks

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



Output



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

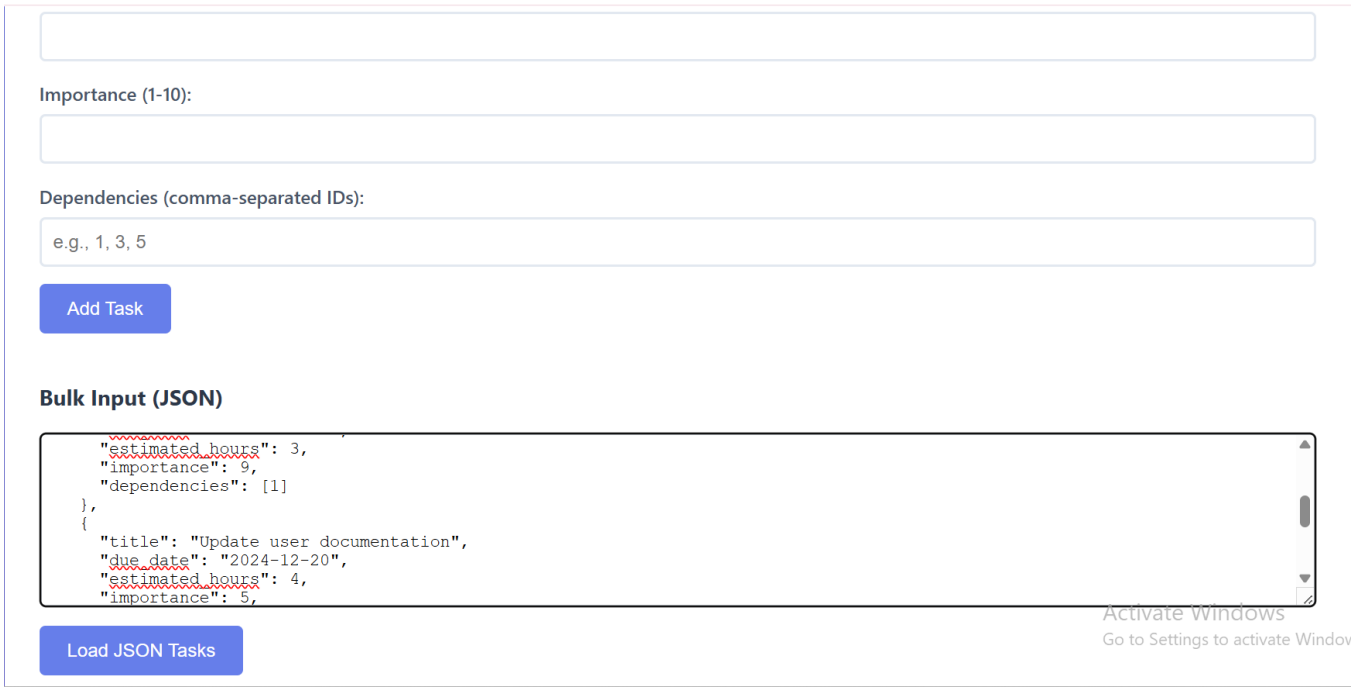
These are the tasks that appears when you manually add as a bulk when you want to avoid single tasks process.

Tasks

You can add your own tasks.

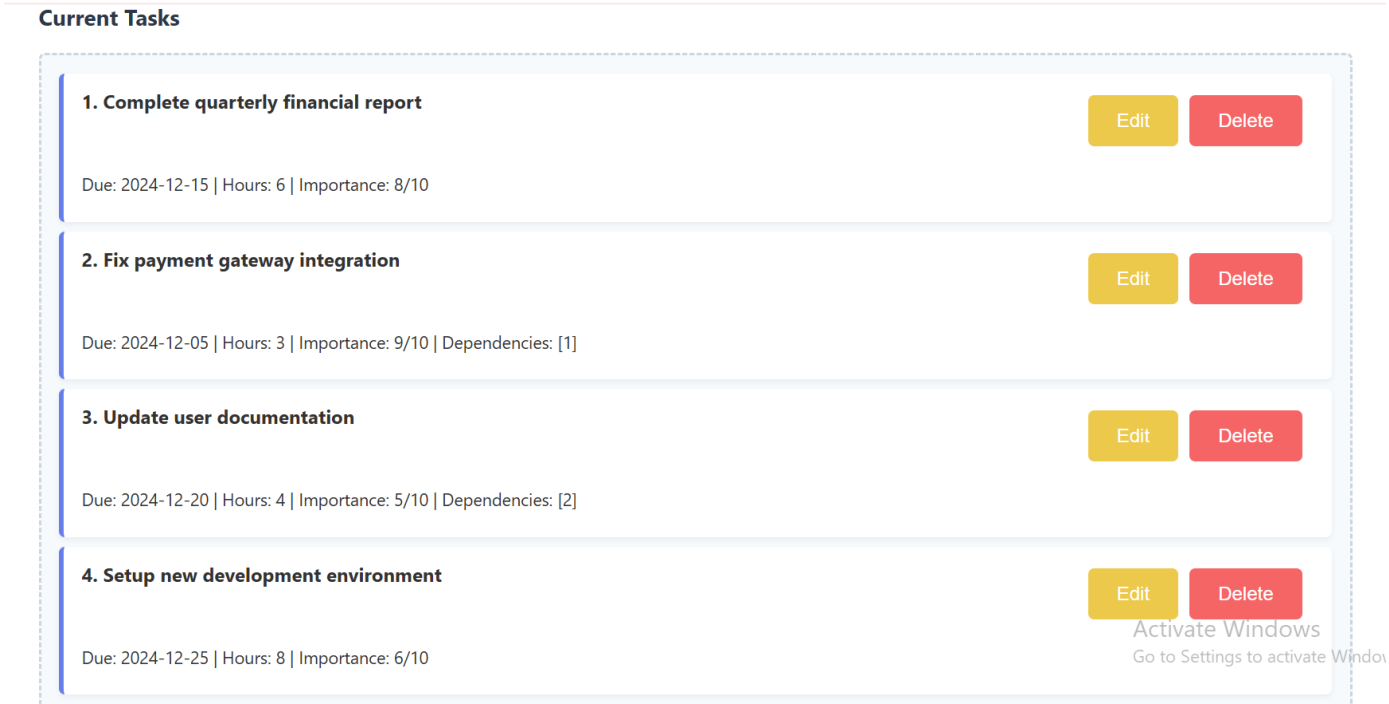
```
[
  {
    "title": "Complete quarterly financial report",
    "due_date": "2024-12-15",
    "estimated_hours": 6,
    "importance": 8,
    "dependencies": []
  },
  {
    "title": "Fix payment gateway integration",
    "due_date": "2024-12-05",
    "estimated_hours": 3,
    "importance": 9,
    "dependencies": [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.

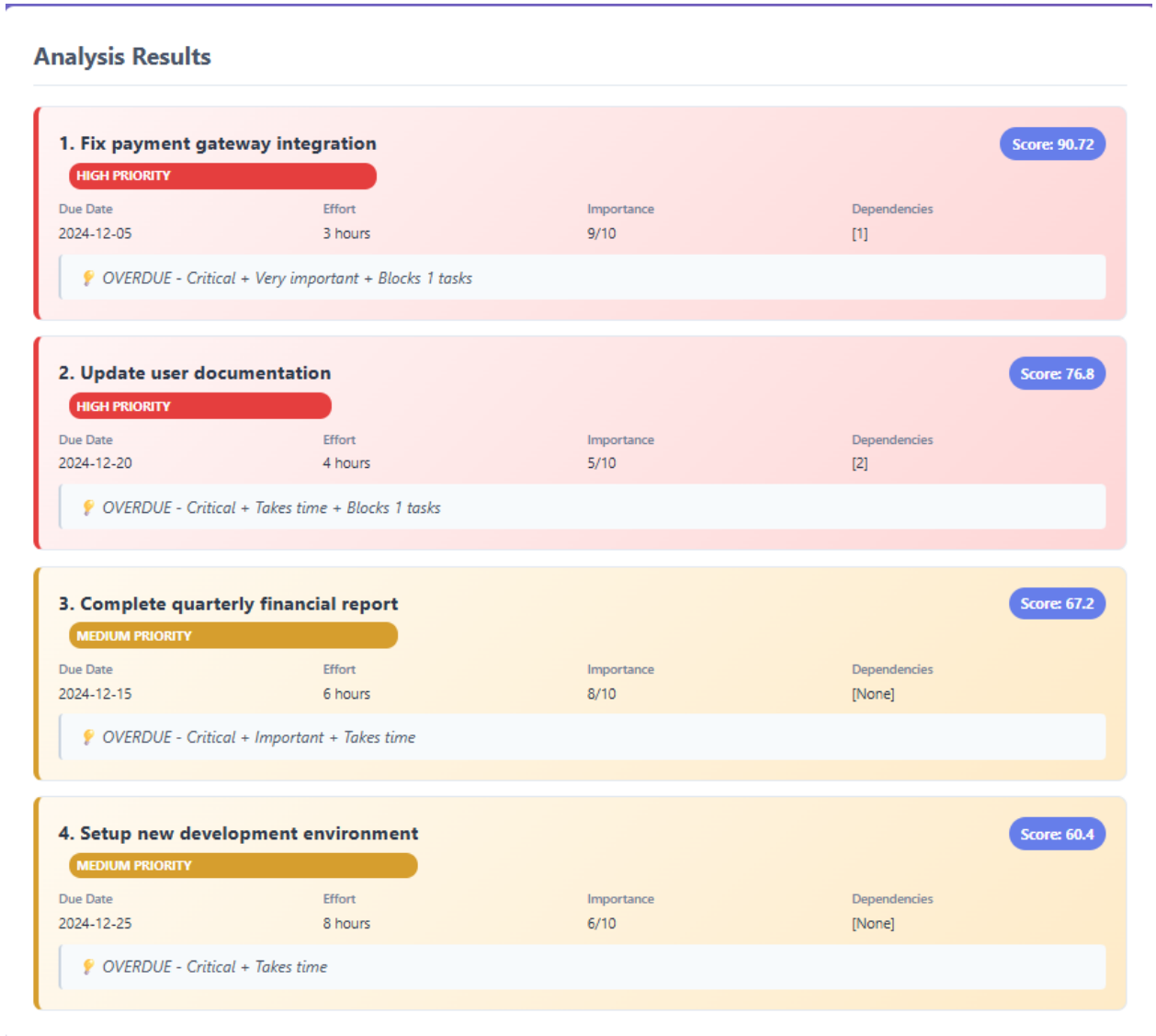


Onloading the JSON Tasks

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



