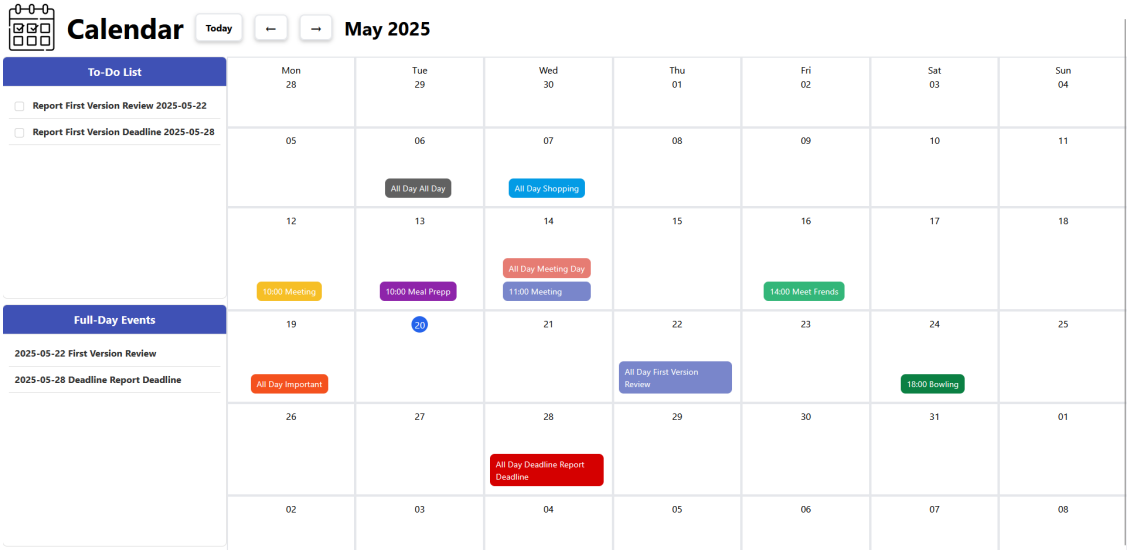
CHALMERS
UNIVERSITY OF TECHNOLOGY



Unified Calendar and Task Management: A Web Application Integrating Google Services

Development and Implementation of a React-Based Productivity Tool

Degree project report in Computer Science and Engineering

Oskar Forsell

DEPARTMENT Computer Science and Engineering

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Degree project report 2025

Unified Calendar and Task Management: A Web Application Integrating Google Services

Development and Implementation of a
React-Based Productivity Tool

Oskar Forsell

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden 2025

Unified Calendar and Task Management: A Web Application Integrating Google Services
Development and Implementation of a React-Based Productivity Tool
Oskar Forsell

Supervisor: Magnus Myreen, Chalmers
Examiner: Muoi Tran, Computer Science and Engineering

Degree project report 2025
Department of Computer Sciecnce and Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: Full Page view of the web application.

Typeset in LaTeX
Gothenburg, Sweden 2025

Unified Calendar and Task Management: A Web Application Integrating Google Services
Development and Implementation of a React-Based Productivity Tool
Oskar Forsell
Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

This thesis project follows the design, development and implementation of a React based web application inspired by features of physical organizational systems. The primary objective was a unified interface capable of gathering information from various online services, specifically Google Calendar and Google Task. The project used the React framework for front-end development, integrating with respective Google API's through Google OAuth 2.0 for authentication and data retrieval. The report follows an adapted agile, sprint-based methodology for solo development, focusing on iterative feature implementation from core calendar functionality to API integration and user interface enhancements.

Keywords: Web Application, Calendar, Agile Development, User Interface, API Integration, Google API, Task Management, Productivity Tool

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

API	Application Programming Interface
CSS	Cascading Style Sheets
UI	User Interface
UX	User Experience
VS Code	Visual Studio Code

Contents

List of Figures

1

Introduction

1.1 Background

Effective planning and organization are fundamental to managing complex lives, whether in a professional or personal context. At its core, planning serves to offload mental burden, providing a accessible repository for important information. Externalizing frees cognitive resources, allowing individuals to focus on task execution rather than on recalling every meeting deadline or commitment. While organizational planning at the enterprise level often involves software suites and dedicated project managers, personal level planning remains a crucial and often daily, endeavour for individuals seeking to stay on top of their schedule and reduce stress.

1.1.1 Existing options

Existing digital planning tools leverage technology to offer portability, accessibility and automations. These tools are typically cloud-based services, allowing users to access their schedule and tasks from virtually any device with an internet connection.

