DOI:10.5937/IIZS25306J

A VISUAL APPROACH TO PROJECT MANAGEMENT USING REACT FLOW

Nikola Jovanov[1], Ivan Palinkaš[1], Eleonora Brtka[1], Vesna Makitan[1], Ema Brtka[2]
[1]University of Novi Sad, Technical Faculty "Mihajlo Pupin", Zrenjanin, Serbia
[2]University of Novi Sad, Faculty of Economics in Subotica, Serbia
e-mail: nikola.jovanov@uns.ac.rs, ivan.palinkas@uns.ac.rs, eleonora.brtka@uns.ac.rs, vesna.makitan@uns.ac.rs, brtka.ema777@gmail.com

Abstract: This paper presents the development of a graph-based project management tool that provides a visual and interactive representation of tasks, sections, and dependencies. The system is implemented using React Flow and custom components, enabling users to define project activities, assign responsibilities, and dynamically adjust dependencies. Unlike traditional approaches that rely on static charts, this tool emphasizes modularity and real-time interaction, offering visualization of individual tasks as well as aggregated sections, supported by forms for data entry. The implementation details include node structures for tasks and sections, edge customization for highlighting dependencies, and aggregation logic for calculating section durations. The solution demonstrates how modern web technologies can support intuitive project visualization and offers potential directions for extending the system with advanced features such as resource allocation and progress tracking.
Key words: Project management, React, React-Flow

INTRODUCTION

Project management is a critical aspect of both industrial and educational contexts, requiring careful planning, coordination, and monitoring of tasks and resources [1]. Despite the availability of numerous commercial and open-source tools, many existing solutions are either too complex for small teams or lack clear visualization of project flows and dependencies [2]. This can make understanding the overall project structure challenging, particularly for educational or research purposes.

The main objective of this work is to develop an interactive, graph-based project management tool that combines simplicity of use with basic project planning functionalities. The proposed solution leverages a modular design, representing tasks and sections as individual components, with dependencies visualized through interactive connections. By providing a clear overview of project structure and enabling dynamic modifications, the tool aims to support both learning and experimentation in project management scenarios.

In this paper, existing solutions for project management and visualization are reviewed, highlighting their strengths and limitations. Based on these observations, the motivation for developing a custom tool is outlined. The following sections present the user interface elements of the system, as well as key parts of the code implementation that enable interactive project planning and visualization.

As an illustrative example, the system is applied to a project involving the installation of solar panels, providing a concrete demonstration of the tool's capabilities.

PREVIOUS WORK

In the field of project management, a wide range of commercial and open-source tools are used in both industrial and educational contexts. Traditional software packages, such as Microsoft Project [3] and Primavera [4], represent industry standards and provide detailed planning, resource tracking, and management of dependencies among activities [2]. However, these tools often require significant training and can be overly complex for small teams or educational purposes [2].

With the rise of agile methodologies and collaborative work, tools like Jira [5], Trello [6], and Asana [7] have emerged, facilitating task tracking and team collaboration. While these tools

offer flexible and visually clear representations, their primary focus is on task management, and the visualization of overall project flows and dependencies remains limited.
In parallel, visualization tools such as Draw.io [8], Miro [9], and Lucidchart [10] enable users to create diagrams and graphs easily. These tools provide flexibility in representing processes, but they do not include project management logic, such as calculating task durations, critical paths, or resource allocation.
Additionally, Business Process Model and Notation (BPMN) tools, such as Bizagi [11] and Camunda [12], focus on modeling business processes. They allow detailed visualization of activity flows and dependencies but are not primarily intended for project management in educational or research contexts.
Based on the analysis of existing solutions, there is a clear opportunity for developing a tool that combines the simplicity of visual representation with basic project management functionalities. This concept forms the basis of the proposed software solution, which leverages the React Flow library to provide interactive visualization of project flows while maintaining minimal learning effort and maximum clarity.

THEORETICAL BACKGROUND

The theoretical background section provides the foundation for understanding the key concepts and technologies used in this work. It introduces essential principles of project management, including task organization, dependencies, and project planning techniques. Furthermore, it presents the technologies employed for implementing the proposed solution, with a focus on React and the React Flow library. Finally, this section outlines how these concepts are applied to the illustrative project example, highlighting the benefits of interactive and visual representation for project management.

