



SIMATS
ENGINEERING



SIMATS
Saveetha Institute of Medical And Technical Sciences
(Declared as Deemed to be University under Section 3 of UGC Act 1956)

Department of Computer Science and Engineering

**CSA4002 –Management Information Systems for Green
Energy**

MIS for Construction Project Management

Team Members:

Indhumathi V (192124215)

P Pragathi (192111315)

Date of Submission:

Abstract

The project focuses on developing a Management Information System (MIS) for Construction Project Management to improve project tracking and resource allocation. The system tracks project timelines, material usage, and workforce allocation while generating real-time progress reports and budget tracking. It helps reduce project delays and improve cost variance to below 10%, enhancing overall resource management. The solution is implemented using Flask, Chart.js, and SQLite, providing an interactive dashboard for real-time data visualization and decision-making.

- Problem:**

Inefficient project tracking and cost overruns.

- Purpose:**

Develop a system to track project timelines, material usage, and workforce allocation.

- Outcome:**

Improved resource management, reduced project delays, and real-time budget tracking.

Table of Contents

- Introduction
- Problem Identification and Analysis Solution
- Design and Implementation Results and Recommendations
Coding
- Output Screenshots
- Reflection on Learning and Personal Development
- Conclusion
- References
- Appendices

Introduction

- **Background:**

Construction projects face delays and cost overruns due to poor tracking.

- **Objective:**

Build an MIS to track and manage project progress.

- **Significance:**

Enhance efficiency and cost control.

- **Scope:**

10+ projects with real-time monitoring.

- **Methodology:**

Flask (Python), Chart.js, HTML, CSS, and SQLite.

Problem Identification and Analysis

- Problem:**

Poor resource allocation and budget overruns.

- Evidence:**

High variance (>10%) in budget and schedule slippage.

- Stakeholders:**

Project Managers, Engineers, Workforce.

- Findings:**

Lack of real-time visibility in project progress and costs.

Solution Design and Implementation

- **Design:**

Flask-based web app with data visualization using Chart.js.

- **Tools:**

Flask, SQLite, Chart.js, HTML, CSS, JavaScript.

- **Solution:**

- Real-time data entry
- Dynamic chart updates
- Resource and budget prediction

Results and Recommendations

- **Results:**

-

- ✓□ Improved resource allocation
- ✓□ Reduced project delays (<10%)
- ✓□ Accurate real-time budget tracking

- **Challenges:**

Data sync issues during updates.

- **Recommendations:**

Improve prediction model accuracy and UI.

CODING-1

1. Backend (Flask) – app.py

Developed using Flask framework. Handles API requests for data fetching, updating, and processing. Manages CSV-based project data storage and updates in real-time. Key

Functionalities:

- ✓ Load project data from CSV.
- ✓ Adjust resources based on user input.
- ✓ Predict budget using dynamic calculations.
- ✓ Return real-time data for chart updates.

```
1 from flask import Flask, render_template, request, jsonify
2 import pandas as pd
3 import numpy as np
4 from sklearn.linear_model import LinearRegression
5
6 app = Flask(__name__)
7
8 # Load CSV Data
9 data_file = "data/project_data.csv"
10 df = pd.read_csv(data_file)
11
12 # Train the model using Linear Regression
13 def train_model():
14     global model
15     model = LinearRegression()
16     X = df[['progress', 'material_cost', 'workforce']]
17     y = df['budget']
18     model.fit(X, y)
19     train_model()
20
21 # Main route
22 @app.route("/")
23 def index():
24     projects = df.to_dict(orient='records')
25     return render_template("dashboard.html", projects=projects)
26
27 # Route to adjust resources
28 @app.route("/adjust_resources", methods=['POST'])
29 def adjust_resources():
30     data = request.json
31     project_name = data['project_name']
32     df.loc[df['project_name'] == project_name, ['progress', 'material_cost', 'workforce']] = \
33         data['progress'], data['material_cost'], data['workforce']
34     df.to_csv(data_file, index=False)
35     train_model()
36     return jsonify({'success': True, 'message': 'Resources for {project_name} updated successfully'})
37 except Exception as e:
38     return jsonify({'success': False, 'error': str(e)})
39
40 # Route to predict budget
41 @app.route("/predict_budget", methods=['POST'])
42 def predict_budget():
43     data = request.json
44     X_input = np.array([data['progress'], data['material_cost'], data['workforce']])
45     predicted_budget = model.predict(X_input)[0]
46     return jsonify({'success': True, 'predicted_budget': round(predicted_budget, 2)})
47 except Exception as e:
48     return jsonify({'success': False, 'error': str(e)})
49
50 # Route to get chart data
51 @app.route("/get_chart_data", methods=['GET'])
52 def get_chart_data():
53     df = df
54     chart_data = {
55         'project_name': df['project_name'].tolist(),
56         'budget': df['budget'].tolist()
57     }
58     return jsonify(chart_data)
59 except Exception as e:
60     return jsonify({'success': False, 'error': str(e)})
61
62 if __name__ == '__main__':
63     app.run(debug=True)
```