Strenghts

- **Portability and Accessibility:** Information is stored online or on devices, making it accessible from anywhere at any time.
- **Automation:** Digital tools can automate recurring tasks and events. It can carry over unfinished items, set reminders and integrate with other digital services, reducing manual efforts when for example finding link to online meetings.
- **Searchability:** Easily search for specific events or tasks.
- **Collaboration:** Many Digital tools facilitate sharing and collaboration on calendars and tasks. facilitating communication and team work.

Weaknesses

- **Dependency on Technology:** Requires a device and often an internet connection, which can be a limitation in certain environments.
- **Security Concerns:** Storing sensitive personal information online introduces cybersecurity risks, though reputable services employ robust security measures.
- **Information Fragmentation:** A significant weakness is the proliferation of specialized digital applications. Users often find themselves navigating multiple apps. One for calendar events, another for daily to-dos, a third for project management, and perhaps a fourth for long-term goals. Leading to a fragmented overview of their commitments.

Examples

Consider Google Calendar as a primary example. While excellent for scheduling, its interface often requires users to switch between calendar view and separate google tasks interface to manage their complete schedule. As shown in these pictures the Non Scheduled To-Do's do not show up in the calendar since they don't have a scheduled time.

1.2 Gap missing in existing options

Despite the abundance of options in tools for planning, some gaps persist in providing a unified interface for collecting multiple services and improved contextual overview through sidebar with more information. The core issue lies in fragmentation of information across different digital services and the lack of holistic visual access that empowers the users to find the information they are looking for at a glance.

Collection of multiple services

When using multiple of these services for different aspects of your scheduling it may lead to not being able to locate the information that you are after since it is on a different service or account. When information is fragmented over many services and applications it invites mistakes in time management, for instance conflicting commitments in different scheduling tools.

Improved display options

The priority of a time management tool is to work effectively for the user enabling them to easily find the information they are looking for. In the existing options there are no options for displaying certain information in separate views, making them easier to find.

1.3 My Solutions

To address the identified gaps, this thesis proposes the development of a React-based web application that serves as a unified calendar and task management productivity tool. The solutions will directly tackle the fragmentation of information and aim to provide a superior overview.

- **Developing a Unified Interface:** The core of the solution is a single web application interface that fetches and displays the data from other existing services. This hopes to eliminate the need for users to juggle multiple application, presenting all critical information for the users schedule in one coherent view.
- **Leveraging Existing Services:** By leveraging existing services there is no need for databases or datacenter and instead the information can be fetched directly from the services eliminating that workload and cost.
- **Enhancing Information Accessibility:** To provide an improved interface for the user, there is a sidebar added that aims to give the users easier access to the information they are after when using the tool. The two main sections are the tasks list where all the tasks are displayed in order. This also includes unscheduled tasks which are not displayed when looking for the tasks in your google calendar.

  The second section is for upcoming all day events. These events are often used for bigger events like deadlines or birthdays but are given no more importance than normal events in the existing options. By also displaying them in this second section the user may view it for a improved overview of big upcoming events.

1.3.1 Platform

This solution will start out being developed in the framework React. This framework is chosen for the possibility to create a web application and the flexibility to later allow for conversion to React native, however that is not a part of this projects scope as of now.

1.3.2 User Data

To begin with, this project will not enable the user to create and save data in the web-application itself, since this project is more focused on actually displaying useful information. The scope of the project might be extended at a later date to include creating events and tasks through an API.

1.4 Results

The project was not able to be completed to the fullest potential within the time frame, however the project demonstrates some of the core concepts, setting it apart from other services. The outcome of this thesis project is a somewhat functional web application that demonstrates a proof of concept for the unified calendar and task management system. The key results include:

- **Google Service Integration:** The thesis project only provides integration for Google Calendar and Tasks services, but may be expanded in the future to include more services through API integrations.
- **Consolidated information Display:** The developed application consolidates events from Google Calendar and tasks from Google Tasks into a single cohesive user interface. This eliminates some of the switching between google applications to gain a better single glance overview of the users commitment.
- **Enhanced Overview:** By integrating tasks and full day events directly alongside the calendar events, the application provides more information and more ways of finding the information. The task list ensures that the user can find their unscheduled tasks as well as their scheduled tasks and the full day events list provides an overview of upcoming big events in the schedule.

2

Theory

To understand the development process, this section provides relevant context. It begins by outlining the knowledge from coursework, technologies and tools used. Lastly, it introduces the workflow of this project.

2.1 Relevant Coursework

2.1.1 Web Applications – DAT076

In the Web Applications course (DAT076), I worked in a team to develop a to-do list application using modern web technologies, with a focus on client-side development using React.js. The course introduced the principles of full-stack web development, which typically includes a frontend (client-side UI), a backend (server-side logic), and a database (persistent data storage). While this project focused on the frontend, the course also covered backend technologies and architectural considerations.