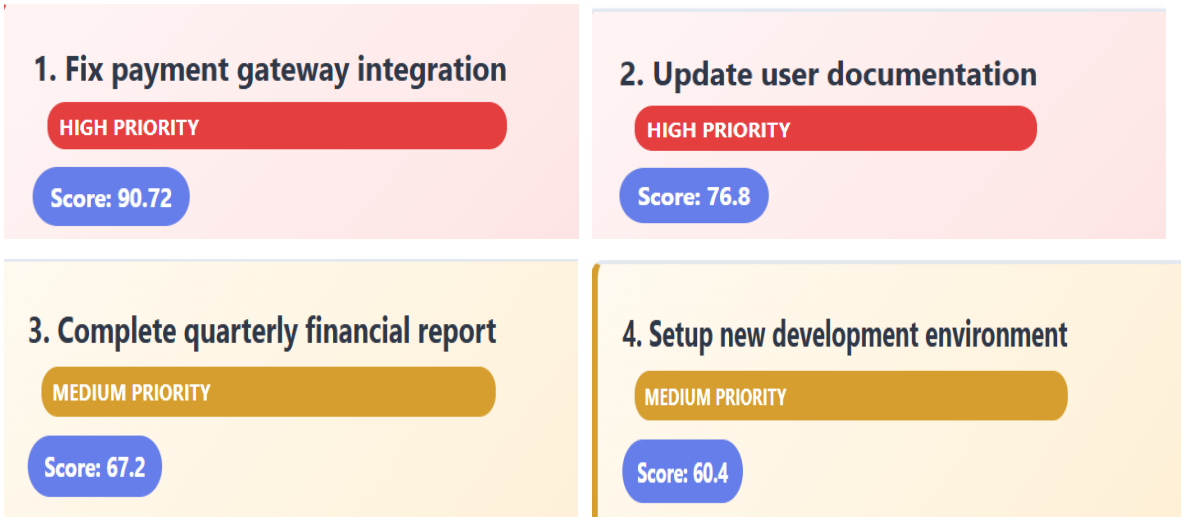
Output



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:**

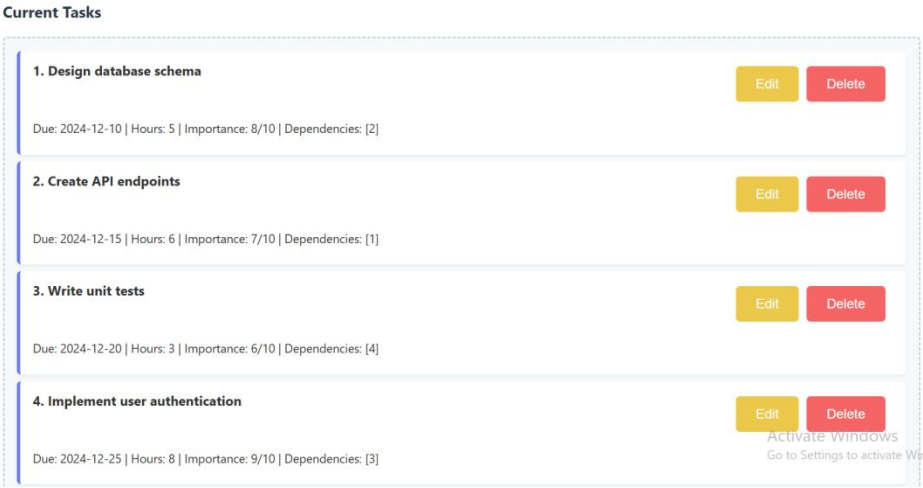
- **Implementation :** Uses depth-first search algorithm to detect dependency loops.

- **Visualization:** Interactive dependency graphs that highlight circular relationships

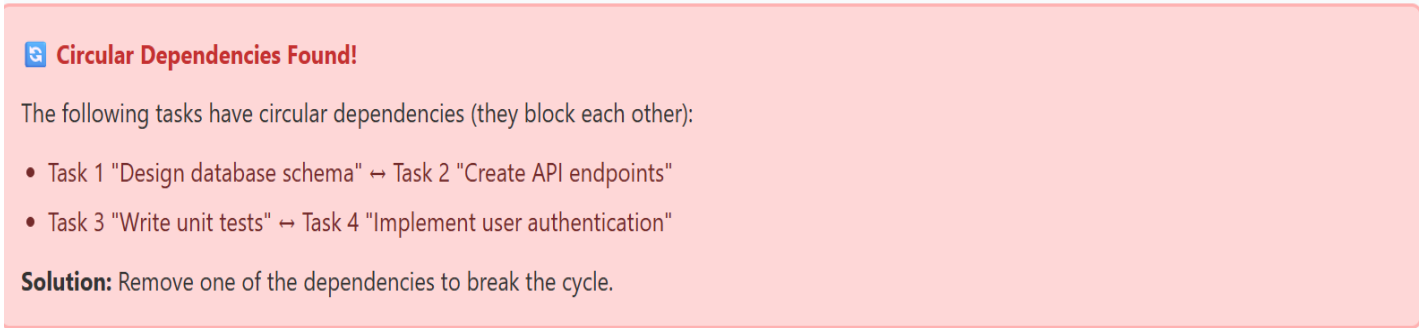
- **Benefit :** Prevents task deadlocks and identifies problematic task chains.

INPUT

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



OUTPUT



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

- **Comprehensive Testing :**

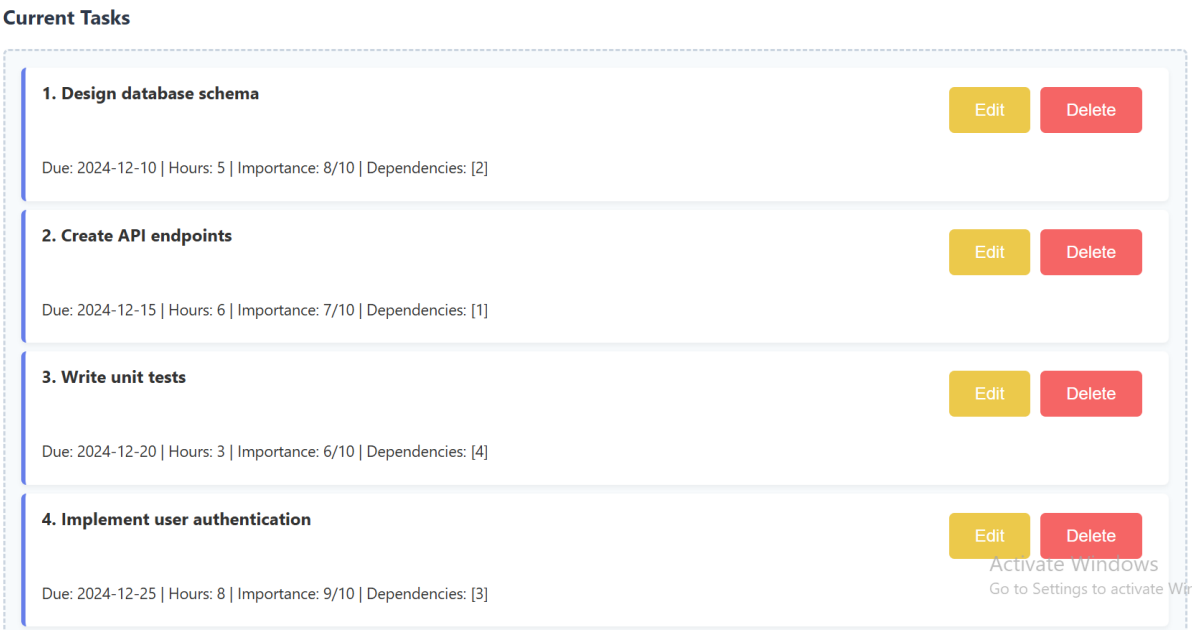
- **Coverage** : Unit tests for all scoring algorithms and edge cases.
- **Strategy Testing :** Validation of different prioritization strategy configurations.
- **Reliability** : Ensures consistent and accurate priority calculations.

