

# HORIZON CAMPUS



**HORIZON  
C A M P U S**

<b>FACULTY &amp; INTAKE</b>	<b>Faculty of IT (NMC) – Intake 11</b>
<b>COURSE MODULE</b>	IT31023 - Systems Administration & Maintenance
<b>NAME OF THE ASSIGNMENT</b>	<b>Assignment 4 - CI/CD Pipeline Integration</b>
<b>STUDENT NAME</b>	U. D. S. Bandara
<b>STUDENT INDEX NO</b>	<b>ITBNM-2211-0112</b>

## Contents

Task Tracker – Final Project Overview.....	3
Overview.....	3
Phase 1 – Strategic Planning & Market Research .....	3
Phase 2 – Development Environment & Workflow .....	3
Phase 3 – CI/CD & Quality Assurance.....	4
Phase 4 – Production Deployment & Documentation .....	6
	9
Key Features .....	10
Main Folder Structure .....	10
Visual Documentation .....	11
Tech Stack.....	15
Future Improvements.....	15
References.....	16
Conclusion .....	16

# Task Tracker – Final Project Overview

---

🔗 GitHub Repository: <https://github.com/SubodhaB/task-tracker>

🔗 Live Demo (GitHub Pages): <https://subodhab.github.io/task-tracker>

## Overview

The Task Tracker Micro SaaS is a lightweight, browser-based application that allows users to manage personal tasks with key features like filtering, prioritization, due dates, download daily, weekly or monthly tasks report (PDF), data backup/restore, and browser notifications. It is built with HTML, CSS, and JavaScript and deployed using GitHub Pages with CI/CD integration. This document presents a complete and honest account of the project's lifecycle from planning to deployment following advanced DevOps and software development practices.

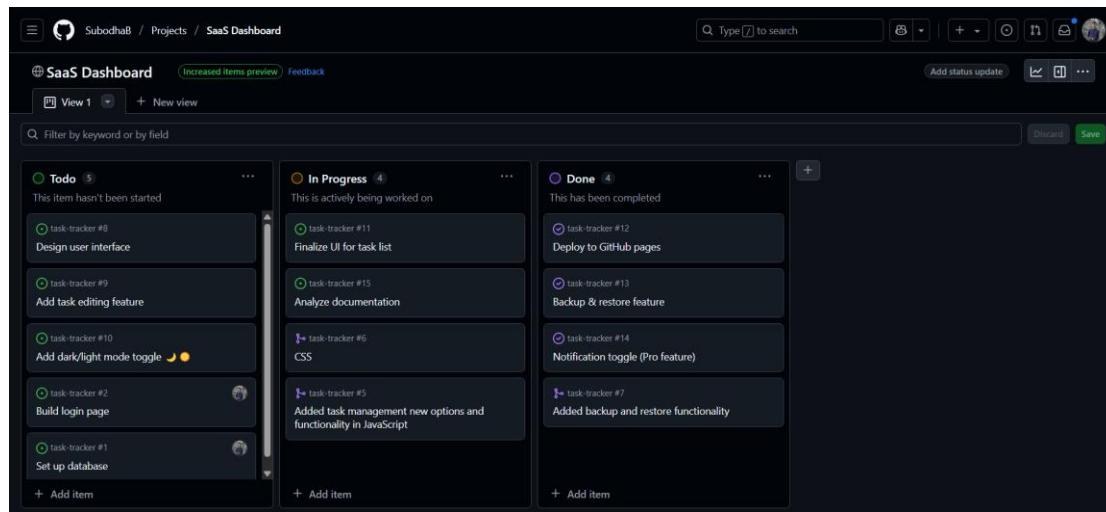
## Phase 1 – Strategic Planning & Market Research

While the scope of the project did not include in-depth external market research, the planning phase focused on understanding the common pain points in task management apps. The app was designed around key user personas (e.g., students and professionals) requiring simplicity, portability, and data retention through local backups. No backend or login system was used, making the app lightweight and focused on the core user journey.

## Phase 2 – Development Environment & Workflow

A GitHub repository was created and used for version control and collaboration. GitHub Projects was used for task planning. Branching and commit best practices were followed throughout development. The code was written using basic HTML, CSS, and JavaScript without external frameworks (like Bootstrap or Tailwind). All code was developed and tested locally before being pushed to the main branch.

## Project Board of Task Tracker



This is the project board was used for task planning.