The main technologies introduced were Node.js, Express.js, and React.js. Node.js is a JavaScript runtime that allows developers to run JavaScript outside the browser, particularly on the server side. It is designed for scalability and efficient I/O handling, making it suitable for building fast, concurrent applications. Express.js is a minimalistic backend framework built on Node.js that simplifies routing, middleware integration, and REST API creation. React.js, used in our project, is a JavaScript library for building interactive user interfaces. It uses a component-based structure and virtual DOM for efficient rendering.

React.js was chosen for its modular design, strong community support, and efficient state management. Its declarative syntax made it easier to implement UI features like calendar navigation, and its component-based architecture allowed for reusable interface elements consistently across the app.

Several core design challenges for web applications were also discussed in the course, including session handling and navigation, both of which were partially addressed in this project:

- **Navigation:** Implemented using React state to allow calendar interactions, such as moving between months and jumping to the current day. This was

managed through the `CalendarHeader.js` component without the need for external routing libraries like React Router.

- **Session Handling:** The app integrates with Google OAuth through the `GoogleCalendarAuth.js` component to authenticate users and access their calendar data. While Google manages authentication, the app does not persist sessions after a page reload, meaning users must re-authenticate each time.

Other topics such as scaling and data persistence were not directly relevant to this project. Since the application was a single-user, client-side app without backend services or a database, there were no scalability concerns. Likewise, data fetched from Google APIs was not persisted locally or in a database; it existed only in memory during each session.

Overall, the course provided valuable practical experience with key frontend technologies and design considerations, many of which influenced later projects, including this thesis.

2.1.2 Agile Software Project Management - DAT257

The Agile course focused on how teams can work together and communicate effectively to reach their goals. While the agile method wasn't fully applicable to this project's team this time. The experience gained through this course and other projects helped adapt the system to a single person team for this project. This will be covered in more detail in section 2.3.1 about sprints and section 3.2 about sprint workflow.

2.1.3 Agile Software Project Management - DAT257

During Agile Software Project Management we worked in teams during exercises and for a project in which we learned and applied the Agile method to develop an application that used an AI Chat-Bot to give the user a recipe that was based on what the user had in their fridge.

Agile is a project management methodology that focuses on how teams can collaborate and communicate effectively to achieve their goals. A core concept within agile frameworks, particularly Scrum, is the "Sprint," which is designed to encourage beneficial team behaviors throughout a project. A sprint is a short, dedicated period of time during which a project team commits to completing a specific set of tasks.

The Sprint cycle is structured around four primary stages: Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective.

- **Sprint Planning:** In this initial stage, the team collaborates to decide which tasks will be undertaken during the upcoming sprint and assigns those tasks to various team members.

- **Daily Scrum:** During this phase, the team members focus on executing their assigned tasks.
- **Sprint Review:** Upon completion of the sprint, the team presents the work they have accomplished.
- **Sprint Retrospective:** The cycle concludes with the team reflecting on the past sprint to identify areas for improvement that can be implemented in subsequent sprints.

While the Agile Methodology is meant for improving teams workflow, focusing on collaboration, task division among members and daily communication (Daily Scrum) to coordinate efforts and address issues.

While the Agile Methodology is meant for improving teams' workflow, focusing on collaboration, task division among members, and daily communication (Daily Scrum) to coordinate efforts and address issues. For a single-member project such as this, the core principles of agile required some adaptations. This involved reinterpreting team-centric practices to suit an individual workflow, prioritizing iterative development and continuous reflection. The specific details of how these agile principles were adapted and implemented within this project, including the modified sprint structure and workflow, are further elaborated in Section 2.3.1 Development Sprints and 3.2 Sprint Workflow.

The "Agile Software Project Management - DAT257" course provided valuable practical experience that influenced the development methodology of this project. This coursework provided the foundation for understanding how to maintain a structured and iterative workflow even without a dedicated team.

Specifically, the emphasis on iterative development, as practiced through the sprint cycles in the course, guided the project's focus on incremental feature implementation. Although the project lacked a traditional team, the discipline of daily planning and progress tracking, conceptualized from the "Daily Scrum" and personal journaling, became integral to maintaining focus and efficiency. Furthermore, the course underscored the importance of continuous reflection, which was emulated through regular self-assessment at the end of each development phase. These reflections, similar to Sprint Retrospectives, enabled proactive problem-solving, that will be expanded on in this thesis report.

2.1.4 Project - DAT067

The project course focused on the way the team communicates, such as formulating project goal, dividing roles and tasks, making plans for timeline, identifying knowledge requirements and finding information needed, developing and evaluating solutions. All of these concepts were useful not only the planning phase of this project but also in the development and evaluation of the features.

2.1.5 Others

While these courses didn't focus directly on applicable concepts to this project. Their concepts helped in visualizing data/code structures and working towards a project goal.