- **Dependency Visualization:**

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

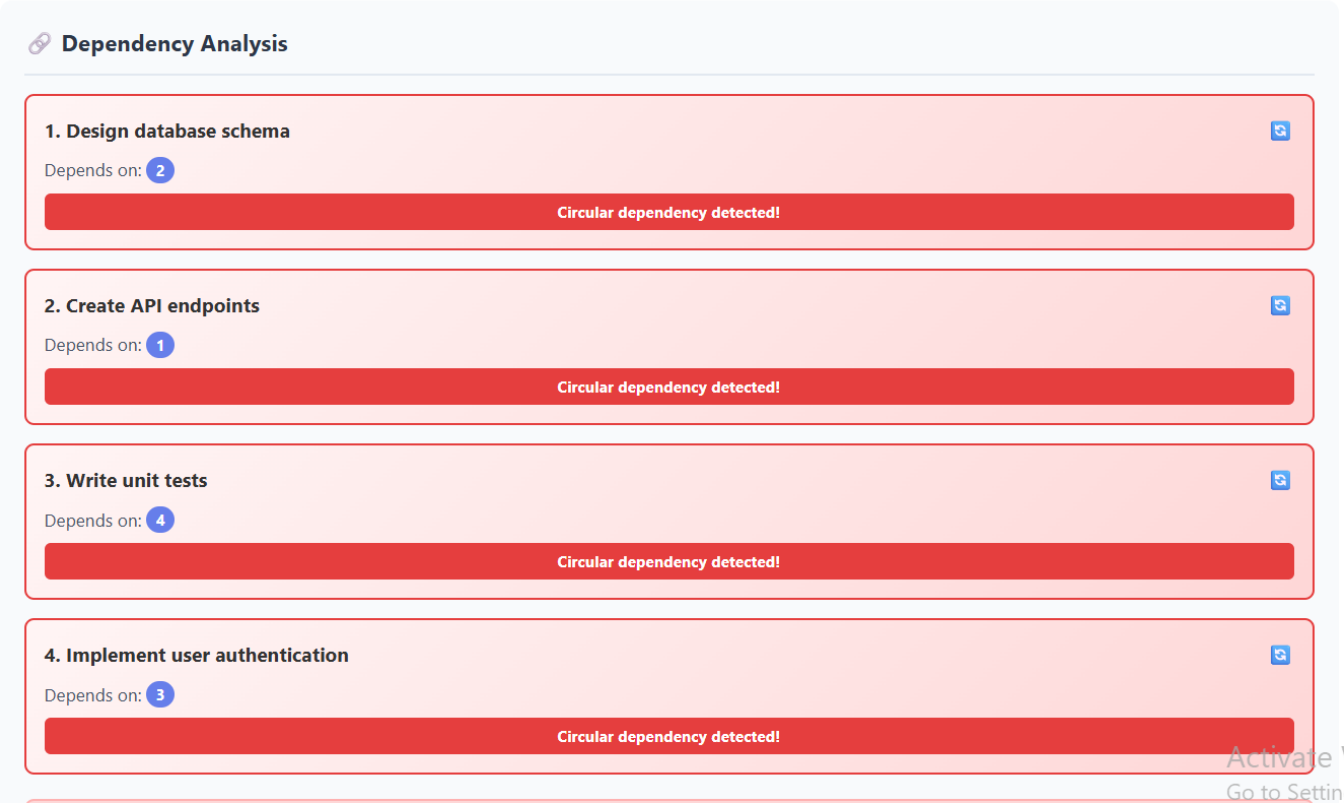
INPUT

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



OUTPUT

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



- **Task Completion Tracking & Progress Monitoring:**

- **Completion Rate Tracking :** Real-time progress metrics

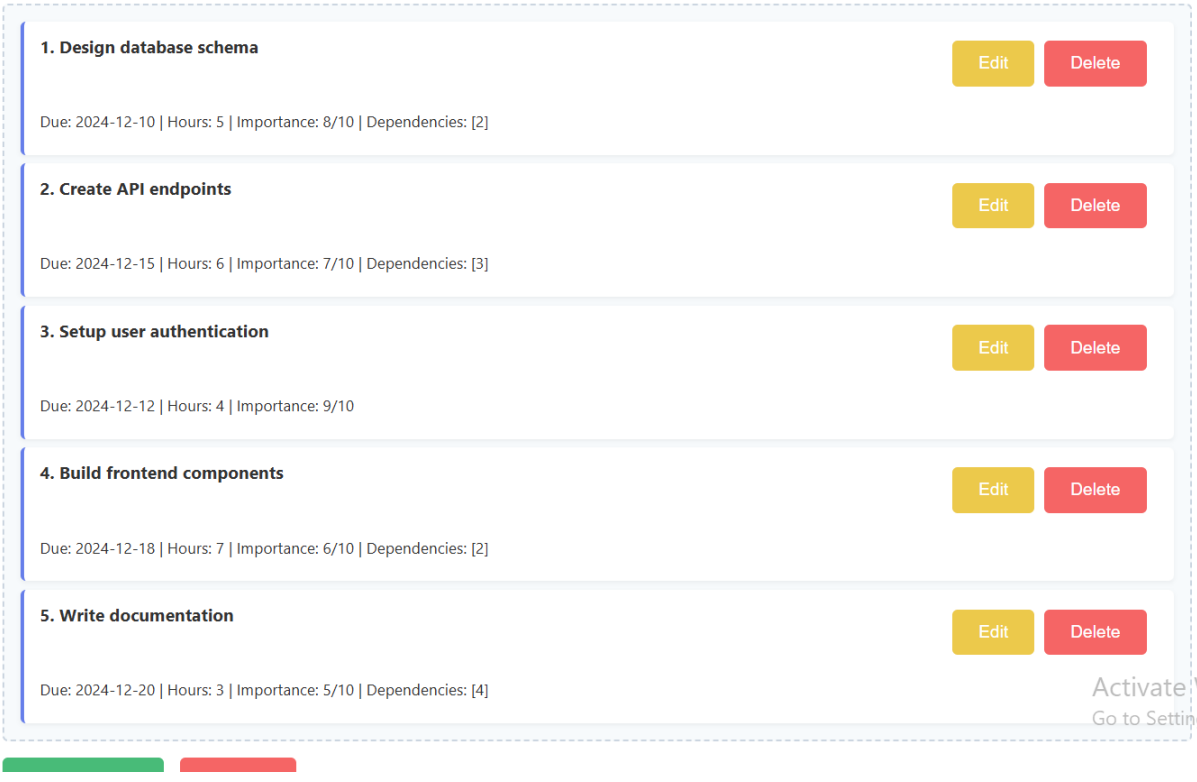
- **Progress Visualization :** Visual completion indicators

- **Performance Metrics :** Total tasks, completed tasks, and completion percentage tracking

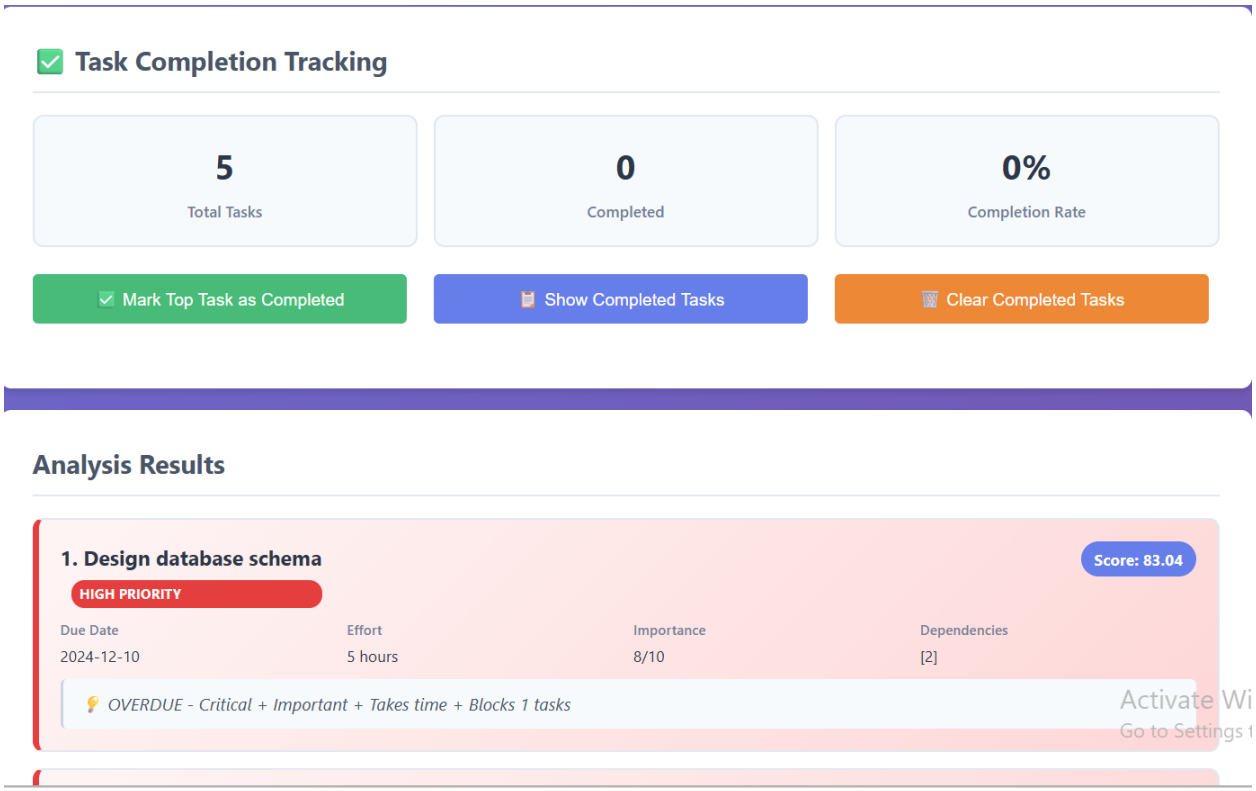
- **Historical Data :** Completion history for productivity analysis.