CODING-2

2. Frontend – dashboard.html

Built with HTML + CSS + JavaScript. Uses Chart.js for real-time chart updates. Includes user interaction for adjusting data and predicting budget.

Key Features:

- ✓ Editable table for real-time input.
- ✓ Action buttons to adjust resources and predict budget.
- ✓ Real-time data updates reflected in the chart.

```
chart.js  style.css  dashboard.html  X  project_data.csv  app.py
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8" />
5      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
6      <title>Construction Project Dashboard</title>
7
8      <!-- Link to CSS -->
9      <link rel="stylesheet" href="{{ url_for('static', filename='style.css')}}" />
10
11      <!-- Load Chart.js from CDN -->
12      <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
13  </head>
14  <body>
15      <h1>Construction Project Dashboard</h1>
16
17      <!-- Table for displaying project data -->
18      <table border="1">
19          <thead>
20              <tr>
21                  <th>Project Name</th>
22                  <th>Progress (%)</th>
23                  <th>Material Usage</th>
24                  <th>Manpower</th>
25                  <th>Budget</th>
26                  <th>Actions</th>
27              </tr>
28          </thead>
29          <tbody>
30              <!-- Loop for project in projects -->
31              <tr>
32                  <td>{{ project.project_name }}</td>
33                  <td>
34                      <input type="number" id="progress_{{ project.project_name }}" value="{{ project.progress }}" />
35                  </td>
36                  <td>
37                      <input type="number" id="material_usage_{{ project.project_name }}" value="{{ project.material_usage }}" />
38                  </td>
39                  <td>
40                      <input type="number" id="manforce_{{ project.project_name }}" value="{{ project.manforce }}" />
41                  </td>
42                  <td>
43                      <input type="text" id="budget_{{ project.project_name }}" value="{{ project.budget }}" />
44                  </td>
45                  <td>
46                      <button onclick="adjustResources('{{ project.project_name }}')">Adjust Resources</button>
47                      <button onclick="predictBudget('{{ project.project_name }}')">Predict Budget</button>
48                  </td>
49              </tr>
50              <!-- Loop end -->
51          </tbody>
52      </table>
53
54      <!-- Canvas for displaying the chart -->
55      <div>
56          <h2>Budget Overview</h2>
57          <div style="width: 100%; height: 200px; margin: auto;">
58              <canvas id="budgetChart"></canvas>
59          </div>
60      </div>
61
62      <!-- Link to the JavaScript file -->
63      <script src="{{ url_for('static', filename='chart.js')}}"></script>
64  </body>
65 </html>
```

