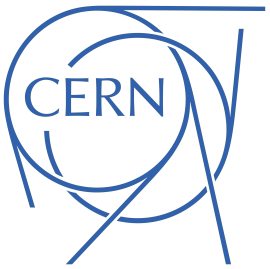


CERN Summer Student Project Report

IMPACT system improvement by implementing Calendar Page for the existing Activities

Wojciech Michaluk
supervised by
Giovanni Chierico *and* Mateusz Jan Kubicki

July 28, 2025

Abstract

During the CERN Summer Student Programme 2025, I worked with the EN-IM-CAP unit to enhance IMPACT, CERN's platform for managing permit-to-work processes. My main contribution was developing a Calendar page featuring an interactive Gantt chart to visualize safety and coordination activities. Built with React and the "@dhx/gantt" library, this component supports zooming, date range navigation, and activity filtering by type.

Through this project, I advanced my skills in React, TypeScript, Java (Spring Boot), and DevOps tools such as GitLab and Docker. Working with senior developers in an agile setting improved my ability to write clean, maintainable code.

Collaborating with teams involved in radiation safety deepened my understanding of large-scale scientific infrastructure and the role of software in ensuring operational safety. This experience strengthened both my technical abilities and my appreciation for software's impact on research environments like CERN.

Contents

1 Introduction

1.1 My role and objectives at CERN

My group and department

I had the opportunity to join the Information Management group on 2 June 2025. The group provides applications and support for engineering information management across the entire Organization and its various projects [4]. It specializes in a range of technologies, including mechanical CAD tools (such as CATIA), Product Lifecycle Management systems (SmarTeam / Aras), the Engineering Data Management Service (EDMS), as well as the Enterprise Asset Management platform (EAM).

The overall goal of the group is to ensure data traceability throughout the full lifecycle of the accelerator complex and technical installations, along with supporting change management processes.

My team

I was a member of the Engineering Department, within the CAP unit, led by Giovanni Chierico. While I collaborated with the entire team, my work was primarily coordinated by Mateusz Jan Kubicki. The team focuses on developing full-stack web applications.

We followed the SCRUM methodology. I participated in team meetings whenever my schedule allowed, particularly when I was free from student lectures. The most important meetings were the daily stand-ups, during which we reviewed the previous day's work and discussed plans for the upcoming day. Additionally, we conducted biweekly sprint planning sessions to estimate our capacity and evaluate the current workload.

These meetings were especially valuable as they provided a platform for collaborative problem-solving, with all team members actively involved in the planning process. Although I was a junior member of the team, I felt that my input was valued and that I could openly share my perspective.

1.2 IMPACT system and its purpose

System descripiton

The primary objective of my team is to migrate the IMPACT system to a more modern architecture and visual style that includes support for mobile devices. IMPACT, which stands for **Intervention Management Planning and Coordination Tool**, is used to manage the permit-to-work processes. The system was launched in 2011 and addresses both safety-related and coordination-related aspects[3].

IMPACT operates through standardized activities that are declared across all CERN installations. It is used to obtain access to specific locations and to coordinate personnel tasks throughout the working day. Over time, it has become a crucial system for the Organization, with approximately 15,000 requests created each year. As such, the team I was part of is responsible for maintaining and improving one of the Organization's mission-critical systems [3].

High-level architecture [2]

The architecture of the system follows a conventional structure in terms of the technologies used. The backend is developed in Java, while data pipelines are managed through executable process graphs defined in Camunda, which orchestrates data flow to the database. The frontend is implemented using React and TypeScript.

IMPACT user interface

The requests within the system, referred to as *Activities*, are categorized into several types, each serving a distinct purpose and associated with a specific form [2]:

- **DIMRs** – Records of Intervention in a Radioactive Environment.
- **WDPs** – Work Dose Planning documents used to estimate radiation exposure for planned tasks, prepared by the activity responsible in collaboration with radiation protection experts.

- **VICs** – Joint Inspection Visits during which safety stakeholders meet on-site to validate and adjust safety measures.
- **Lockouts** – Activities related to energy isolation and lockout procedures.
- **Fire Permits** – Authorizations ensuring that adequate fire prevention measures are in place before hot work is performed.
- **IS37s** – Specific intervention-related documents (internal designation).
- **Cotes de Coupure** – Activities involving technical cutting points or intervention zones.
- **Tasks** – A mechanism used by the application to delegate operations to users, such as signatures or data completion. They are widely applied across all safety documents and Activities.