- Object Oriented Programming - DAT050
- Object Oriented Applications - DAT055
- Data Structures and Algorithms - DAT495

2.2 Technologies and Tools

Necessary

2.2.1 API(Application Programming Interface)

An Application Programming Interface (API) is a set of rules and protocols that allows different software applications to communicate and exchange data. APIs abstract the underlying complexity of software systems, providing developers with a standardized way to access functionalities and data without needing to understand the internal implementation details. This promotes modularity, interoperability, and efficient software development by enabling the integration of various services and components [10]. The concept of an API is necessary and fundamental to this project's core objective: creating a unified interface that gathers information from external online services. The entire application is built around consuming these interfaces from Google and other providers to fetch and display user-specific data.

2.2.2 Google API

Google APIs are a collection of interfaces that allow developers to integrate various Google services into their applications. The Google Cloud Console provides a web-based graphical interface for managing these cloud resources and services, including access to Google APIs [6, 5]. For this project, the integration of Google Calendar and Google Tasks APIs was necessary as a core objective. These specific APIs were utilized to fetch and synchronize user calendar events and task list data, which form the central information displayed within the application. The Google Cloud Console was used to configure necessary credentials, set up the OAuth consent screen, and manage API access permissions, all critical steps for enabling the application's functionality.

2.2.3 Google OAuth 2.0

Google OAuth 2.0 is a secure authorization framework that enables applications to obtain limited access to user accounts on Google services. It allows users to grant applications permission to access specific resources without sharing their login credentials, enhancing security and privacy [1]. Google OAuth 2.0 was necessary for this project as the critical security mechanism for user authentication and authorization. Its implementation was essential to allow the application to securely access and manage a user's private data from both Google Calendar and Google Tasks APIs, ensuring adherence to security best practices for delegated authorization. This secure login flow was a significant part of the development.

Optional

2.2.4 React Framework

React is a JavaScript library for building user interfaces (UIs). It enables developers to create reusable UI components and efficiently update the UI in response to data changes [7].

For this project, React was chosen as the foundational framework due to its component-based architecture, which was highly advantageous for developing an interactive web application like the Unified Calendar and Task Management system. The project required a dynamic and responsive user interface capable of seamlessly integrating data from multiple services (Calendar and Tasks).

React's approach to UI development allowed for the creation of distinct, modular components—such as individual calendar events, task items, navigation elements, and interactive forms. This modularity facilitated efficient development, easy maintenance, and the ability to independently manage different parts of the UI. Furthermore, React's efficient DOM updates, driven by its virtual DOM, ensured a smooth and performant user experience. The framework's robust state management capabilities were crucial for handling the application's data flows and ensuring that the UI accurately reflected the current state of integrated Calendar and Task data.

2.2.5 GitHub

GitHub is a web-based platform for version control using Git. It is widely used for storing, managing, and collaborating on codebases [4]. During this project, GitHub was utilized for version control, specifically by creating new branches from ‘Main‘ for every significant feature implementation. While highly recommended for any software project to track changes and facilitate collaboration (even solo), it is categorized as optional in the strictest sense as the project's core functionality

does not depend on its online presence or collaborative features, though its use significantly streamlined development and provided robust version history.

2.2.6 Visual Studio Code

Visual Studio Code (VS Code) is a popular, open-source source code editor developed by Microsoft. It supports a wide range of programming languages and offers features like debugging, Git integration, and a rich ecosystem of extensions [2]. VS Code served as the primary Integrated Development Environment (IDE) for this project. Although not strictly necessary for the application's runtime, a code editor is indispensable for development. VS Code was chosen for its comprehensive feature set and customizability via extensions, which greatly enhanced development efficiency and code quality. Any capable code editor could theoretically be used, but VS Code provided the development environment for this project.

Personal Use

2.2.7 Obsidian.md

Obsidian is a powerful note-taking and knowledge management application that operates on a local folder of plain text Markdown files [8]. While not a direct web development tool, it proved highly valuable for personal use in documentation and knowledge organization throughout the project. It was primarily used as a daily note-taking tool to log progress, document challenges encountered, and plan future development steps. This practice was particularly beneficial for compiling weekly summaries of insights, accomplishments, and areas for improvement, directly aiding in the planning of subsequent development sprints and reflecting on the overall project trajectory.

2.2.8 ChatGPT

ChatGPT is a large language model (LLM) developed by OpenAI, capable of generating human-like text and assisting with various natural language processing tasks. In a development context, it can provide support for code generation, debugging, and documentation assistance. For personal use during this project, ChatGPT and other LLMs served as a productivity aid, assisting in finding technical information, summarizing complex documentation, and deciphering error messages. While highly beneficial for accelerating research and problem-solving, its use was supplementary to core development activities.