## Phase 3 – CI/CD & Quality Assurance

A CI/CD workflow was configured using GitHub Actions. The pipeline ensures linting and build success before deployment. GitHub Pages was used to host the project. All features were tested manually on multiple browsers to ensure compatibility. Console logs were used for runtime observability. LocalStorage was used for persistence, and browser notifications were scheduled with validation.

### All workflows

A screenshot of the GitHub repository interface, specifically the "Actions" tab. On the left, there's a sidebar with "Actions", "New workflow", "All workflows", "CI/CD Pipeline", and "pages-build-deployment". The main area shows "All workflows" with a list of "68 workflow runs". Each run is listed with a green checkmark, the name of the workflow (e.g., "changed some unclear things on UI and Improved PDF report format...", "pages build and deployment", "updated readme and screenshot of UI", "Update README.md", "pages build and deployment"), the branch it ran on ("main"), the time it ran ("2 days ago", "3 days ago", etc.), and a "..." button. A search bar at the top right says "Filter workflow runs".

CI/CD Pipeline & pages build deployment

## GitHub Deployment

The screenshot shows the GitHub Pages settings page for a repository. On the left, there's a sidebar with sections like General, Access, Collaborators, Moderation options, Code and automation, Branches, Tags, Rules, Actions, Webhooks, Environments, Codespaces, and Pages. The Pages section is currently selected. The main area is titled "GitHub Pages" and contains information about the site being hosted at <https://subodhab.github.io/task-tracker/>. It shows the site was last deployed by Subodhab 2 days ago. Below this, there's a "Build and deployment" section with a "Source" dropdown set to "Deploy from a branch". Under "Branch", it says "Your GitHub Pages site is currently being built from the main branch." There are buttons for "main" and "/(root)". At the bottom, there's a link to "Learn how to add a Jekyll theme to your site."

*GitHub Pages was used to host the project*

## How We Tested

Throughout the development of the Task Tracker project, rigorous local testing was conducted to ensure each feature worked smoothly and reliably. The application was tested manually in multiple modern web browsers (Chrome, Firefox, and Edge) by opening the index.html file directly. We validated all key user interactions including task creation, priority selection, deadline setting, filtering (All, Pending, Completed), task completion toggling, deletion, localStorage persistence, backup/restore from JSON, PDF report generation, and browser notifications. Console logs (console.log) were used extensively to trace events and debug logic during development.

The screenshot shows the Task Tracker application running in a browser. The app has a header "Task Tracker" and a sub-header "Analyze Your Day". It features a text input "Enter a task", a date picker "mm/dd/yyyy", a priority selector "Medium", a "Add Task" button, and a "Enable Notifications" checkbox labeled "Pro Feature". Below these are three tabs: "All", "Pending", and "Completed". A specific task "Test Console Task" is listed under the "Completed" tab, with details: "4/25/2025, 12:33:39 AM | Priority: High". At the bottom, there's a "Generate PDF Report" button. To the right of the browser window is a developer tools' Console tab showing the following JavaScript code:

```
> console.log("Task Tracker Loaded");
> Task Tracker Loaded
VM34:1
< undefined
> tasks.push({
  text: "Test Console Task",
  priority: "high",
  dueDate: new Date().toISOString(),
  completed: false
});
localStorage.setItem("tasks", JSON.stringify(tasks));
renderTasks();
< undefined
```

A red arrow points from the text "add a task programmatically and display it" to the "tasks.push" line in the console log. Another red arrow points from the text "Java Script runs properly" to the "Task Tracker Loaded" log entry.

*Test the Task Tracker via Console*

We also tested across different screen sizes to ensure the layout remained responsive and clean. Each time a new feature was implemented, we committed the changes to a separate Git branch (e.g., feature/pdf-report, feature/notification), pushed them to GitHub, and reviewed the GitHub Actions CI/CD deployment log to confirm successful automated deployment to GitHub Pages. This combination of real-time browser testing and continuous integration validated the stability and quality of the final app.

#### Phase 4 – Production Deployment & Documentation

The project was deployed on GitHub Pages through GitHub Actions. Backup and restore options were added using the Blob API. Disaster recovery is enabled through downloadable JSON backups. Notifications are optional and labeled as "Pro Feature" for future monetization or gated access. Documentation includes TDD, PRD, Runbook, C4 diagrams, and a wiki.