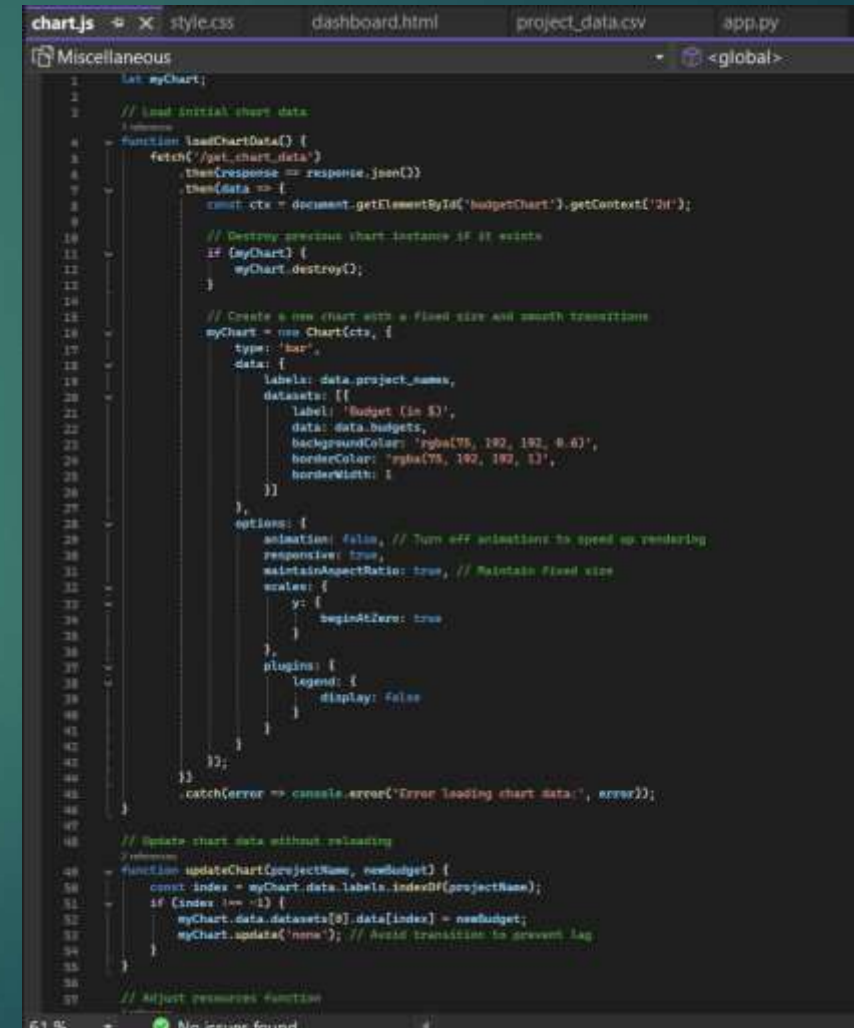
CODING-3

3. Chart Integration – chart.js

Uses Chart.js for real-time visualization. Dynamically updates when table data is adjusted.

Key Functionalities:

- ✓ Load data from Flask backend
- ✓ Update chart directly without reload
- ✓ Smooth transition for real-time effect



```
chart.js  ✕ style.css  dashboard.html  project_data.csv  app.py
Miscellaneous  <global>

1  let myChart;
2
3  // Load initial chart data
4  function loadChartData() {
5    fetch('/get_chart_data')
6      .then(response => response.json())
7      .then(data => {
8        const ctx = document.getElementById('budgetChart').getContext('2d');
9
10         // Destroy previous chart instance if it exists
11         if (myChart) {
12           myChart.destroy();
13         }
14
15         // Create a new chart with a fixed size and smooth transitions
16         myChart = new Chart(ctx, {
17           type: 'bar',
18           data: {
19             labels: data.project_names,
20             datasets: [{
21               label: 'Budget (in $)',
22               data: data.budgets,
23               backgroundColor: 'rgba(75, 192, 192, 0.6)',
24               borderColor: 'rgba(75, 192, 192, 1)',
25               borderWidth: 1
26             }]
27           },
28           options: {
29             animation: false, // Turn off animations to speed up rendering
30             responsive: true,
31             maintainAspectRatio: true, // Maintain fixed size
32             scales: {
33               y: {
34                 beginAtZero: true
35               }
36             },
37             plugins: {
38               legend: {
39                 display: false
40               }
41             }
42           }
43         });
44       })
45       .catch(error => console.error('Error loading chart data:', error));
46
47   // Update chart data without reloading
48   function updateChart(projectName, newBudget) {
49     const index = myChart.data.labels.indexOf(projectName);
50     if (index !== -1) {
51       myChart.data.datasets[0].data[index] = newBudget;
52       myChart.update('none'); // Avoid transition to prevent lag
53     }
54   }
55
56   // Adjust resources function
```