2.2.9 Figma

Figma is a collaborative, web-based design tool primarily used for creating user interface (UI) designs and interactive prototypes [3]. For personal use in this project, Figma was utilized to improve the planning process for the user interface. Specifically, during Sprint 5 3.3.5, a prototype of the application's UI was created in Figma to visualize the layout of the sidebar with its integrated sections for the To-Do list and Full-Day Events. This allowed for a more structured design phase prior to coding, facilitating better visualization and planning of the UI elements before implementation.

2.3 Development Sprints

The concept of a Sprint, primarily from the Scrum framework, is designed to encourage beneficial behaviors within a project team, focusing on iterative development and continuous improvement [9]. Sprints are defined as short, fixed periods during which a team dedicates itself to completing a specific set of tasks. A standard Sprint cycle is typically divided into four main sections: Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective. Durin Sprint Planning, the team collectively decides which tasks will be addressed within the upcoming sprint and assigns responsibilities for those tasks. The Daily Scrum is a brief daily meeting where team members synchronize their activities, report on progress, discuss blockers, and re-plan for the day ahead. At the Sprint Review, the team demonstrates the work completed during the sprint to stakeholders, showcasing the new functionalities. Finally, the Sprint Retrospective allows the team to reflect on the completed sprint, identifying successes and areas for improvement to enhance future sprint cycles.

This sprint-based methodology was chosen for this project due to its inherent benefits in managing software development: it promotes a structured approach to iterative feature implementation, encourages continuous feedback and regular self-assessment. These aspects were particularly relevant given the project's goal of integrating multiple external APIs and developing a dynamic user interface.

2.3.1 Adaptation for a Single-Member Team

While traditionally applied to multi-member teams, the core principles of the agile sprint were adapted to suit the single-member nature of this project. Instead of a "team" dedicating to work, the sprints were re-conceptualized as periods for developing "Major Features," each given a goal deadline rather than a strict time-boxed commitment. The adaptation for each sprint section was as follows:

- **Sprint Planning:** This involved the individual developer defining the scope of the "Major Feature" to be implemented, breaking it down into smaller, man-

ageable tasks, and prioritizing them for the upcoming period. This ensured a clear focus and roadmap for development.

- **Daily Scrum:** Instead of a team meeting, the Daily Scrum was adapted into a personal daily note-taking routine. At the start of each day, tasks for the day were planned. At the end of the day, a journal entry was updated to reflect completed work, challenges faced, and immediate next steps. This served as a continuous progress tracking.
- **Sprint Review:** The Sprint Review was performed through self-demonstration and testing of the completed "Major Feature." This involved thorough internal testing to ensure functionality and adherence to initial requirements before considering the feature complete.
- **Sprint Retrospective:** This involved a critical self-reflection at the end of each "Major Feature" development period. The developer reviewed what went well, what challenges arose, and what could be improved in subsequent development phases. This continuous learning loop, formalized through journaling and summarizing these reflections, significantly contributed to refining the workflow and decision-making for future tasks.

3

Methods

This section describes the development approach used for this project, explains the workflow and "Sprint-based" workflow and the technical decisions made throughout this project. It provides a timeline of development and explains how tools and technologies were applied and implemented for the calendar application.

3.1 Testing

In software development, a variety of formal testing strategies are available to ensure code quality and reliability. These include unit testing for verifying the correctness of individual components or functions in isolation, integration testing to assess how different components interact, and end-to-end (E2E) testing that simulates real user interactions across the full system. Additionally, snapshot testing is often used in front-end development to detect unintended UI changes by comparing rendered output over time. These structured approaches improve test coverage, ensure consistency, and enhance maintainability.

However, due to the limited scope and strict time constraints of this project, a formal testing framework was not implemented. Instead, testing was performed manually during the development process. This included interacting with the user interface to confirm that components behaved as expected, and checking individual features immediately after their implementation or modification. This informal testing method allowed for the quick identification and resolution of obvious bugs and usability issues.

Although manual testing lacks the rigor and reproducibility of automated approaches, it was deemed sufficient for the needs of this project. The focus was on rapid prototyping and iterative feature development, which benefited from the flexibility and immediacy of manual testing. In a future iteration or expanded version of this project, incorporating a structured testing strategy would be advisable to support scalability and maintain long-term code quality.

3.2 Sprint Workflow

To adapt sprint to a single person they were instead periods for a "Major Feature" instead of short periods of time. These Major features were given a goal deadline instead of the other way around. I used a daily note as a journal at the start of the day to plan what needs to be worked on and at the end of the workday to update it on what was completed. The Features had priorities so that the next task was easy to find.