Project Management

Project management is a structured approach to planning, organizing, and controlling resources to achieve specific objectives within a defined timeframe [1, 13]. A project is typically composed of multiple activities or tasks, each with a defined start and end point, dependencies on other tasks, and associated resources [1,13]. Key concepts include task sequencing, critical path, and Work Breakdown Structure (WBS), which help in organizing complex projects into manageable components [1, 13].
A particularly important aspect of project planning involves network-based methods, such as Critical Path Method (CPM) and Program Evaluation and Review Technique (PERT) [1, 14, 15, 16]. These methods use directed graphs to represent activities and their dependencies, making it possible to identify the longest sequence of dependent tasks that determines the overall project duration [1, 16]. Recognizing the critical path helps project managers focus on activities that directly impact project completion time, while also enabling the identification of time reserves (slack) in non-critical tasks [1, 16].
Visualizing projects through such network diagrams plays a crucial role in understanding task relationships and identifying potential bottlenecks. While Gantt charts primarily emphasize the timing and duration of tasks [17], network-based visualizations highlight task dependencies and the logical flow of activities [16], making alternative paths easier to identify. This makes them particularly valuable for both industrial and educational contexts, where clarity of relationships between tasks is essential. Studies have shown that interactive and visual tools based on network concepts improve comprehension and facilitate collaboration, particularly in educational and research environments [18, 19, 20].
By combining these principles with modern visualization techniques, it becomes possible to create a tool that is both easy to use and effective in conveying the structure and flow of projects. The proposed solution builds on these ideas by offering an interactive environment where project tasks are represented as nodes, dependencies as edges, and users can dynamically explore project structure in an intuitive manner.

React

React is a widely used JavaScript library for building interactive user interfaces, particularly in web applications [21]. Its component-based architecture allows developers to create modular and reusable elements, which can be combined to construct complex applications [21, 22].
In the context of this work, React provides the foundation for building an interactive project management tool. By leveraging components, it is possible to represent each task, user, or connection as an independent element, which can be easily modified or extended. This modularity enhances maintainability and simplifies the implementation of dynamic features, such as real-time updates of project status, drag-and-drop functionality, and interactive visualization of task dependencies.
Furthermore, React's integration with modern libraries enables seamless incorporation of visualization frameworks, such as React Flow, providing the necessary tools to represent project workflows in a clear and interactive manner. This combination allows users to intuitively understand project structure and manage tasks effectively, without being overwhelmed by the complexity often found in traditional project management software.

React Flow

React Flow is a JavaScript library built on top of React that enables the creation of interactive, node-based diagrams [23]. It allows developers to represent complex relationships visually, using nodes to symbolize entities or tasks, and edges to depict dependencies or flows between them.
In the context of project management, React Flow provides an intuitive way to visualize the structure of a project. Each task or activity can be represented as a node, with edges indicating dependencies such as "predecessor-successor" relationships. Users can interact with the diagram through drag-and-drop functionality, dynamically rearranging tasks or modifying connections, which enhances understanding and decision-making [23].
One of the key advantages of React Flow is its modularity and extensibility. Custom node types, styles, and interactive behaviors can be easily integrated, allowing for project-specific adaptations, such as color-coding tasks by priority, highlighting critical paths, or dynamically calculating project duration based on dependencies [23]. This flexibility makes it particularly suitable for educational and research applications, where simplicity and clarity are crucial, while still providing an accurate representation of project workflows.
By combining React's component-based architecture with React Flow's graph visualization capabilities, it becomes possible to create a project management tool that is both highly interactive and visually intuitive, making complex project structures easier to comprehend and manage.
The following section presents the implementation of these concepts in an interactive project management application.

IMPLEMENTATION

The application is structured as a graph-based project management tool. Tasks and sections are represented as nodes, with dependencies modeled as edges. Each task stores attributes such as duration and assigned team, enabling aggregation and visualization. The system was implemented using React Flow for interactive graphs, along with custom components for data entry and role assignment.

User Interface

The application interface is designed to support both visualization and project data management. Its central element is the project graph, where sections and tasks are represented as nodes, and their dependencies as edges. Tasks are visually distinguished by

color according to the assigned name or role, while section nodes provide a higher-level grouping of activities. This layout enables a clear overview of the project structure and makes navigation smoother. An example of these features is shown in Figure 1.