Each type of activity is associated with a dedicated form containing fields tailored to its specific requirements.

2 My final project

2.1 Calendar

Description of the feature

For my final project, I was assigned to develop a calendar view for the existing access requests in the system. This feature was requested by a current user of IMPACT. Consequently, we organized a meeting with them to gather the requirements. There was a need to visualize activities in a calendar format similar to a Gantt chart. This would allow users to identify specific tasks, view their durations, and check their current statuses (e.g., approved or cancelled). Additionally, users would be able to organize the workflow within their section by assigning specific individuals to activities during appropriate time slots. The calendar was intended to be interactive, enabling users to easily navigate through the schedule—by week or by year—using month-based traversal.

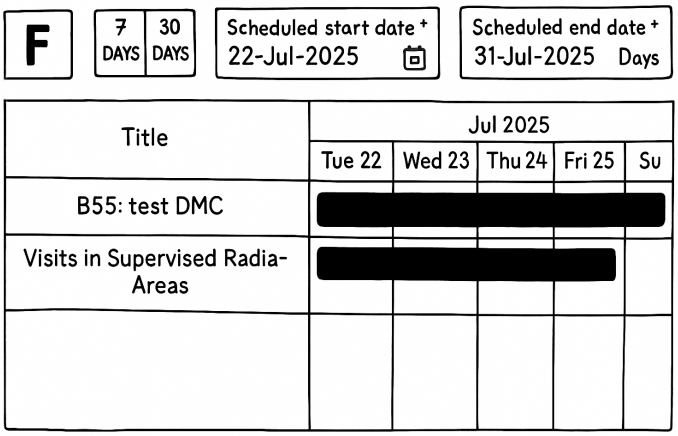


Figure 1: High-level Calendar User Interface

Used Technologies

The page was implemented primarily on the frontend side of the project, as there was no need to modify backend data processing. Research was conducted to identify the most suitable React library for the task. The `"@dhx/gantt"`[1] (version 9.0.13) library was selected due to its support for fully customizable and functional Gantt charts. An additional advantage of this library was that another unit within our group already possessed the necessary license for its use.

During the project, I worked with Java (Spring Boot) and React. I extensively used Git for version control, with the repository hosted on GitLab. All of my code was reviewed by senior team members, and I also had the opportunity to review code written by other team members. Emphasis was placed on writing clean, self-descriptive code. As part of this approach, we avoided adding comments in favor of using meaningful naming conventions.

I used IntelliJ IDEA and Visual Studio Code as my integrated development environments (IDEs), and I also gained some experience with Docker containers. Participating in this project allowed me to further develop my programming skills and deepen my understanding of Git and its operations.

2.2 Implementation

The feature was designed as a separate page within the user interface. This decision enabled me to work in parallel with the rest of the team, ensuring that no other releases interfered with my development process. The architecture of the feature consists of five distinct files:

```
calendar/
├─ Calendar.tsx
├─ CalendarCard.tsx
├─ ColorMappings.tsx
├─ Toolbar.tsx
└─ Types.ts
```

CalendarCard.tsx

Serves as the wrapper for the entire feature and acts as the parent component for the rest of the project. It is accessible via the URL: `/calendar`. This component uses standard JSX syntax along with Material UI components. It contains the logic for sending requests to the backend with the hook `useInfiniteQuery`. It defines all constants used within the calendar, and handles the logic for passing filters to the backend in order to retrieve and display relevant activities. It also connects the `Toolbar` with the `Calendar`.

- **Backend Query Handling** – The component uses the `useInfiniteQuery` hook to fetch paginated data from the backend, applying active filters and transforming the response into Gantt chart-compatible tasks.

```
const { data } = useInfiniteQuery({
  queryKey: ['activities', JSON.stringify(filters)],
  queryFn: async ({ pageParam = 0 }) => {
    const res = await getActivities({ filter: filters, page: pageParam })
        ;
    const rawData = res?.content ?? [];
    const tasks = rawData.map((activity) => ({
      id: activity.id,
      title: activity.title ?? 'Task',
      start_date: parseISO(activity.scheduledStart),
      end_date: parseISO(activity.scheduledEnd),
      responsible: extractResponsible(activity),
      section: extractSection(activity),
      color: getSectionColor(section),
    }));
    return { tasks, page: res.pageable.pageNumber, totalPages: res.
        totalPages };
  },
  getNextPageParam: (lastPage) =>
    lastPage.page + 1 < lastPage.totalPages ? lastPage.page + 1 :
        undefined,
});
```