INPUT

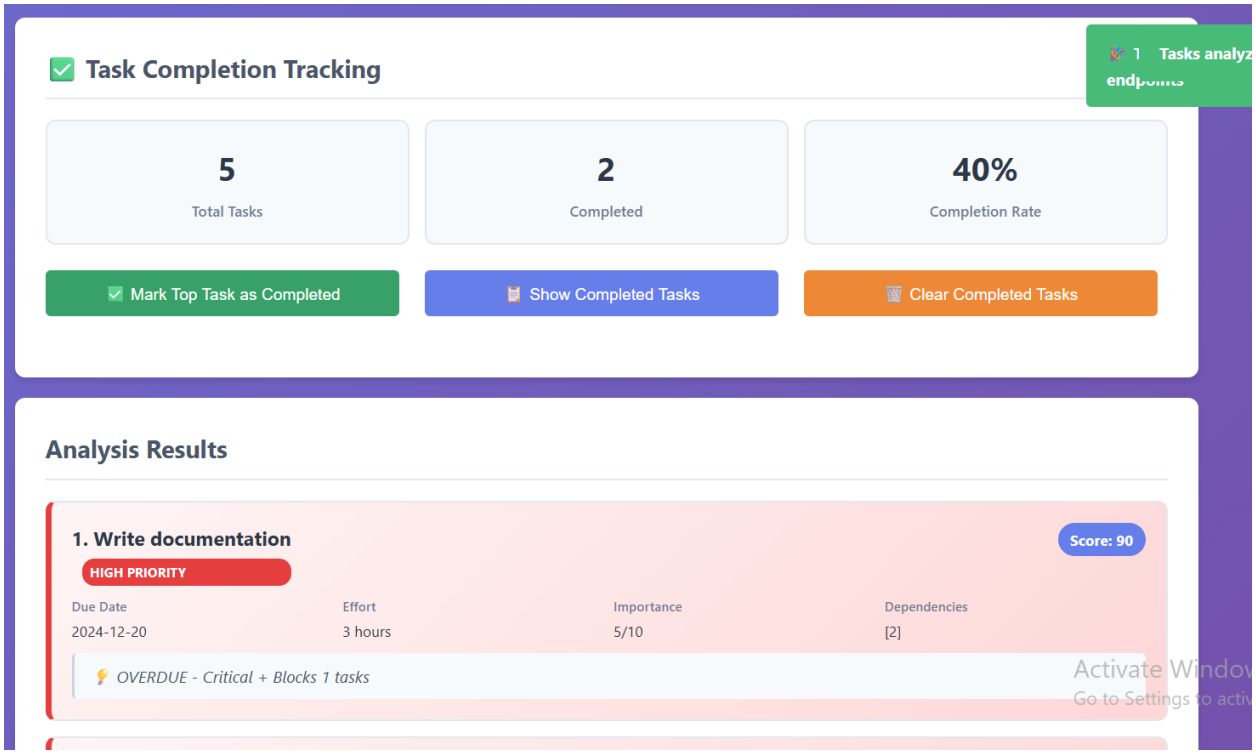
1.When given tasks



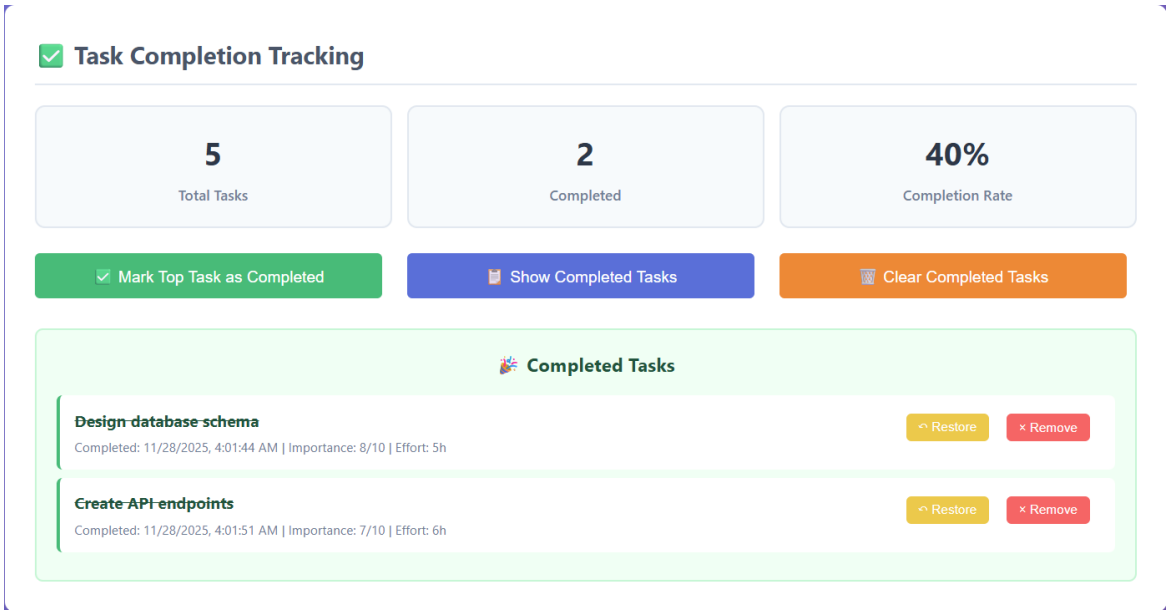
2.Starts tracking



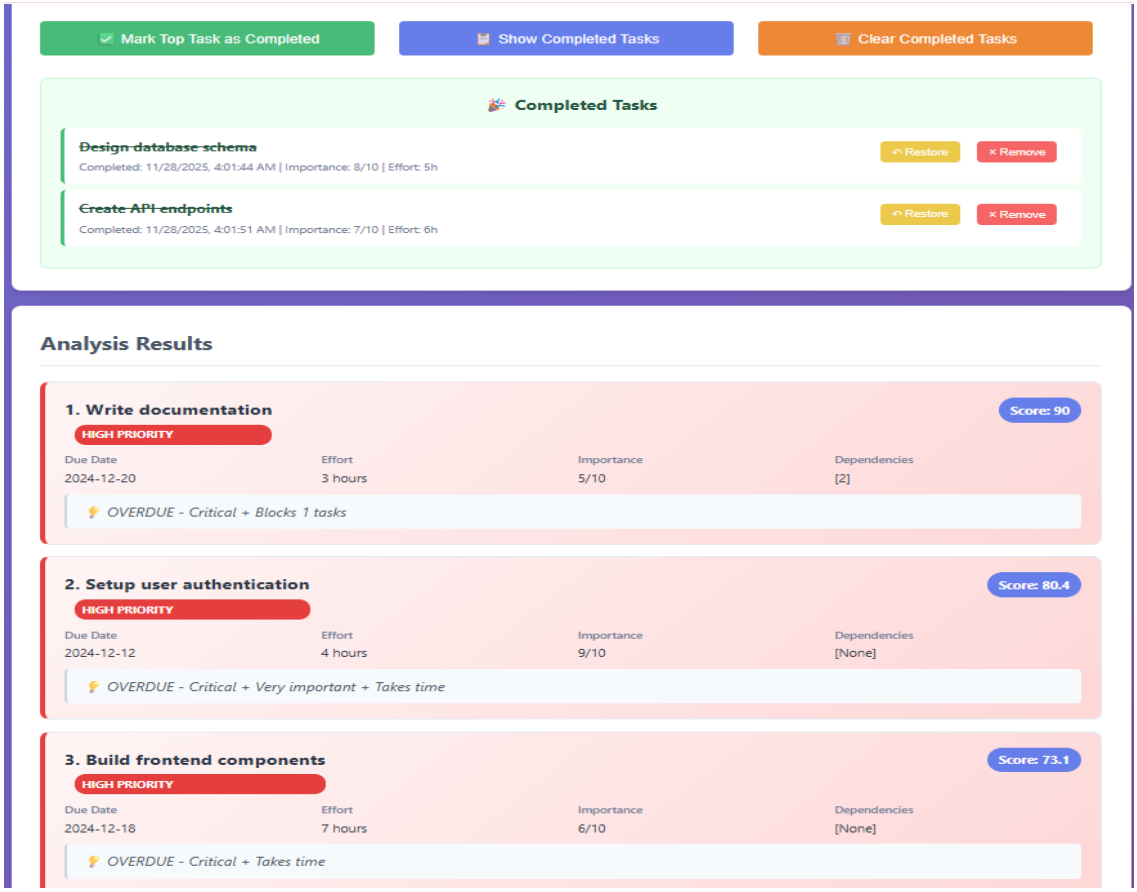
3.After completion of the two highest priority tasks.