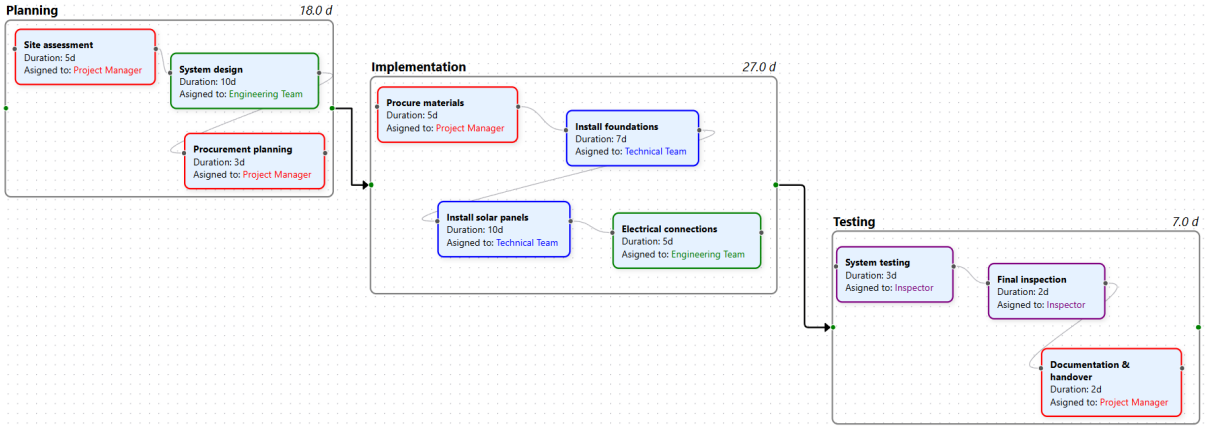

Fig. 1. *Example of a project plan*

The figure illustrates the complete project as represented in the application, including the Planning, Implementation, and Testing sections. Each section contains its respective tasks, clearly indicating task durations and assigned teams.
A form is provided for adding new tasks to the project. The user can specify the task name, duration, time unit, and the team member responsible for its execution. This ensures that all relevant details are captured and visualized appropriately in the project layout. An example of this form is shown in Figure 2.

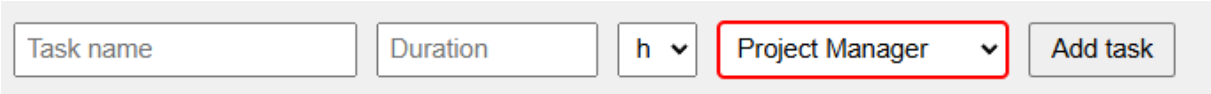

Fig. 2. *Form for adding tasks*

Another form allows the addition of new team members. The user can enter a name and choose a color from drop-down menu for visual identification in the project. This helps to organize and distinguish between contributors within the project view. It is a rather simple form, and example is shown in Figure 3.

Fig. 3. *Form for adding asignees*

Code Implementation

The project's interactive functionality is realized through modular React components, each responsible for representing specific elements of the project. Task nodes, section nodes, and form components work together to provide a dynamic and intuitive interface for project visualization and management.

The TaskNode component represents an individual task within the project. It encapsulates all relevant information about a task, including its name, duration, assigned team member, and visual style, such as color and border. Task nodes also provide handles for connecting to other

nodes, enabling the representation of task dependencies through edges. This modular design allows each task to be self-contained, making it straightforward to modify or extend task properties without affecting other components. Listing 1 illustrates the structure of a TaskNode component, showing how task details are displayed and how handles are included for connecting tasks.

```
<div>
    <div>
        <strong>{name}</strong>
    </div>
    <div>
        Duration: {duration}{unit}
    </div>
    <div>
        Asigned to: <span style={{ color }}>{assignee}</span>
    </div>
    <Handle type="target" position={Position.Left} style={{ background: "#555" }} />
    <Handle type="source" position={Position.Right} style={{ background: "#555" }} />
</div>
```

Listing 1. *Source code for TaskNode component*

Dependencies between tasks are represented using edges, which connect source and target handles on TaskNodes. These edges indicate the order of task execution and help visualize the flow of the project. Each edge can be customized with visual attributes, such as color, thickness, and arrow style, to improve clarity and distinguish between different types of dependencies.

Listing 2 shows an example of an edge connecting two sections.

```
{
  id: 'section-2-section-3',
  source: 'section-2',
  target: 'section-3',
  markerEnd: { type: MarkerType.ArrowClosed }
}
```

Listing 2. *Structure of an edge*

For section-to-section edges, the type is set to "smoothstep" to create a visually appealing curved connection, and the stroke width is increased to make these edges more prominent than standard task-to-task connections.

The SectionNode component represents a group of related tasks within the project. Each section node aggregates the tasks it contains, calculating the total duration based on the durations of its child tasks. Visually, section nodes provide a higher-level overview, making it easier to understand the structure and timeline of the project. Although the visual style is similar to task nodes, section nodes summarize information rather than focusing on individual task details.

To determine the total duration of a section, the SectionNode component iterates over all contained TaskNode components. It reads each task's duration and unit of time, converting them to a standard unit before summing. This ensures that tasks specified in different units, such as hours or days, are correctly accounted for in the section's total duration. By doing so, the SectionNode provides an accurate summary of the workload for the section, without the need to inspect individual tasks.