Listing 1: Backend query using useInfiniteQuery

- **Effect Hook for Data Aggregation** – The `useEffect` hook flattens and stores the paginated query results.

```
useEffect(() => {
  if (data?.pages) {
    const allData = data.pages.flatMap((page) => page.tasks);
    setActivities({ data: allData, links: [] });
  }
}, [data]);
```

Listing 2: Flatten and store fetched data

- **Saved Search and Filter Handling** – Functions manage filter state in local storage and allow resetting to default configurations.

```
const handleSavedSearchStore = (id) => {
  setLocalStorageState({ ...localStorageState, savedFilterId: id });
};

const handleResetFilters = () => {
  handleSavedSearchStore(undefined);
  filterForm.reset({ filters: [] });
  queryClient.removeQueries(['activities']);
  handleSearch();
};
```

Listing 3: Store and reset filters

- **Date Filter Updates** – When users adjust date filters, the corresponding logic updates the query parameters and resets the pagination.

```
const handleStartDateChange = (date) => {
  setStart(date);
  setOrUpdateFilter({
    field: 'scheduledEnd',
    operator: 'ON_OR_AFTER',
    value: date,
    type: 'DATE',
  });
  setCurrentPage(0);
};

const handleEndDateChange = (date) => {
  setEnd(date);
  setOrUpdateFilter({
    field: 'scheduledStart',
    operator: 'ON_OR_BEFORE',
    value: date,
    type: 'DATE',
  });
  setCurrentPage(0);
};
```

Listing 4: Start and end date filter updates

Toolbar.tsx

The `Toolbar.tsx` component manages user controls for setting the visible range and zoom level of the Gantt chart. It includes two date pickers for specifying start and end dates, with validation logic to prevent incorrect ranges. It also provides a dropdown menu to choose the zoom level (e.g., day, month). The layout is built with Material UI components and styled using flexbox properties.

```
<Box display="flex" alignItems="center" gap={2} mb={2} mt={2}>
  <DatePicker
    label={intl.formatMessage({ id: 'calendar.field.scheduledStart' })}
    value={startDate}
    onChange={(d) => d && onStartChange(d)}
    shouldDisableDate={(d) => !!d && d.getTime() > endDate.getTime()}
    format={dateFormat}
    slotProps={{ textField: { size: 'small', required: true } }}
  />
  <DatePicker
    label={intl.formatMessage({ id: 'calendar.field.scheduledEnd' })}
    value={endDate}
    onChange={(d) => d && onEndChange(d)}
    shouldDisableDate={(d) => !!d && d.getTime() < startDate.getTime()}
    format={dateFormat}
    slotProps={{ textField: { size: 'small', required: true } }}
  />
  <FormControl size="small" sx={{ minWidth: 120 }}>
    <InputLabel>{intl.formatMessage({ id: 'calendar.field.zoom' })}</
       InputLabel>
    <Select
      value={zoom}
      label={intl.formatMessage({ id: 'calendar.field.zoom' })}
      onChange={(e) => onZoomChange(e.target.value)}
    >
      {zoomLevels.map((level) => {
        const zoomLabelKey = `calendar.field.zoom.${level.toLowerCase()}`;
        return (
          <MenuItem key={level} value={level}>
            {intl.formatMessage({ id: zoomLabelKey })}
          </MenuItem>
        );
      })}
    </Select>
  </FormControl>
</Box>
```