#### C4 Diagram Overview

The **C4 Model** helps describe the architecture of software systems at multiple levels of detail. For the Task Tracker application, here's how the four layers apply:

##### Context Diagram:

The Task Tracker is a client-side web application used by individual users to manage daily tasks. Users interact with it through a browser. There's no backend server; all data is stored locally using localStorage.

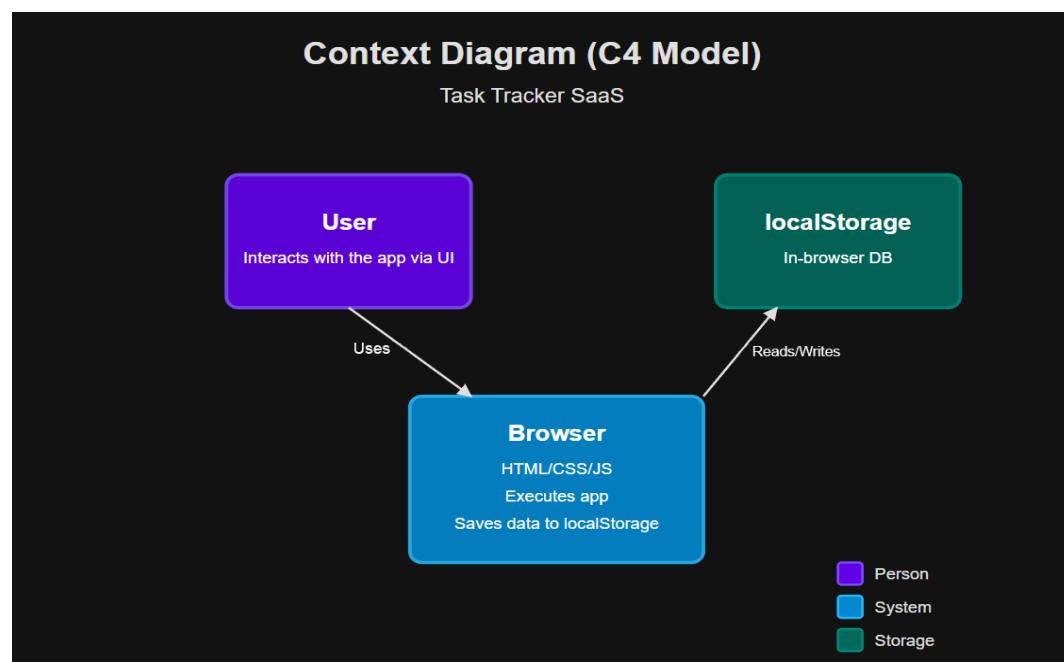


Figure 1.1 – Image of Context Diagram

### Container Diagram:

The application runs entirely in the user's browser. It consists of three main containers:

- ✓ index.html (UI & Structure)
- ✓ style.css (Design & Responsiveness)
- ✓ script.js (Functionality, Logic, Storage, Notifications, PDF generation)