The function that checks whether a task belongs to a section evaluates the positions and dimensions of both the task and section nodes. This ensures that each section aggregates only the tasks that are visually contained within its bounds, and correctly calculates total duration with unit conversion. Described function is shown in listing 3.

```
const isTaskInsideSection = (taskNode, sectionNode) => {
  const [sx, sy] = [sectionNode.position.x, sectionNode.position.y];
  const [sw, sh] = [sectionNode.style?.width || 500, sectionNode.style?.height || 300];

  const [tx, ty] = [taskNode.position.x, taskNode.position.y];
  const [tw, th] = [taskNode.style?.width || 150, taskNode.style?.height || 50];

  return tx >= sx && ty >= sy && tx + tw <= sx + sw && ty + th <= sy + sh;
};
```

Listing 3. *Function to determine if a task belongs to a section*

Form components interact directly with the project graph, allowing users to add new tasks and team members. When a task or assignee is added through the corresponding form, the graph updates dynamically, reflecting the changes in real time. This integration ensures that all project data remains consistent and immediately visible, providing an intuitive link between data entry and project visualization.
This modular approach ensures that the project's structure can be easily extended and modified, while keeping the codebase organized and maintainable.

CONCLUSION AND FURTHER WORK

The developed project management tool provides an interactive and visual approach to planning and monitoring tasks within a project. By combining React with React Flow, the application allows users to clearly see task dependencies, section groupings, and overall project structure, while maintaining an intuitive interface for adding tasks and team members. The tool was demonstrated on the example of a solar panel installation project, where it enabled a clear and structured representation of all tasks and sections, making the overall plan more transparent and easy to follow.
Further improvements could include the integration of resource management, automatic calculation of critical paths, and support for additional task attributes, such as priorities or deadlines. Moreover, enhanced visualization features, such as filtering, or dynamic highlighting of critical tasks, could improve usability for larger projects. In addition, extending the system to support financial aspects such as costs and expenses would provide a more comprehensive project overview. Expanding the tool to support collaborative multi-user environments and real-time updates would further align it with modern project management practices and research-oriented use cases.

REFERENCES

[1] Kezner, Harold, "Project Management: A Systems Approach to Planning, Scheduling, and Controlling" (2025), Wiley, Hoboken
[2] Magrea, Romeo, Magrea, Camelia, "Open Source Approach to Project Management Tools" (2011), Informatica Economica vol. 15
[3] Microsoft Project, URL: https://learn.microsoft.com/en-us/project/
[4] Primavera, URL: https://www.oracle.com/construction-engineering/primavera-p6/
[5] Jira, URL: https://www.atlassian.com/software/jira
[6] Trello, URL: https://trello.com/
[7] Asana, URL: https://asana.com/uses/project-management
[8] Draw.io, URL: https://www.drawio.com
[9] Miro, URL: https://miro.com
[10] Lucidchart, URL: https://www.lucidchart.com/
[11] Bizagi, URL: https://www.bizagi.com/
[12] Camunda, URL: https://camunda.com/
[13] A Guide to the PROJECT MANAGEMENT BODY OF KNOWLEDGE (PMBOK Guide) (2017), Project Management Institute, Inc.

[14] Bishnoi, Nisha, "Critical Path Method (CPM): A Coordinating Tool (2018), Managt Sci Tech, International Research Journal of Management Science & Technology
[15] Bianconi, Francesco, "PERT Diagrams" (2024), Data and Process Visualisation for Graphic Communication. Springer, Cham.
[16] Agyei, Wallace, "Project Planning And Scheduling Using PERT And CPM Techniques With Linear Programming: Case Study" (2015), International Journal of Scientific & Technology Reasearch
[17] K, K, Ramachandran, K, K, Karthick, "Gantt Chart: An Important Tool of Management" (2019), International Journal of Innovative Technology and Exploring Engineering
[18] Zoss, Angela, Maltese, Adam, Uzzo, Stephen, Borner, Katy, "Network visualisation literacy: Novel approach to measurement and instruction" (2018), Network Science In Education. Springer, Cham.
[19] Kuosa, Kirsi, Distante, Damiano, Tervakari, Anne, Cerulo, Luigi, Fernandez, Alejandro, Koro, Juho, Kailanto, Meri, "Interactive Visualization Tools to Improve Learning and Teaching in Online Learning Environments" (2016), International Journal of Distance Education Technologies
[20] Brugliera, Patrick, "The Effectiveness of Digital Learning Platforms in Enchancing Student Engagement and Academic Performance (2024), Journal of Education, Humanities and Social Research
[21] Gackenheimer, Cory, "Introduction to React" (2015)
[22] Osmani, Andy, "Learning JavaScript Design Patterns", O'Reilly Media (2023 – Second Edition)
[23] Official React Flow documentation, URL: https://reactflow.dev/