Listing 5: Toolbar with date pickers and zoom selector

Calendar.tsx

The `Calendar.tsx` component renders the Gantt chart using the `@dhx/gantt`[1] library in a non-React manner for greater flexibility. It initializes and configures the chart, defines custom tooltips, styles task details, and enables dynamic zooming. User interaction is enhanced by allowing CTRL-key-based tooltip persistence and task click redirection using React Router. It also listens for changes in task data, zoom level, and date range to re-render the chart accordingly.

```
const Calendar: React.FC<CalendarProps> = ({ tasks, zoom, startDate, endDate,
    dateFormat }) => {
  const container = useRef(null);
  const navigate = useNavigate();
  const formatDate = gantt.date.date_to_str(dateFormat);

  useEffect(() => {
    gantt.config.readonly = true;
    gantt.plugins({ tooltip: true });

    gantt.templates.tooltip_text = (start, end, task) => `
      <div class="gantt-tooltip">
        <b>ID:</b> ${task.id}<br/>
        <b>Title:</b> ${task.title}<br/>
        <b>Start:</b> ${formatDate(task.start_date)}<br/>
```

```
        <b>End:</b> ${formatDate(task.end_date)}
      </div>`;

    gantt.templates.task_text = (start, end, task) =>
      `<span class="gantt-task-details">${task.id}</span>`;

    gantt.ext.zoom.init({
      levels: [
        {
          name: 'Days',
          scales: [
            { unit: 'week', step: 1, format: '%M %Y' },
            { unit: 'day', step: 1, format: '%D %d' },
          ],
        },
        {
          name: 'Months',
          scales: [
            { unit: 'month', step: 1, format: '%F %Y' },
            {
              unit: 'week', step: 1,
              format: (date) => {
                const weekStart = gantt.date.week_start(new Date(date));
                const weekEnd = gantt.date.add(weekStart, 6, 'day');
                const f = gantt.date.date_to_str('%d');
                return `${f(weekStart)} - ${f(weekEnd)}`;
              },
            },
          ],
        },
      ],
    });

    gantt.config.start_date = new Date(startDate);
    gantt.config.end_date = new Date(endDate);
    gantt.init(container.current);
    gantt.parse(tasks);

    return () => {
      gantt.clearAll();
      gantt.ext.tooltips.tooltip.hide();
    };
  }, []);

  useEffect(() => {
    gantt.ext.zoom.setLevel(zoom);
    gantt.render();
  }, [zoom]);

  useEffect(() => {
    gantt.clearAll();
    gantt.parse(tasks);
  }, [tasks]);

  return <div ref={container} style={{ width: '100%', height: 'calc(85vh - 80
      px)' }} />;
};
```

Listing 6: Core logic of Calendar.tsx

ColorMappings.tsx

The `ColorMappings.tsx` file defines a mapping between activity types and their associated colors. These mappings are used to visually distinguish different activities in the calendar view, making it easier for users to identify and interpret them at a glance. Colors are grouped by system-defined, technical, and other categories.

```
const sectionColors: Record<string, string> = {
  // System Colors
  DEFAULT: 'rgb(61,185,211)',
  SUMMARY: 'rgb(240,240,240)',
  SUMMARY_TEXT: 'rgb(69,69,69)',
  MARKED: 'rgb(255,0,0)',
  All: 'rgb(205,199,255)',

  // Technical (TE)
  TE-ABT: 'rgb(55,86,35)',
  TE-CRG: 'rgb(0,176,80)',
    ...
};
```

Listing 7: Color scheme for activity types

Types.ts

The `Types.ts` file defines reusable TypeScript types and enums used throughout the calendar component. This approach improves code maintenance, type safety, and clarity across the project. It includes definitions for zoom levels, date ranges, Gantt chart task but also props for main components.

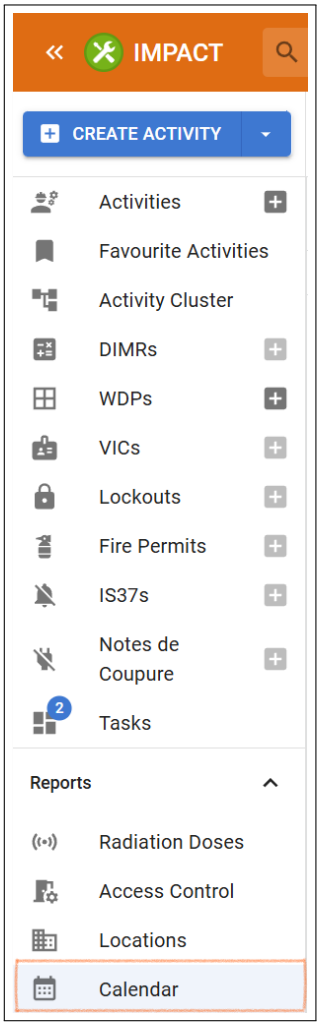
```
export enum ZoomLevel {
    Days = 'Days',
    Months = 'Months',
}

export interface GanttTask {
    id: string | number;
    title: string;
    description: string;
    start_date: Date;
    end_date: Date;
    duration: number;
    responsible: string;
    section: string;
    color: string;
}

export interface CalendarProps {
    tasks: { data: GanttTask[]; links: GanttLink[] };
    zoom: ZoomLevel;
    startDate: Date;
    endDate: Date;
    dateFormat: string;
}
```

Listing 8: Key type definitions used in the calendar

2.3 Final result



Calendar Feature. The Calendar feature is easily accessible from the side menu on the main IMPACT page. It is located at the bottom of the "Reports" section.