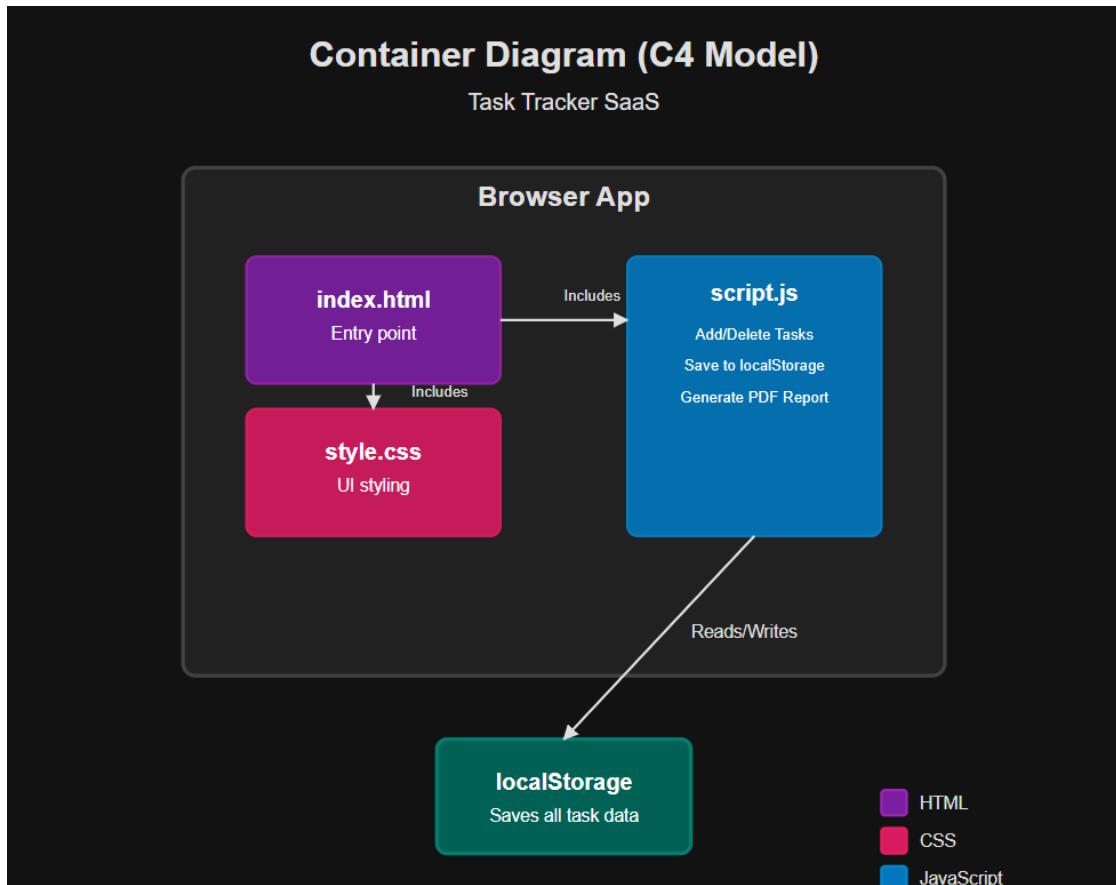


Figure 1.2 – Image of Container Diagram

### Component Diagram:

The script.js file is modularly organized into components such as:

- ✓ Task Management (add/delete/filter/complete)
- ✓ Backup & Restore
- ✓ PDF Report Generator
- ✓ Notification System
- ✓ Data Persistence with localStorage