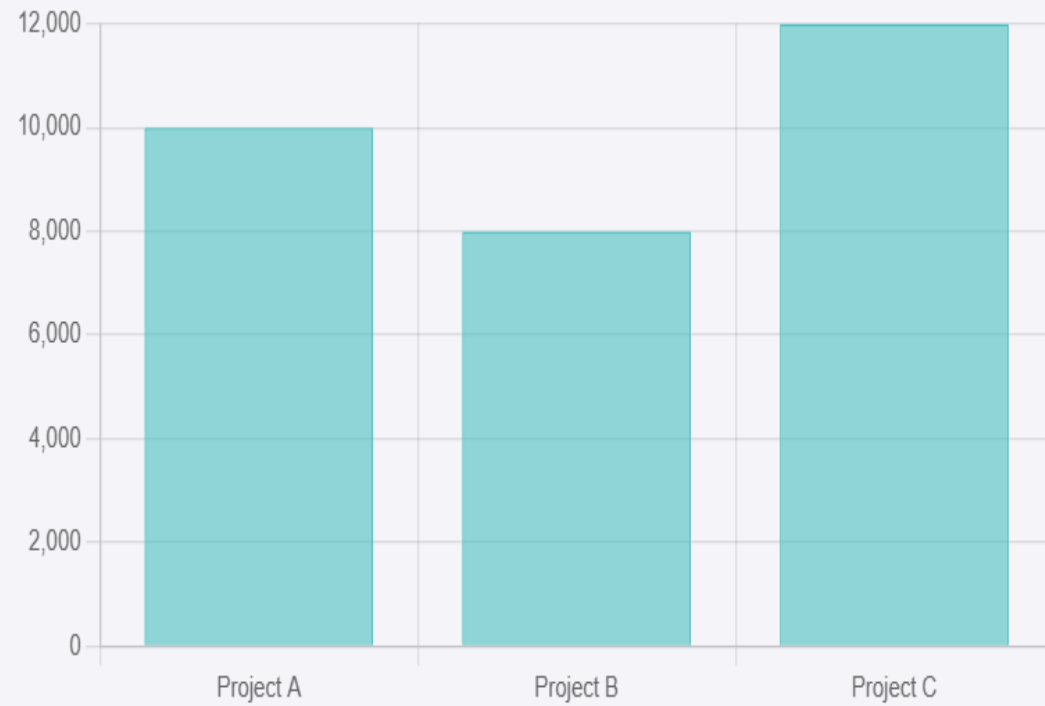
OUTPUT DASHBOARD OVERVIEW



Project Name	Progress (%)	Material Usage	Workforce	Budget
Project A	<input type="text" value="75"/>	<input type="text" value="124"/>	<input type="text" value="20"/>	10000
Project B	<input type="text" value="50"/>	<input type="text" value="100"/>	<input type="text" value="10"/>	8000
Project C	<input type="text" value="90"/>	<input type="text" value="140"/>	<input type="text" value="23"/>	12000