Figure 2: Accessing the Calendar page from the side menu

The Gantt chart on the Calendar page is limited to the date range selected in the toolbar. Activities are color-coded based on a palette provided by the users who requested the feature. A column displays the activity titles, and a tooltip with additional details appears when hovering over an activity.

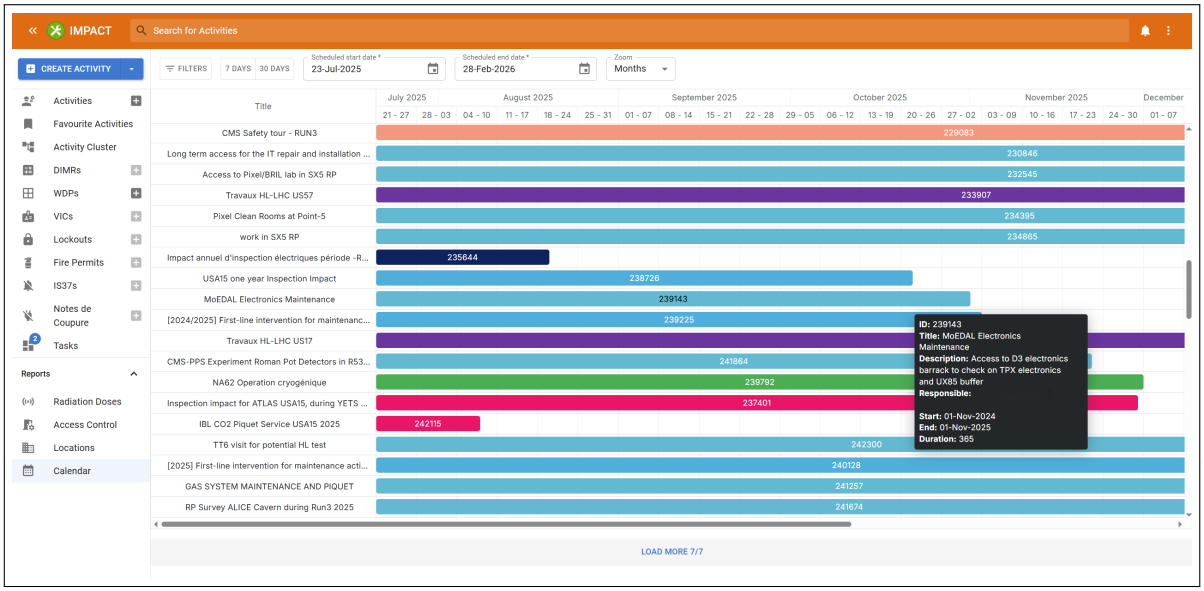


Figure 3: General view of the Calendar page in IMPACT

The toolbar includes selectable filters, predefined date ranges, date pickers, and a zoom level dropdown menu.