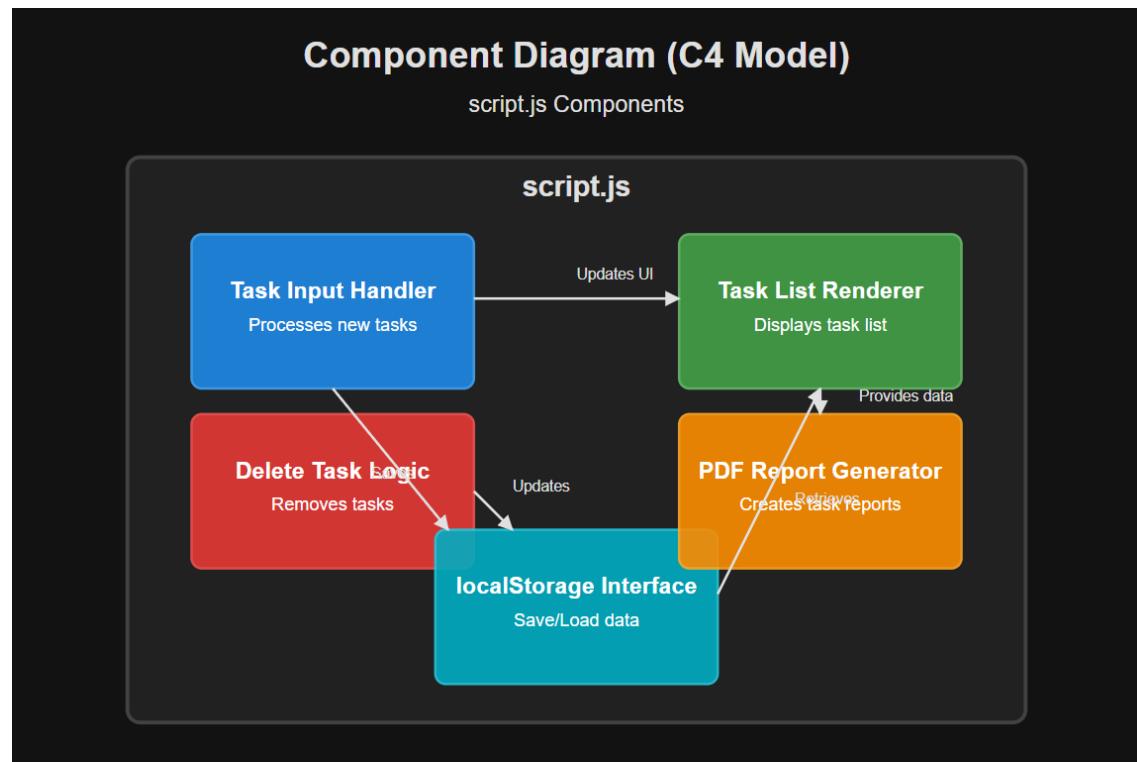
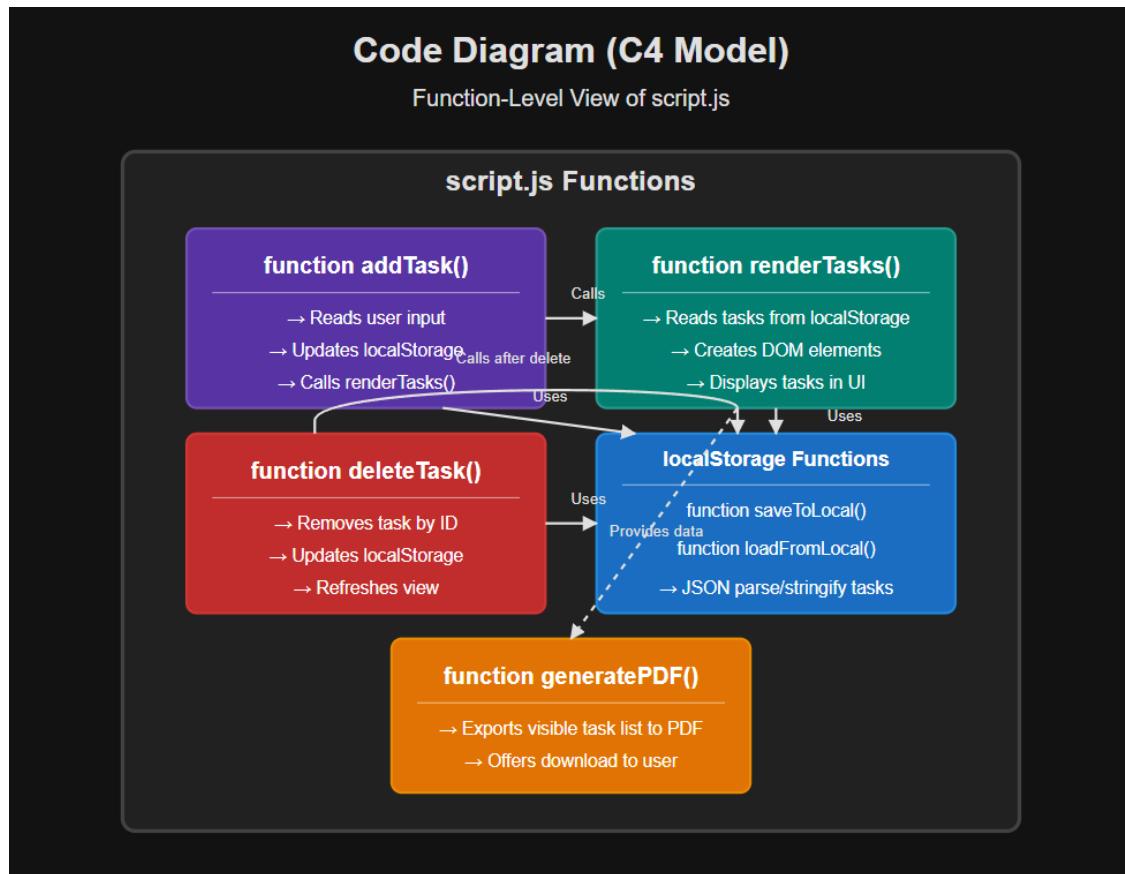


Figure 1.3 – Image of Component Diagram

### Code:

As it's a single-page app (SPA) written in vanilla JS, each function in script.js acts as a code-level component handling a specific operation (e.g., saveTasks(), generatePDF(), scheduleNotifications()).