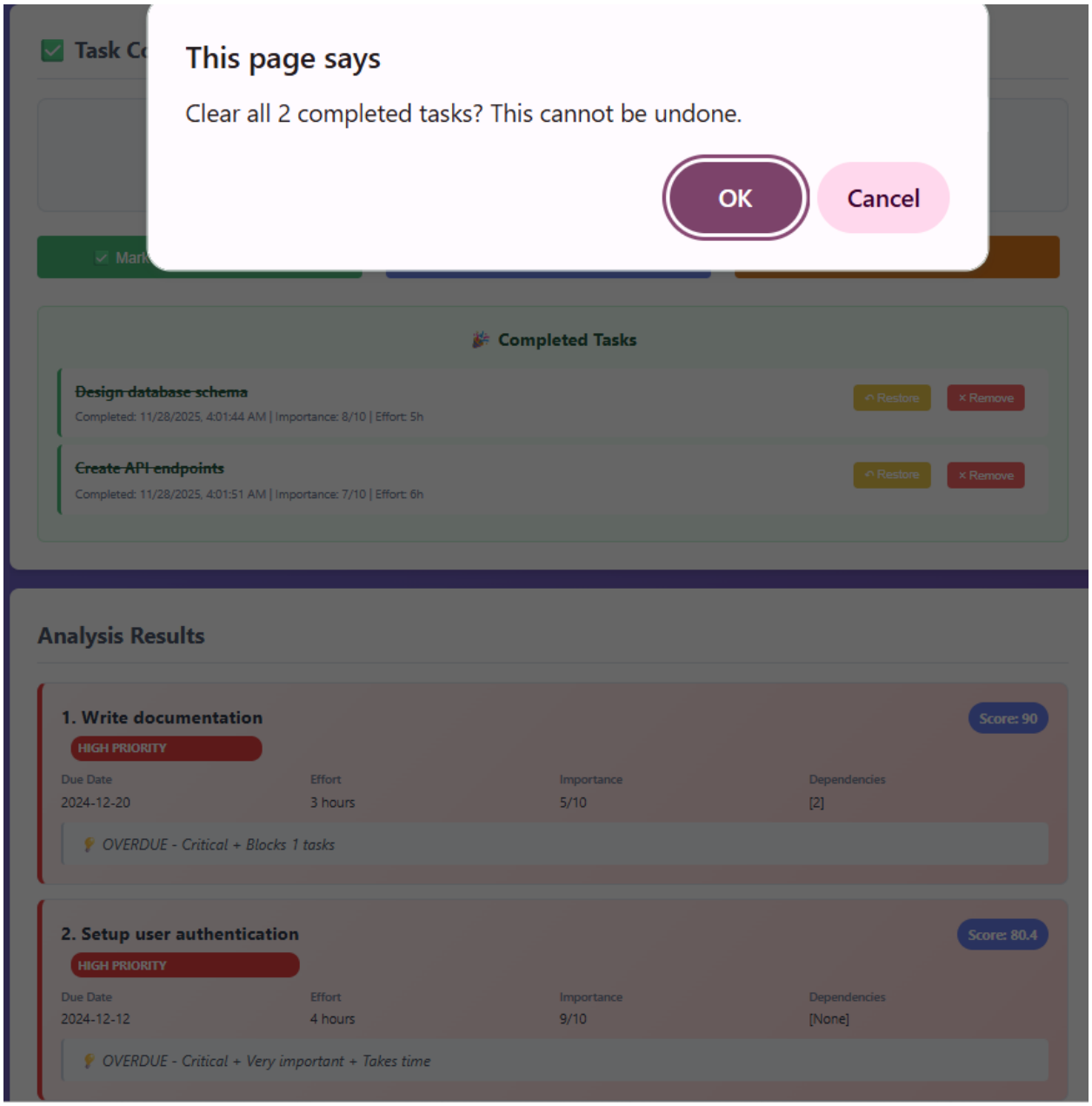
4. You can see the completed tasks



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

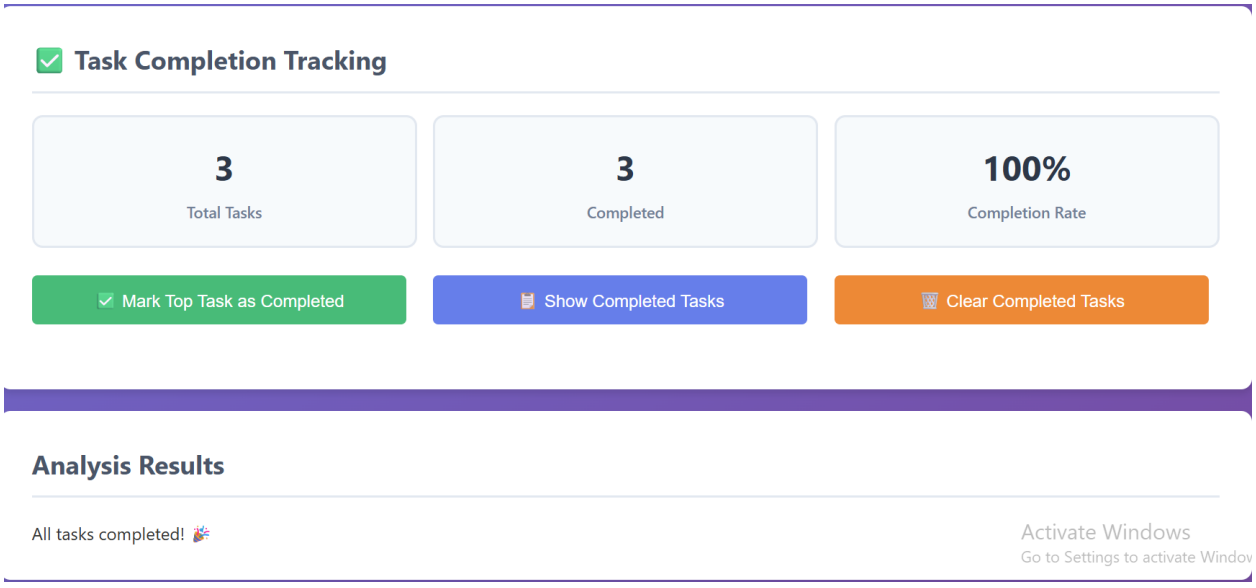


6. You can also see the automatic updating of task numbers and dependency number.



OUTPUT

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



- **Enhanced UI/UX:**

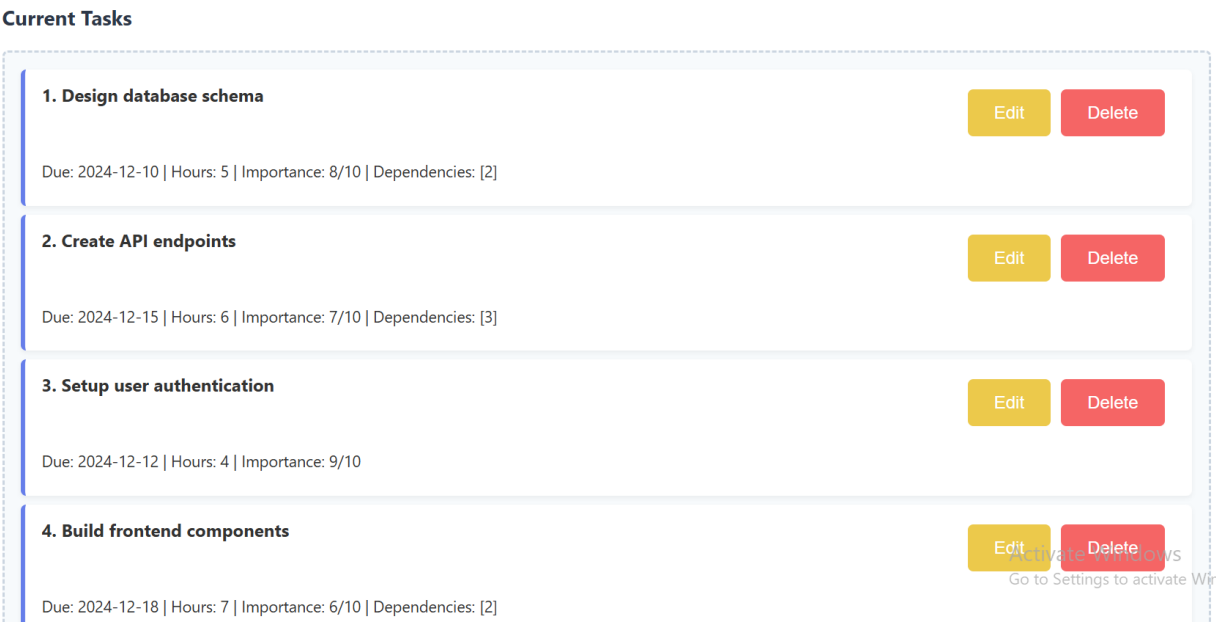
- **Edit/Delete Operations :** Full task management capabilities.

- **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

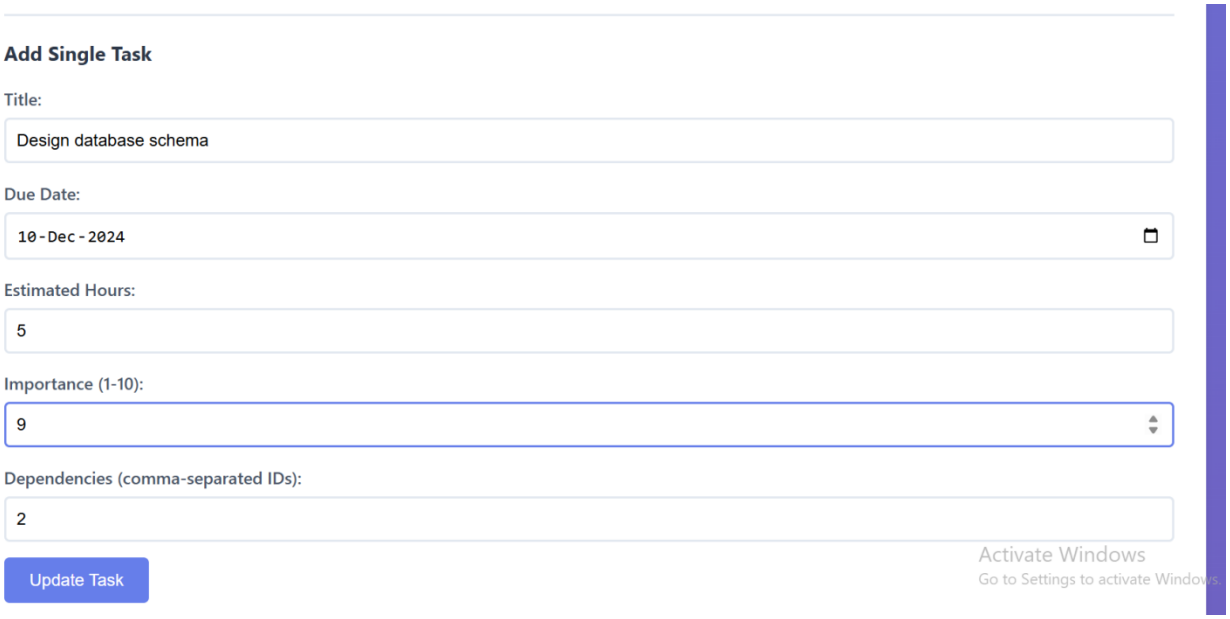
If you enter information incorrectly, you can click 'Edit' to correct it. You can also click 'Delete' to remove a task entirely.