Figure 4: Toolbar for adjusting the Gantt chart view

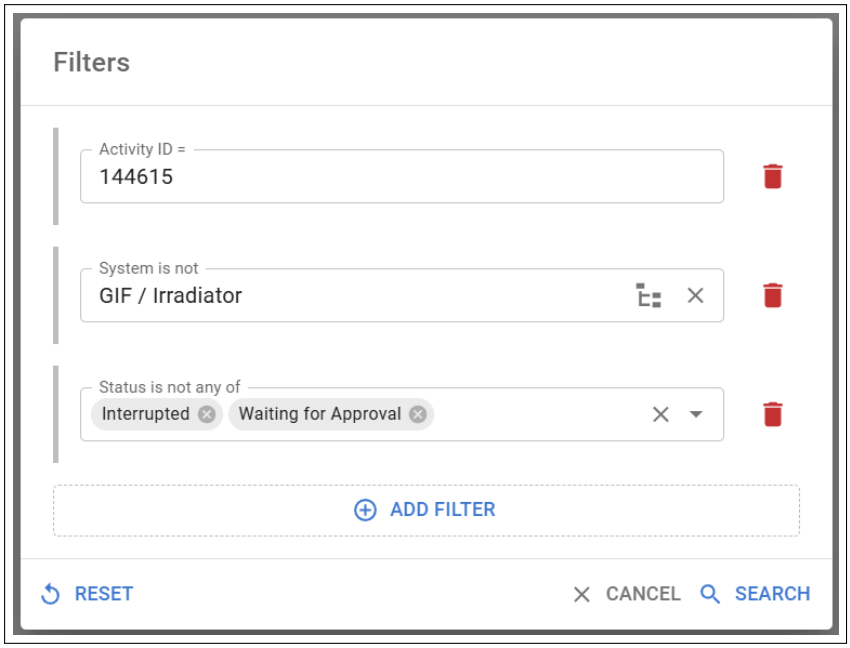


Figure 5: Advanced filtering options using multiple filters

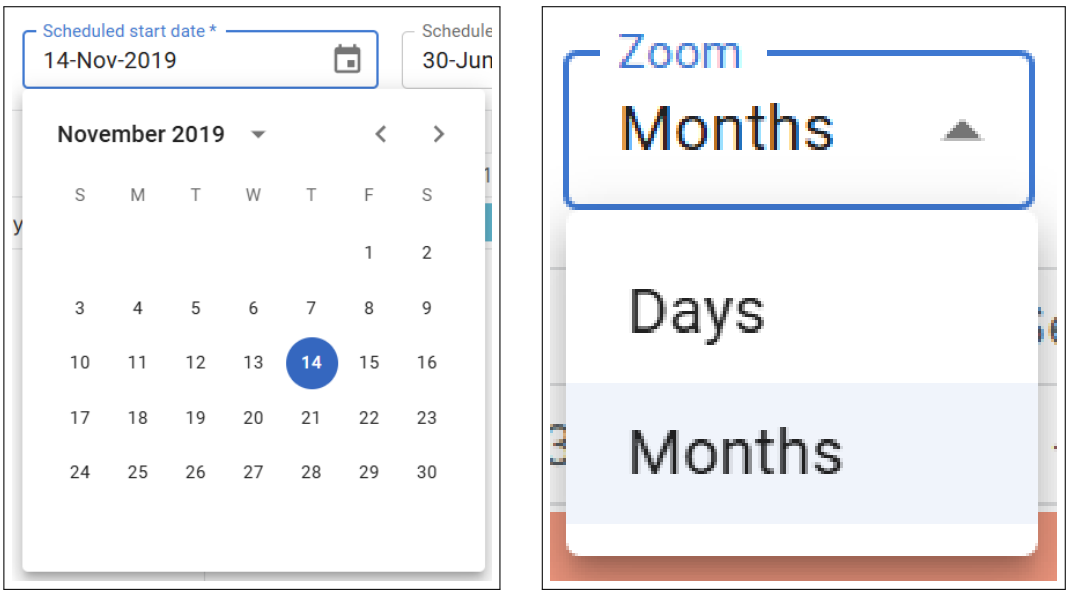


Figure 6: Adjusting the date range and zoom level

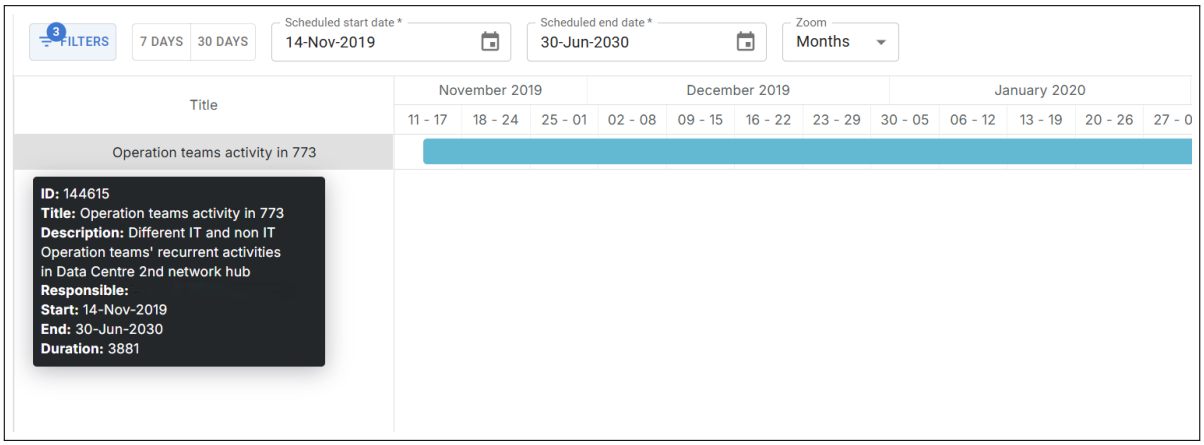


Figure 7: View of the Calendar page after applying filters

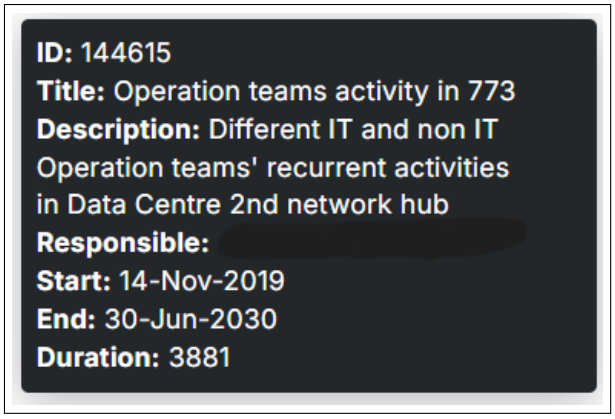


Figure 8: Tooltip displaying activity details on hover

The final result is a user-friendly, read-only Gantt chart embedded within the IMPACT application.

It provides a clear visual overview of existing activities, enhanced with color-coded entries, tooltips, and advanced filtering options. Users can navigate through predefined or custom date ranges and adjust the zoom level to focus on specific time frames. Although the chart itself is read-only, each activity is clickable and redirects the user to the corresponding task page, where the details can be edited. This solution offers an intuitive and efficient way to browse, review, and manage project activities.

3 Other tasks made for the team

Familiarizing with IMPACT