## Key Features

Feature	Description
<b>Task Management</b>	Add, delete, and mark tasks as completed
<b>Priority Levels</b>	High, Medium, Low labels shown with each task
<b>Due Date Support</b>	Select a deadline for each task with calendar input
<b>Filters</b>	Show All / Pending / Completed tasks
<b>PDF Report Generation</b>	Export all tasks to a well-formatted PDF document
<b>Notifications (Pro)</b>	Alert user 5 minutes before due tasks (with permission)
<b>Backup/Restore</b>	JSON-based export/import for safe offline storage
<b>CI/CD Pipeline</b>	Auto-deploy via GitHub Actions to GitHub Pages

## Main Folder Structure

```
root/
├── index.html      # Main application interface
├── style.css        # Custom styles
└── script.js        # JavaScript logic (tasks, filters, notifications, PDF, backup)
└── .github/
    └── workflows/
        └── ci-cd.yml # GitHub Actions deployment workflow
```

Figure 2 – Image of Main folder Structure

## Visual Documentation

### Main Interface: Task List, Filters, and Action Buttons

The main interface of the Task Tracker is designed to be simple and user-friendly. It includes:

- A **task list** where all created tasks are displayed.
- **Filter buttons** (All, Pending, Completed) allowing users to quickly sort and view tasks based on their status.
- **Action buttons** for each task to:
  - ✓ Mark the task as completed.
  - ✗ Delete the task from the list.
- An intuitive layout that ensures users can interact with their tasks effortlessly, all styled with modern custom CSS.

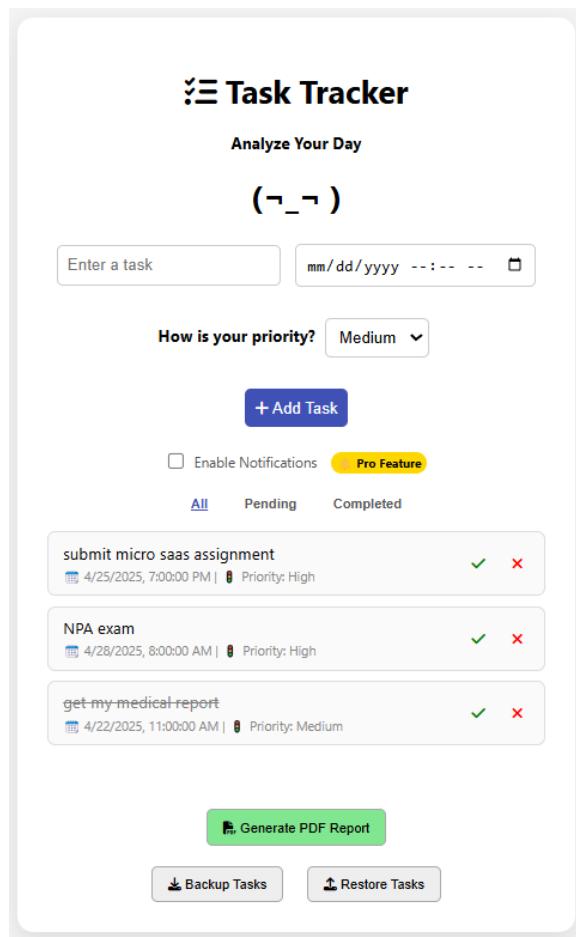


Figure 3.1 - Main interface showing the task list, filters, and action buttons

## Task Form: Add Tasks with Priority & Due Date

While the form is not a pop-up modal, it behaves like one by staying focused and centered on the interface. This form includes:

- A text input to enter the task name.
- A dropdown selector to choose priority (Low, Medium, High).
- A datetime picker to set the task's due date and time.
- A submit button that adds the task to the list and saves it to local storage.

It ensures task creation is quick and organized, and priority + due date are reflected in the generated report and notifications.

The screenshot shows a modal window titled "Task Tracker" with the subtitle "Analyze Your Day". Inside, there is a text input field containing "(¬\_¬)" and a date/time picker showing "04/22/2025 11:00 AM". Below these are priority selection buttons labeled "Low", "Medium" (which is highlighted in blue), and "High". A checkbox for "Enable Notifications" is present, along with a "Pro Feature" button. At the bottom of the modal, there are tabs for "All", "Pending", and "Completed", followed by two task entries: "submit micro saas assignment" (due 4/25/2025, 7:00:00 PM, Priority: High) and "NPA exam" (due 4/28/2025, 8:00:00 AM, Priority: High). A green "Generate PDF Report" button is located at the bottom right of the modal.