3.3 Sprint Timeline

3.3.1 Sprint 1 – Foundations and First Decisions

Sprint Planning The primary goal of the first sprint was to establish the foundation of the web application using React. I planned to integrate TailwindCSS for a utility-first CSS approach and speed up development. Additionally, I aimed to explore calendar implementation options and complete the initial planning report for this project. This would be the App component in the architecture as seen in figure 3.2.

Daily Scrum As a solo developer I was able to adjust priorities as challenges arose. Early in the sprint I encountered persistent issues with getting some aspects of TailwindCSS working correctly for my project. I ultimately decided to go with traditional CSS because of the issues in combination with more structure that I found useful as a beginner to CSS.

Sprint Review By the end of the sprint, the React environment was fully configured, and the base layout using regular CSS was established. Although I initially intended to use TailwindCSS, switching to plain CSS turned out to be a productive decision that allowed for better control and understanding. I also evaluated several third-party calendar solutions but concluded that building a custom calendar would offer more learning value and flexibility for this specific use case. The planning report for this project was finalized and submitted during this sprint.

Sprint Retrospective Looking back, one key improvement area was overestimating the ease of integrating new tools like TailwindCSS without prior setup experience. Overall, the sprint provided a solid starting point and clarified the scope and direction of the application.

3.3.2 Sprint 2 – Building the Calendar Core

Sprint Planning The objective of the second sprint was to build the core calendar interface specifically, a functional month view that could be the foundation for future features. This included rendering a navigable calendar grid and establishing a header section with control buttons for switching between months. Additionally, I aimed to review the overall project structure and implementation choices with a more experienced developer for constructive feedback. The things worked on during this sprint are the calendar grid in section Month and Day, and the CalendarHeader all found in figure 3.2

Daily Scrum While working individually, I continued maintaining a structured task list and documented progress daily. The majority of this sprint was focused on implementing the logic for generating the calendar's month grid dynamically, considering different month lengths and start days. After this, I implemented a header section with buttons that allowed navigation between months. Attention was paid to ensuring a clean and scalable code structure for future extension. I also prepared discussion points and questions for a peer review meeting scheduled for the end of the sprint.

Sprint Review By the end of the sprint the calendar's core functionality was in place, including dynamically generated month view, header section displaying current month and year and navigation Buttons to switch between months updating the view accordingly.

Sprint Retrospective While the goals of the sprint was meet I still did not feel comfortable with the codebase and structure. that is evident later on when comparing the structure of the early sprints code with the later sprints code. The example in particular is that the events inside the grid system are not components themselves in the file structure but instead created inside the component called Day found in figure 3.2 this in comparison to the task list where the task as a separate component makes the code and structure more readable and maintainable.

3.3.3 Sprint 3 – Prototyping Events and Journaling Workflow

Sprint Planning Throughout this sprint, I followed a consistent habit of logging daily progress, which supported both technical reflection and time management. Early in the sprint, I designed and implemented a modal window that could be triggered by a new button in the calendar interface. This modal allowed users to input and view event details. Although limited in functionality, this served as a valuable prototype for testing state handling and user interaction. The part of this sprint working on prototyping the events in the calendar is not in use in the

application but was the start of events found in both the Day section that displays the events and the GoogleCalendarauth that creates the events found in figure 3.2.

Daily Scrum Throughout this sprint, I followed a consistent habit of logging daily progress, which supported both technical reflection and time management. Early in the week, I designed and implemented a modal window that could be triggered by a new button in the calendar interface. This modal allowed users to input and view event details. Although initially limited in functionality, this served as a valuable prototype for testing state handling and user interaction. Concurrently, I restructured my journaling system by introducing weekly summaries to group related insights, challenges, and accomplishments across each development week.

Sprint Review By the end of this sprint, the following deliverables were completed:

A new event creation modal was added and made accessible via a dedicated button within the calendar UI. Events created through the modal were rendered within the calendar view, supporting basic input and display functionality. The journaling method used in Obsidian was successfully revised to include weekly summaries, improving the structure and usability of project documentation. These outcomes laid essential groundwork for connecting the application to real event data via the Google Calendar API in future sprints.

Sprint Retrospective This sprint highlighted the practicality of journaling and summarizing the journals helped in future planning and staying on track during a sprint.

3.3.4 Sprint 4 – API Integration and Authentication Challenges

Sprint Planning The main objective of Sprint 4 was to integrate the Google Calendar API into the application, enabling it to fetch and display real calendar events from the user's Google account. During this sprint the prototyping from the earlier sprints where implemented, working on the GoogleCalendarAuth component found in figure 3.2.