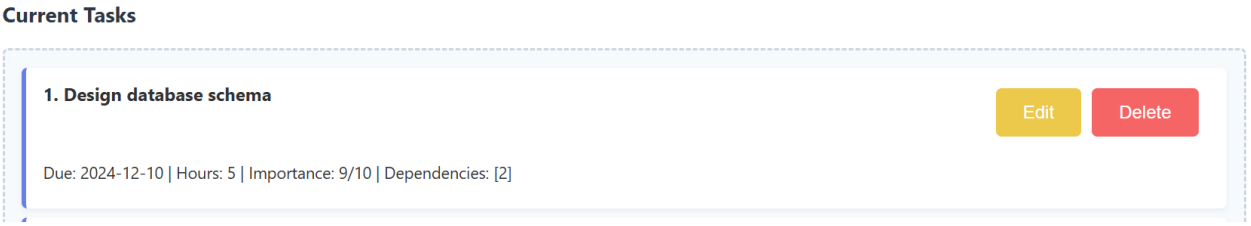
After

1.Edit :

Changing importance of Task 1 from 8 to 9.



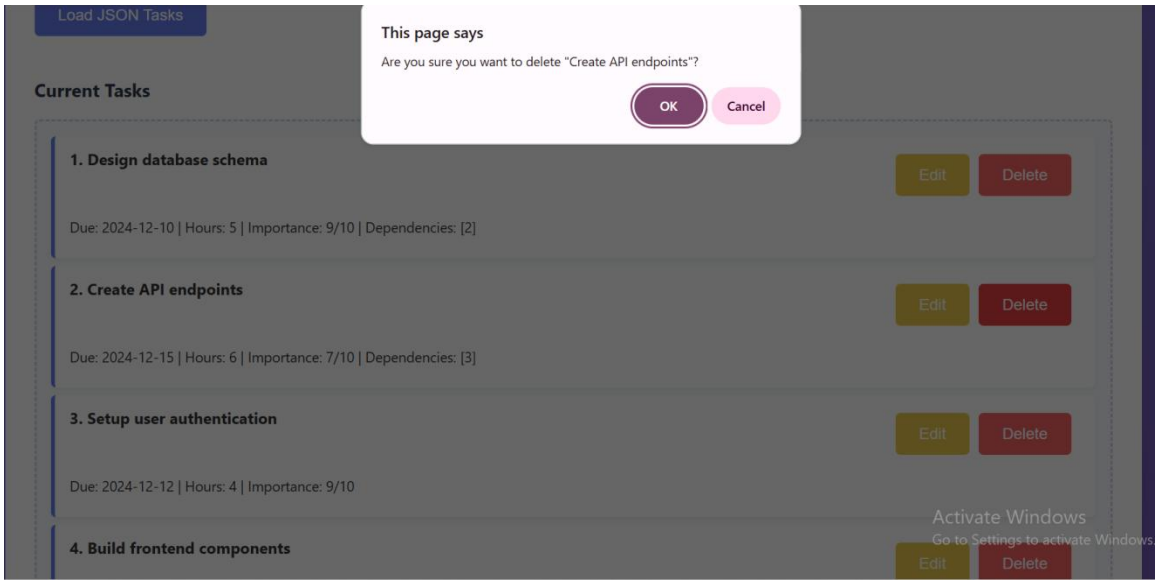
Before , the importance of Task 1 is 8 . After editing it got changed to 9.



2. Delete:

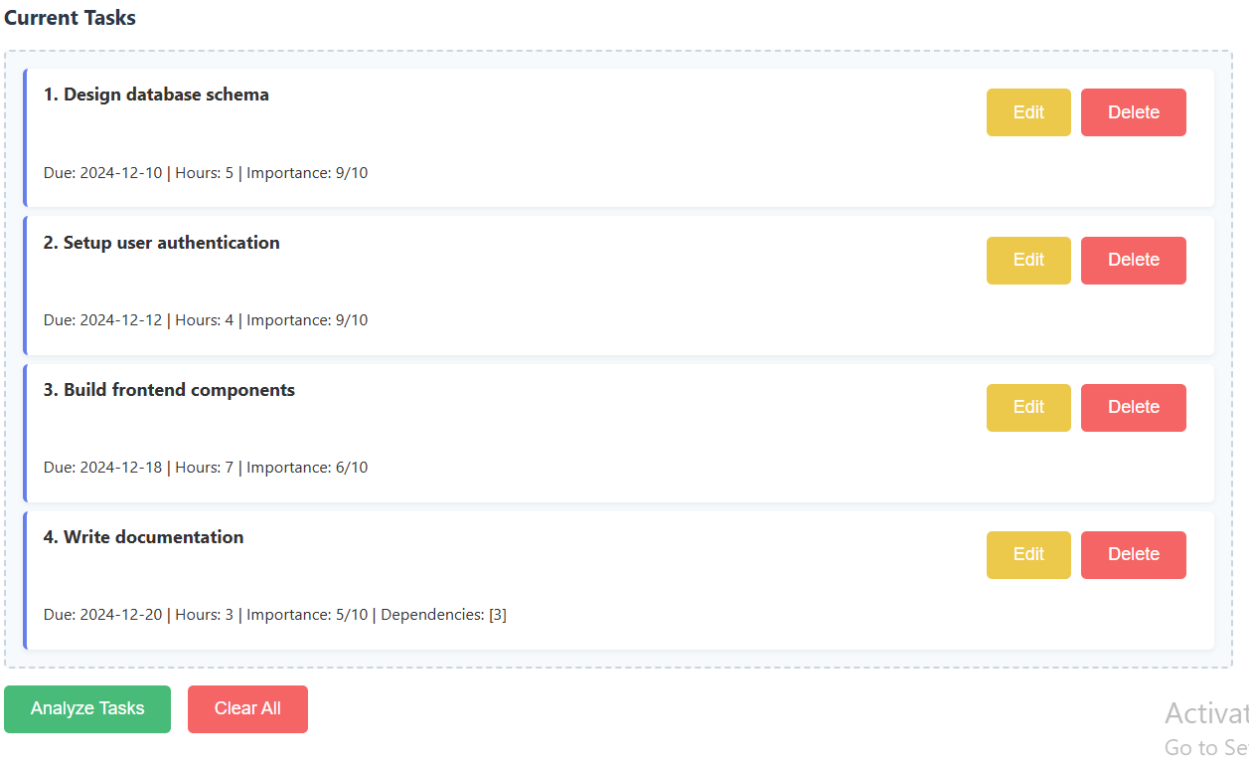
Before:

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".