The IMPACT codebase is quite complex, so it took some time to become familiar with its structure. I began by reviewing the onboarding documentation. My first completed task involved correcting the styling of the questions in the activity forms. The issue was that, for activities in inactive states, some fields were not properly disabled, which gave users the false impression that they were still editable. Completing this task gave me a solid understanding of the frontend logic of the IMPACT system, which proved helpful later when I was implementing the Calendar page.

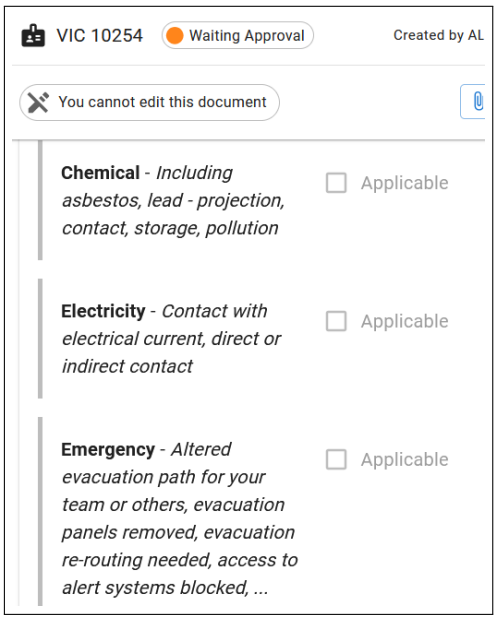


Figure 9: Despite the checkboxes being disabled, the text was not greyed out. This issue has been fixed

Global search

The global search functionality inside IMPACT was not working as expected, and I was tasked with resolving the issue. The root cause turned out to be an incorrect backend API endpoint being used in the request. In addition to fixing the technical issue, I improved the user interface of the search feature to make it more intuitive. The display of search results was enhanced by including key details such as status, title, ID, and owner of the activities. Clicking on a search result now correctly redirects the user to the corresponding activity page.