Daily Scrum Throughout the sprint, I faced challenges in implementing the Google Calendar API. Due to a lack of planning and research, I initially wasted time attempting to implement both OAuth 2.0 and direct API calls simultaneously, not realizing they were distinct solutions to the same problem. Another struggle during the sprint was in configuring the Google Cloud Console correctly. Significant time was spent navigating the Google Cloud Console, where misconfigured OAuth consent screens, credential scopes, and redirect URIs caused repeated authentication

errors. Tshese issues required multiple iterations of trial-and-error configuration and re-reading documentation.. Eventually, I was able to correctly set up a functional OAuth 2.0 flow suitable for my React-based project.

Sprint Review Despite the early setbacks the sprint concluded successfully with the following outcomes. OAuth 2.0 authentication was correctly implemented, allowing users to log in and authorize access to their calendar data. The application could fetch event data from the Google Calendar API and prepare it for integration into the custom calendar interface. The Google Cloud Console project was correctly configured, including the necessary credentials, consent screen, and permissions. This marked the transition from working with placeholder data to real, dynamic input from users' calendars, significantly enhancing the application's functionality and relevance.

Sprint Retrospective This sprint reinforced the need for understanding external tools and services and their configuration when applying them to a project instead of going in head first. The challenges with OAuth setup and console configuration initially slowed development, but the process provided valuable experience in debugging third-party service integration. Going forward, I plan to allocate more time during sprint planning and Daily Scrum to understand the tools and services I am working with before trying to implement them.

3.3.5 Sprint 5 – Sidebar Design and Planning Improvements

Sprint Planning Sprint 5 was focused on implementing a Sidebar component in the application's layout. The goal of the sidebar was to provide a dedicated space for displaying important information to the user. Specifically To-do's and Upcoming Full-Day events such as birthdays, deadlines or major events. Because of the past sprint I planned to begin with a user interface prototype before development. During this sprint the sidebar was created aswell as prototypes for the Task List and FulldayEventsList, all found in figure 3.2.

Daily Scrum Looking for UX prototyping tools led me to Figma, a web-based interface design tool to sketch and explore potential layouts. I started with making the current view then adding a sidebar and sectioning it between To-do's and Full Day Events. After finalizing the mock-up, I started working on making something similar in this project and made a static sidebar component within the React application. This initial implementation included placeholder list items to represent the future data.

Sprint Review By the end of the sprint, a sidebar layout was designed in Figma to serve as a reference for development. A React sidebar component was created and embedded into the main layout of the application. The sidebar displayed placeholder

full-day events, allowing for visual testing and feedback without back-end data integration at this stage. This sprint introduced a more structured design phase to the workflow and laid the groundwork for displaying personalized, high-priority information alongside the calendar view.

Sprint Retrospective This sprint highlighted the benefits of more thoughtful planning and design before implementing, even though the task might have been less complex then the tasks of sprint 4. This sprint was the first sprint focused on something more unique for this application and the planning before development also improved the structure of the codebase in comparison to the code from the earlier sprints.

3.3.6 Sprint 6 – Task Integration and UI Enhancements

Sprint Planning The primary goal for Sprint 6 was to integrate the Google Tasks API into the application, enabling users to access and view their personal tasks alongside calendar events. This would expand the application's utility by consolidating both time-bound events and ongoing tasks into a single interface. A secondary objective was to improve the way information was displayed for both events and tasks, offering users more context at a glance. The section of this sprint working on integrating tasks was mostly working on implementing Google Tasks inside the GoogleCalendarAuth component aswell as the tasks inside the TaskList, while the improved UI elements where found in the task elements aswell as events elements found in the day component in figure 3.2.

Daily Scrum Drawing on lessons from the Google Calendar API integration, I explored the requirements for accessing the Google Tasks API using OAuth 2.0. After reviewing the relevant scopes and endpoints, I implemented the authentication and API call logic necessary to fetch task data from the user's account. In parallel, I worked on enhancing the event and task display components within the application, expanding the level of detail shown.

Sprint Review By the end of the sprint, the following milestones had been achieved:

Successful integration with the Google Tasks API, using the existing OAuth 2.0 authentication flow. Retrieval of the user's task lists and individual tasks. Updates to the UI components for both calendar events and tasks, enabling more detailed and informative displays. Preliminary visual alignment of tasks and events within the app interface, preparing for future refinement in their layout and categorization. These additions brought the application closer to functioning as a personal productivity dashboard, combining calendar-based scheduling with task tracking.

Sprint Retrospective This sprint reaffirmed the value of building reusable authentication logic and applying it across multiple Google APIs. The integration process was smoother than previous attempts, largely due to prior experience with OAuth configuration.

3.3.7 Sprint 7 – Final Touches and User Experience Polish

Sprint Planning For the final Sprint of project focused on enhancing the accessibility of informaion within the application for the user. The goal was to implement clickable popups for both the calendar events in the month view and the Google tasks in the To-do's. Including additional details that wouldn't practically fit in the overview layout. This sprint focused on the TaskInfoModal and EventInfoModal found in figure 3.2.