Figure 3.2 - Form for adding new tasks with priority and due date options

## Daily Task Report – PDF Export Functionality

A dedicated button labeled “Generate PDF Report” allows users to export all tasks into a downloadable PDF file. The exported PDF includes:

- The task name
- Its priority level
- Due date and time (if set)
- Completion status (Completed / Pending)

This feature is especially useful for reviewing daily productivity or printing physical task records.

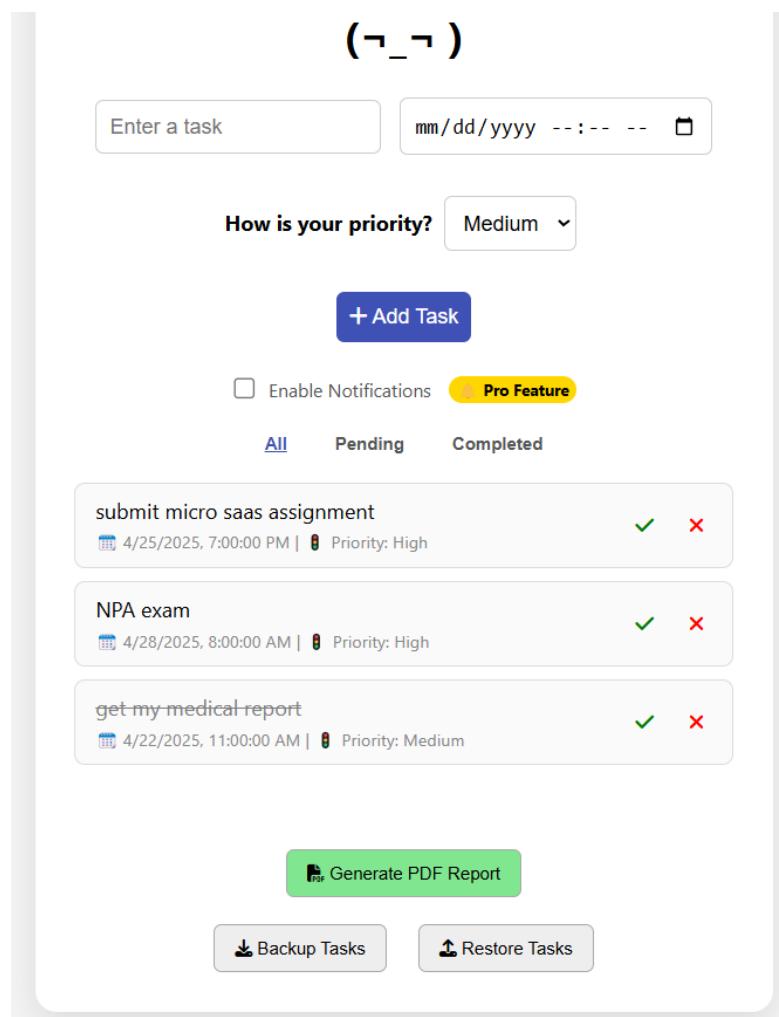


Figure 3.3 - The daily task report PDF export functionality

## Browser Notifications for Task Reminders (Pro Feature)

This optional Pro Feature sends real-time browser notifications for upcoming tasks. How it works:

- If enabled by the user via a toggle switch
- Notifications are scheduled 5 minutes before a task's due time
- The browser will show an alert:  
“⌚ Task Reminder: Your task ‘XYZ’ is due soon!”

It helps users stay on track and never miss an important deadline.