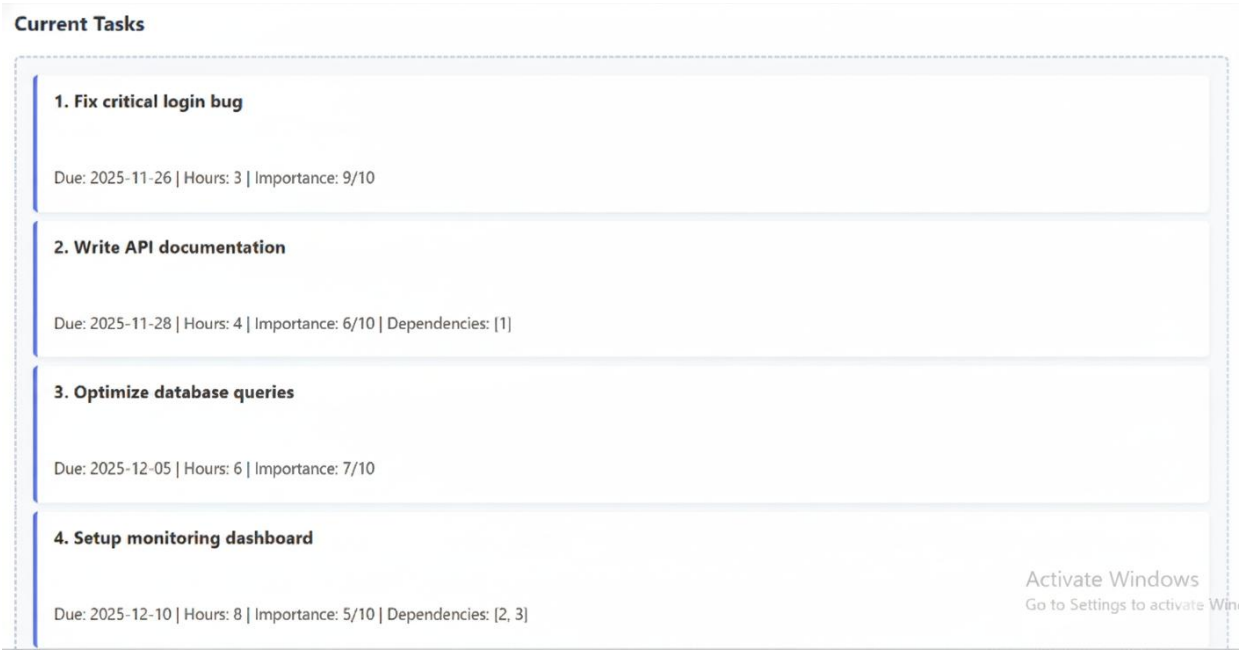


6. Challenges & Solutions

Data Correction Complexity :

- **Challenge:**

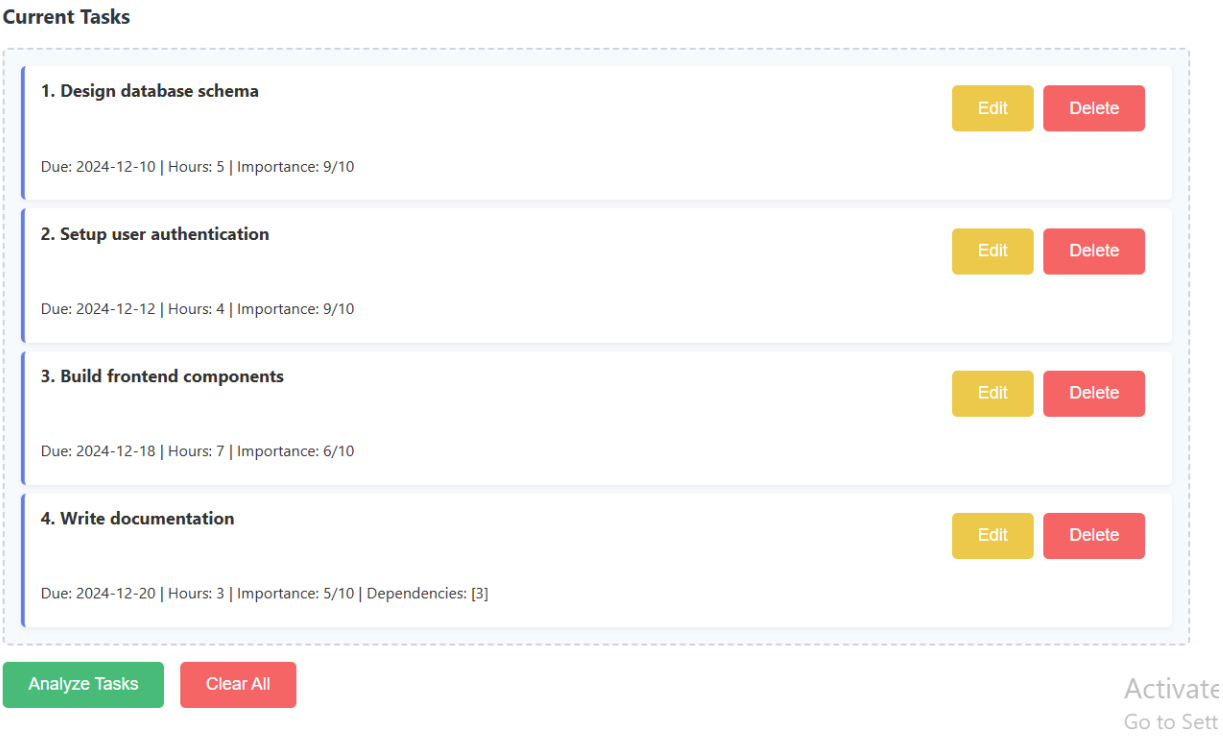
Initial system lacked task modification capabilities, forcing users to delete and recreate tasks for any changes.



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.

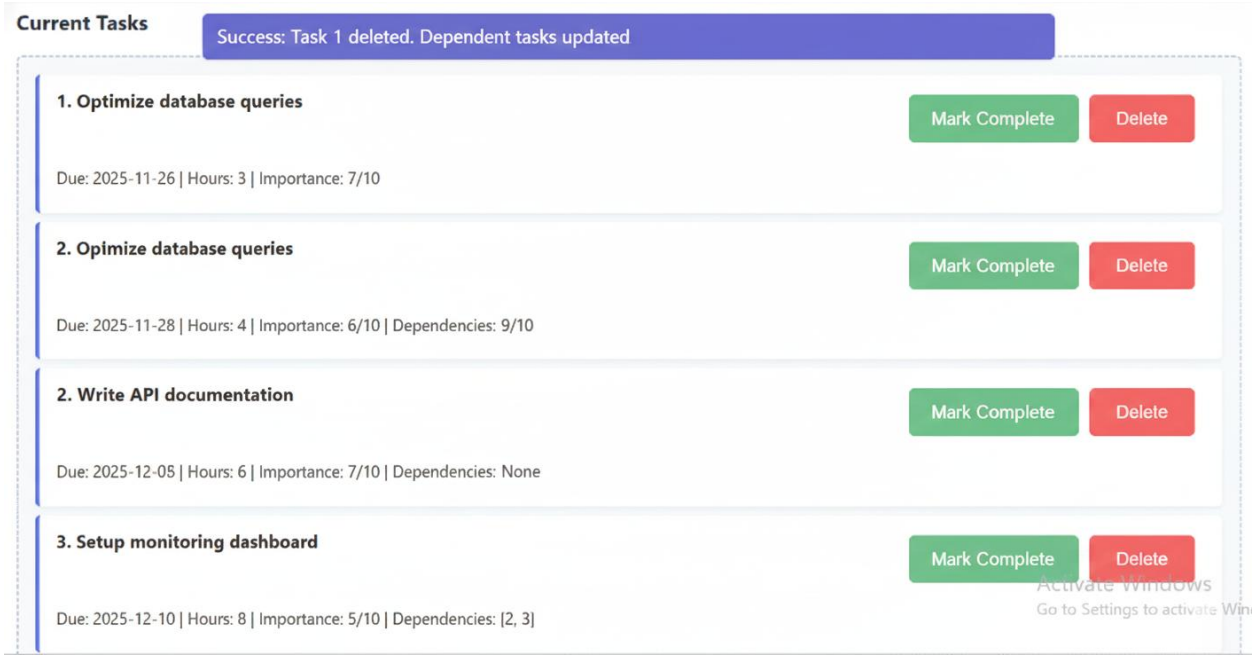


Here , individuals can edit/delete whenever they want.

Dependency Management Issues :

- **Challenge:**

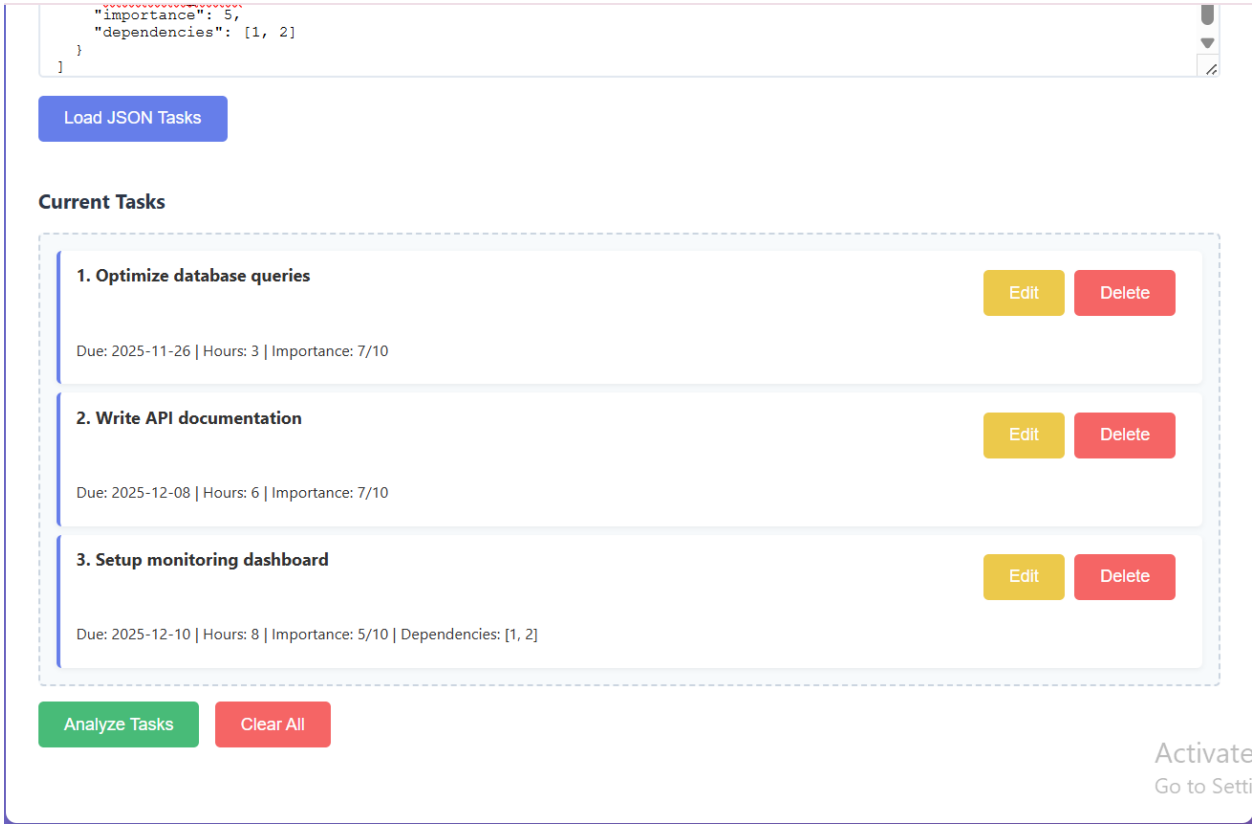
Task deletion caused broken dependency references, creating orphaned tasks relationships and system errors.



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.