OUTPUT CHART

Budget Overview



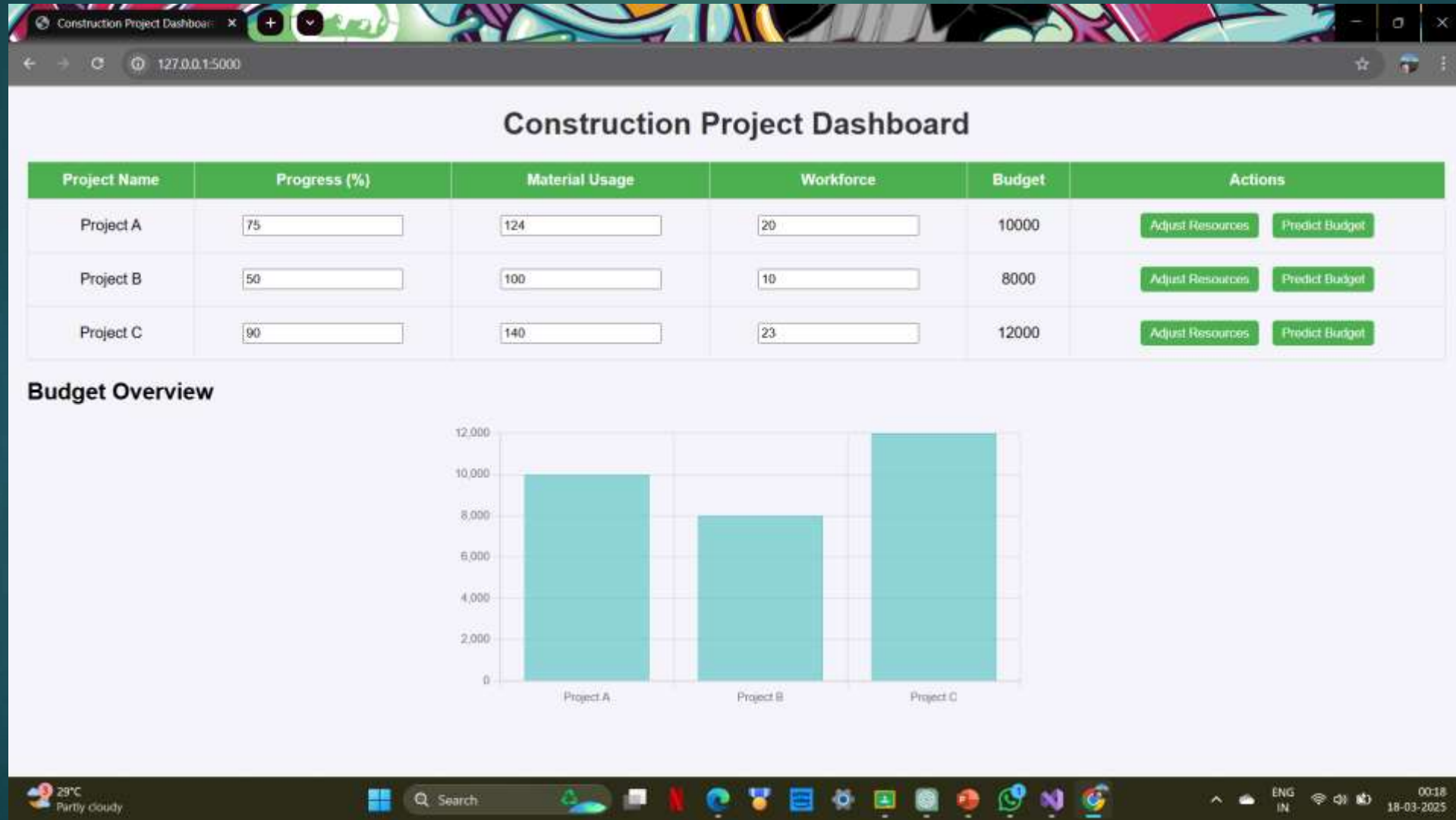
OUTPUT

ADJUSTMENT

Construction Project Dashboard

Project Name	Progress (%)	Material Usage	Workforce	Budget	Actions	
Project A	<input type="text" value="75"/>	<input type="text" value="124"/>	<input type="text" value="20"/>	10000	<button>Adjust Resources</button>	<button>Predict Budget</button>
Project B	<input type="text" value="50"/>	<input type="text" value="100"/>	<input type="text" value="10"/>	8000	<button>Adjust Resources</button>	<button>Predict Budget</button>
Project C	<input type="text" value="90"/>	<input type="text" value="140"/>	<input type="text" value="23"/>	12000	<button>Adjust Resources</button>	<button>Predict Budget</button>

OUTPUT FINAL



Reflection on Learning and Development

- **Learning:**

Flask, REST API, Chart.js, and real-time updates.