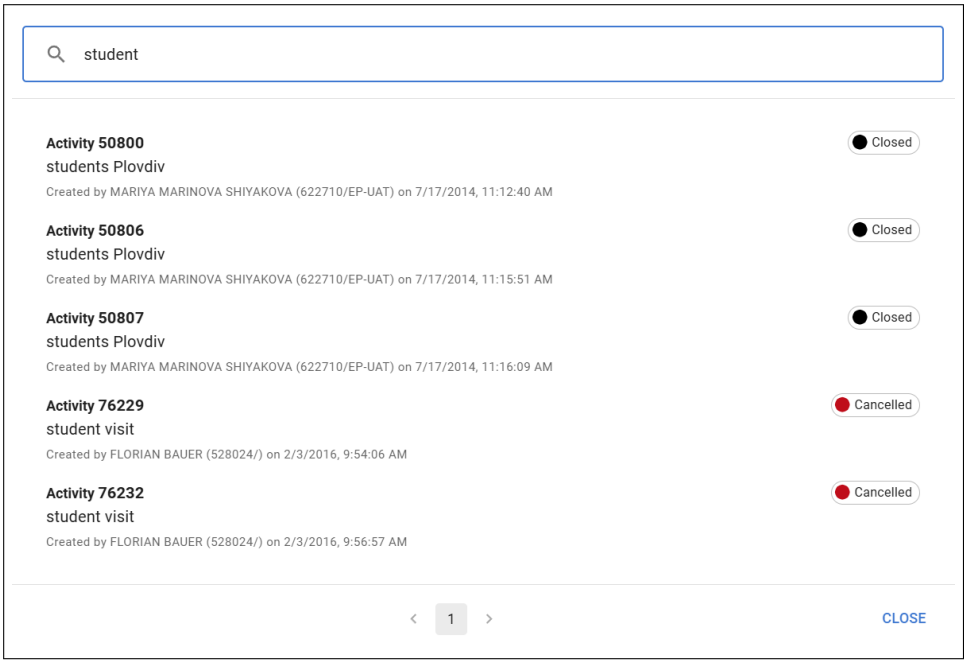


Figure 10: The redesigned global search interface in the IMPACT tool

4 My experience with the Summer Student project

Developer experience

Through the project, I deepened my skills in TypeScript, React, Java, and modern development tools such as GitLab CI/CD, Docker, and REST APIs. Regular code reviews and agile collaboration with experienced developers taught me how to write clean, maintainable code and deliver production-ready features. I also gained insight into how software supports safety-critical operations in complex environments like CERN.

Lectures [5]

The lectures I attended were extremely engaging, particularly those focused on particle physics. As a Data Science student, I had no prior in-depth knowledge of this subject. The material was presented in a clear and structured manner, making it accessible and easy to follow.

I especially appreciated the lectures on the Foundations of Statistics, which provided a valuable recap of core statistical concepts. Since statistics has always been one of my favourite parts of my university studies, I took the opportunity to revisit and reinforce the material.

The lectures concerning CERN's facilities helped me understand the institution's significant role in the modern scientific landscape and its contributions to the advancement of science. I am sincerely grateful to all the professors who delivered lectures to our group.

Campus visits

During the program, I had the opportunity to visit several scientific facilities at CERN. We participated in an organized tour of the ALICE experiment, where I learned about the collider and its significance. I also visited the antimatter factory, where antimatter particles are produced and experiments involving their properties are conducted.

Additionally, I toured the Data Center, which provided valuable insight into the vast amount of data generated and processed by CERN. I was impressed by its advanced infrastructure and the international collaboration that enables the use of data clusters worldwide.

Furthermore, I had the chance to visit the SM18 building, where particle beams are accelerated, and to observe the superconducting magnets firsthand. I also enjoyed the Science Gateway, which presents fundamental physics principles in an accessible and easy-to-understand manner.

Travelling

Undoubtedly, CERN is situated in a picturesque region. The Summer Student Program provided an opportunity not only to learn and connect with my working unit team but also to travel and meet with people from around the world growing in the international environment. During our trips to the mountains and bivouacs by Lake Geneva, I experienced unforgettable moments for which I am extremely grateful. I am sure that those memories will stay with me for a long time.

Figure 11: Sunflower field located near the CERN Swiss site

Figure 12: View of the Chamonix mountain range, situated close to Mont Blanc

References

[1] *dhtmlxGantt Library*. Accessed: 2025-07-23. URL: https://dhtmlx.com/docs/products/dhtmlxGantt/.

[2] *Impact Documentation*. Accessed: 2025-07-23. URL: https://impact-ng.docs.cern.ch/dev/documentation/.

[3] *Impact Tool Evolving*. Accessed: 2025-07-23. URL: https://home.cern/news/announcement/cern/impact-tool-evolving.

[4] *Information Management Group at CERN*. Accessed: 2025-07-23. URL: https://en.web.cern.ch/group/im.

[5] *Students Lectures Timetable*. Accessed: 2025-07-23. URL: https://indico.cern.ch/event/1508891/timetable/.