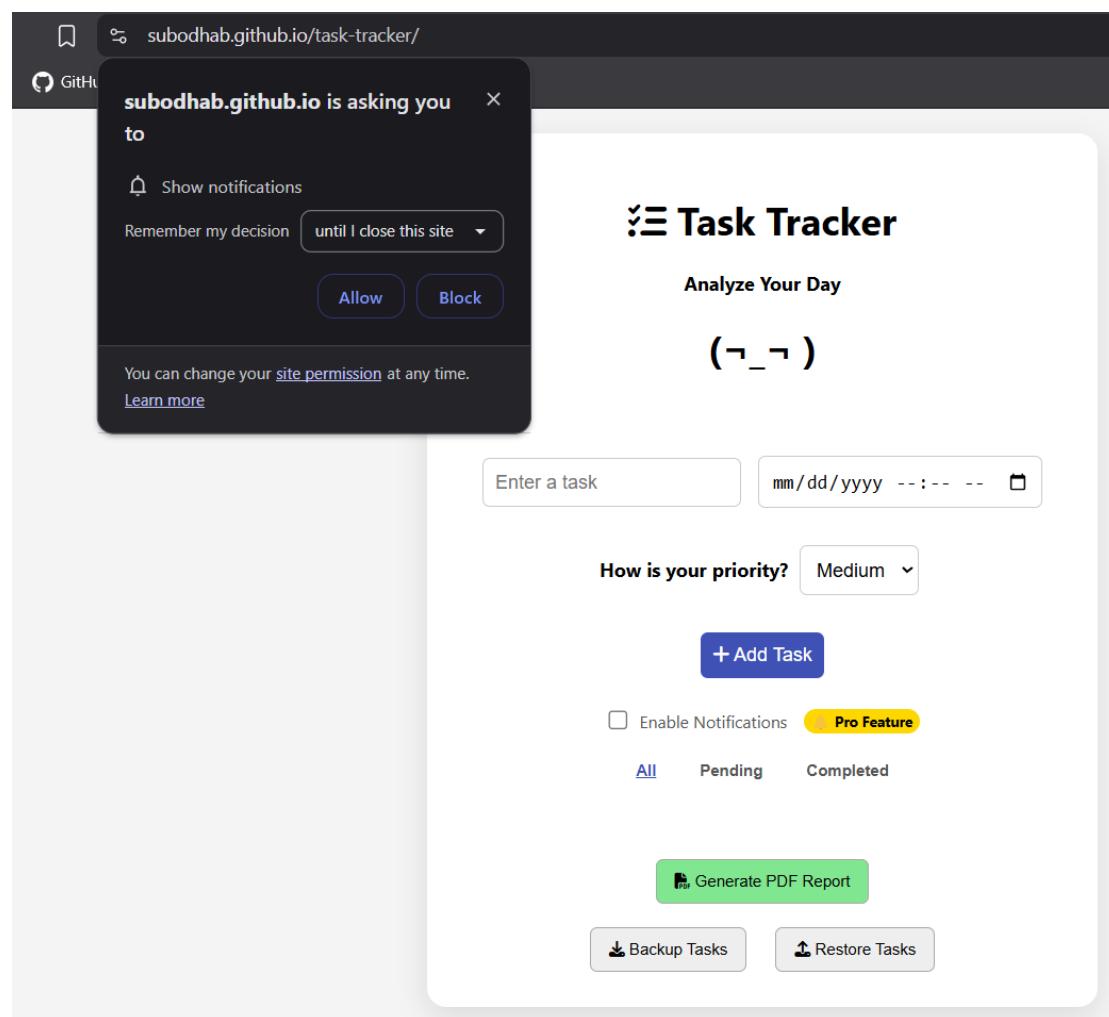


Figure 3.4 - Browser notification implementation for task reminders

## Tech Stack

- HTML, CSS, JavaScript (Vanilla)
- jsPDF library for PDF report generation
- Git, GitHub, GitHub Pages, GitHub Actions
- No backend or database used

## Future Improvements

- Implement task editing functionality
- Add dark mode theme support
- Use service workers for offline mode
- Migrate to IndexedDB or cloud backend (e.g., Firebase)
- Include user login and sync
- Enhanced PDF report customization options
- Integrate charts and analytics for task completion metrics

## References

Topic	Source	Link
HTML/CSS/JavaScript	MDN Web Docs	<a href="https://developer.mozilla.org">developer.mozilla.org</a>
Web Storage API (localStorage)	MDN	<a href="https://developer.mozilla.org/en-US/docs/Web/API/LocalStorage">MDN localStorage</a>
Storing & Retrieving JSON in localStorage	Web Dev Simplified	<a href="https://www.youtube.com/watch?v=KJzXWVgkQHw">YouTube</a>
Deploy to GitHub Pages with CI/CD	GitHub Pages Docs	<a href="https://docs.github.com/en/pages/getting-started-with-github-pages/deploying-your-github-pages-site-with-contINUOUS-delivery">pages.github.io</a>
C4 Model Explained	c4model.com	<a href="https://c4model.com/">c4model.com</a>
Exporting HTML as PDF using jsPDF	CodeSource	<a href="https://codesource.io/">codesource.io</a>

## Conclusion

This project demonstrates a complete, end-to-end SaaS workflow without relying on backend infrastructure. It emphasizes DevOps integration, simplicity in design, and clarity in documentation. A standout feature is the PDF report generation capability that allows users to download comprehensive daily task summaries. All objectives were met, and the application runs without errors or external dependencies.