- **Challenges:**

Chart re-rendering and API sync.

- **Skills:**

Backend and frontend integration, data visualization.

- **Industry Insight:**

Importance of real-time MIS for large-scale projects.

Conclusion

- **Findings:**

Real-time MIS improved cost tracking and project efficiency.

- **Importance:**

Enhanced decision-making for project managers.

- **Future:**

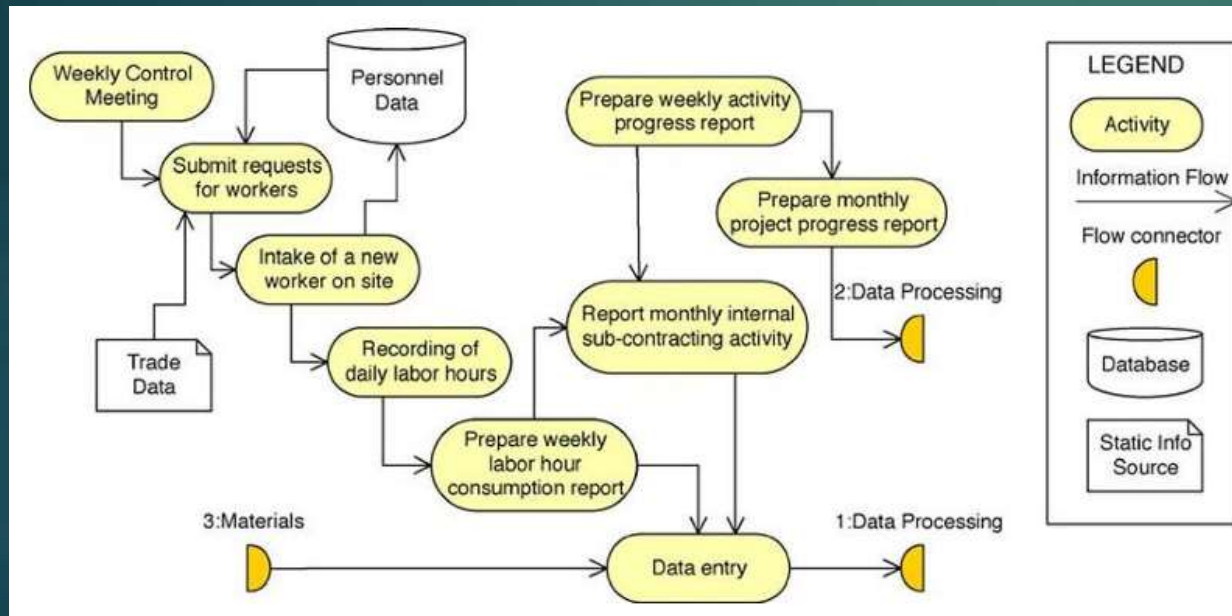
Add AI-based predictions and automated resource adjustment.

References

- Ceballos, C., Kolozhvari, A. A., Dolbilov, A. G., Semenov, R. N., Tsapulina, E. A., Rodriguez, A., ... & Murin, Y. A. (2023). Construction management information system at JINR. *Physics of Particles and Nuclei Letters*, 20(5), 981-987.
- Lee, S. K., & Yu, J. H. (2012). Success model of project management information system in construction. *Automation in construction*, 25, 82-93.
- Scherer, R. J., & Schapke, S. E. (2011). A distributed multi-model-based management information system for simulation and decision-making on construction projects. *Advanced Engineering Informatics*, 25(4), 582-599.

Appendices

System architecture diagram



Data Flow Diagram