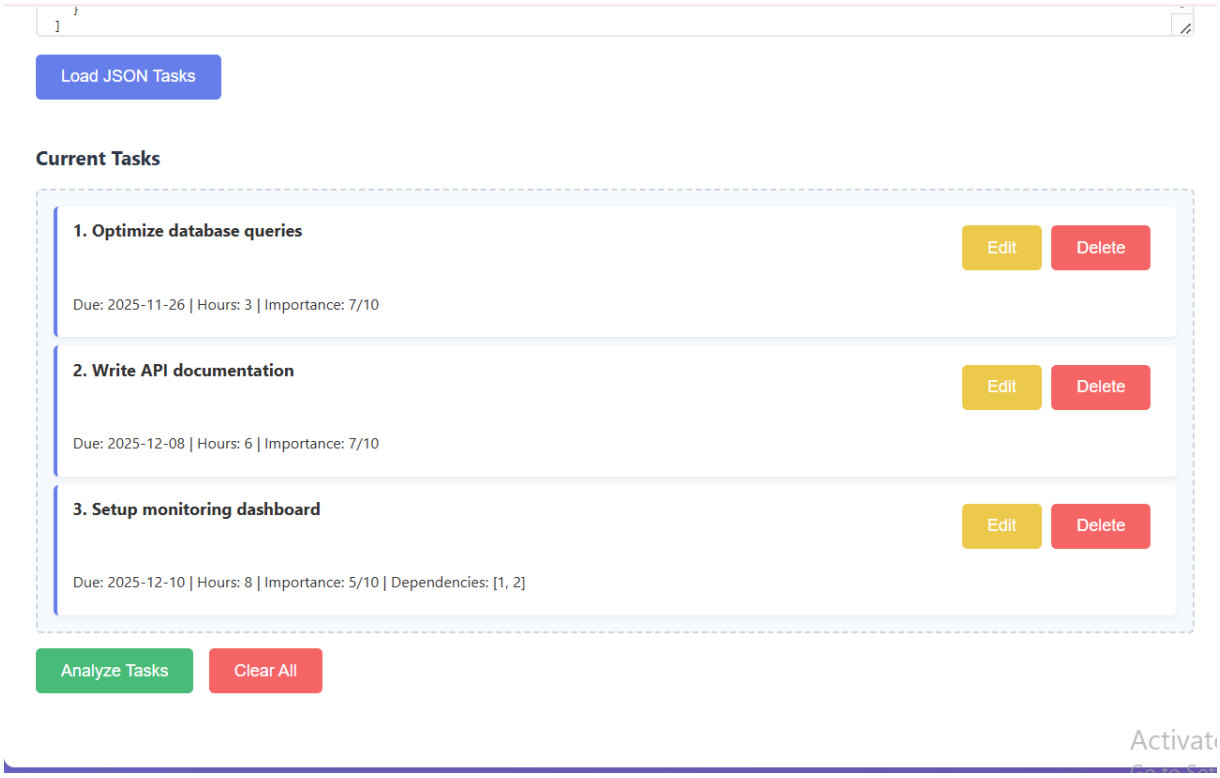
Now, it automatically analyzes and gives updated changes.

Algorithm Score Boundaries :

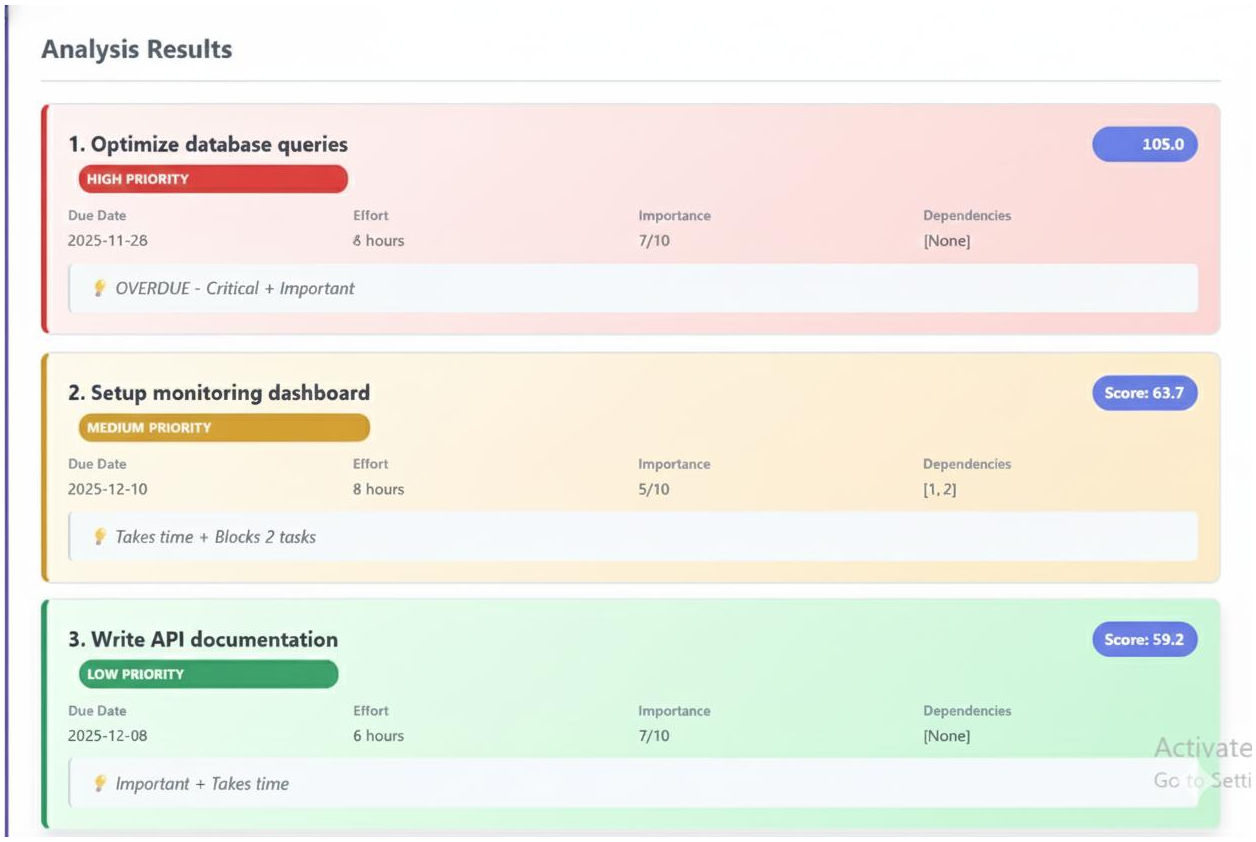
- **Challenge:**

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

INPUT



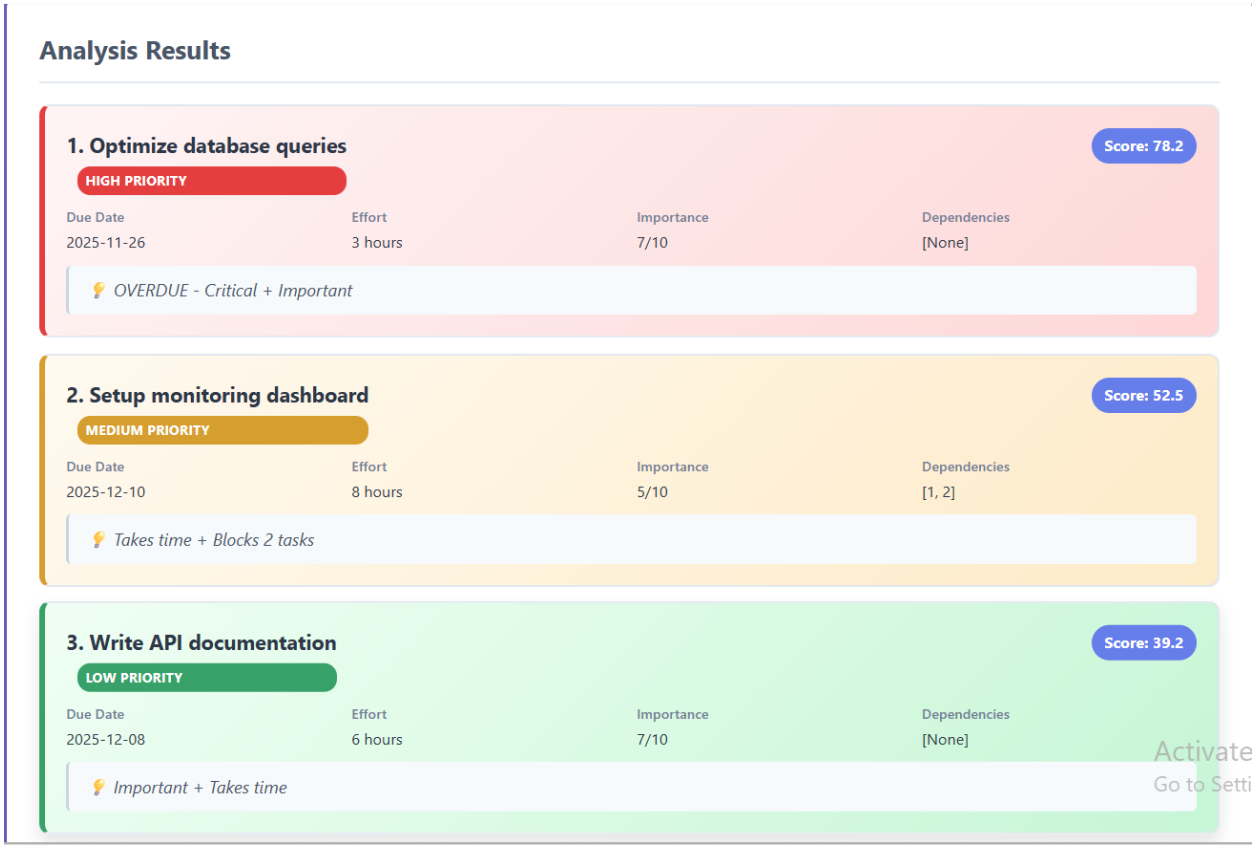
OUTPUT



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.



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