Daily Scrum Development during this sprint was streamlined. I began by outlining some data fields in the data recieved from the API's and created separete modals for the events and tasks. After that added the onClick events on calendar items and task entries.

Sprint Review By the end of the sprint: Modal components were implemented for both events and tasks. Users could now click on a calendar event or task to view more detailed information in a popup.

Sprint Retrospective As the final sprint of this project, this phase focused on polishing the user experience. The successful implementation of interactive modals marked the completion of all major planned features. Reflecting on this project, this sprint highlighted the importance of small, focused enhancements that can greatly improve usability.

3.4 Codebase Design

App

Entry point that assembles the layout of components. This will serve as the connection between all the other components in the web application.

CalendarHeader

The CalendarHeader is a section on the top of the webpage and serves as mostly as a navigation interface, it is able to switch out the grid in the calendar for different months and years to display for the user.

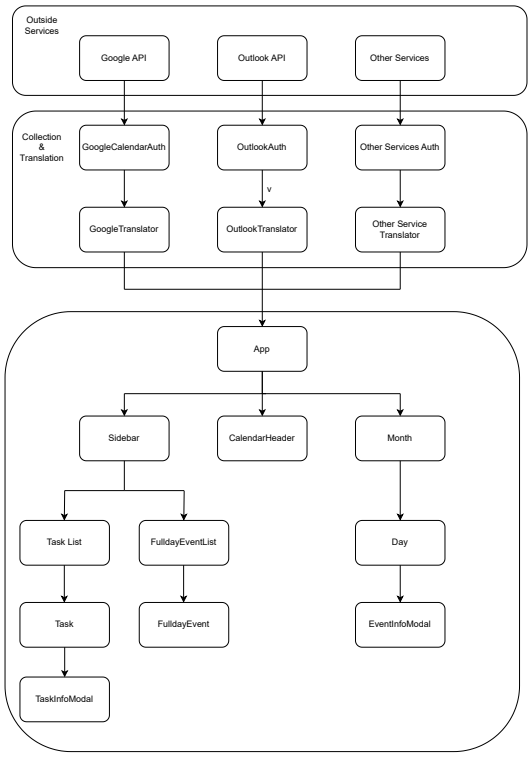


Figure 3.1: Planned Architecture

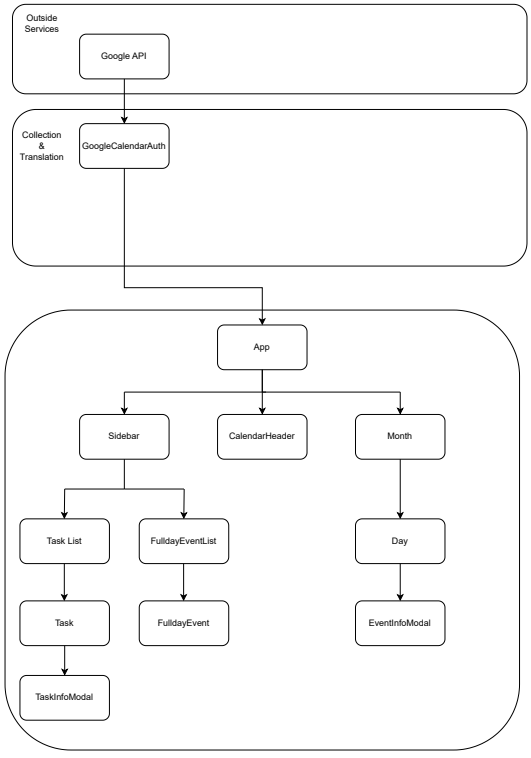


Figure 3.2: Current Architecture

Sidebar

The Sidebar contains two major sections. The fist being the Task List, the task list is populated by tasks from the users Google Task account through the GoogleCalendrAuth component and displays them inside the task list. The second major section is the FullDayEventsList which is populated by collecting the events from the users Google Calendar account through GoogleCalendarAuth and then sorting and removing events that are not relevant for this section.

Task

Each task that populates the task list contains a checkbox, followed by a title followed by a date for the tasks deadline.

TaskInfoModal

When clicking on one of the tasks that populate the task list the task clicked will expand down, creating a drop down with more information, currently containing a close button, a status and the due date again.

FulldayEvent

Each full day event is collected from the events of the user. Sorting out and removing events not relevant to this section, this means removing all events that are not all day events, removing all events that are passed events, and sorting the events to the order of their date.

Month

The month is populated by a grid of day components that is 6 tall 7 wide to make sure all days in the month are displayed. This component is also responsible for putting the correct days in the correct position on the grid.

Day

The Day components represents a day in the callendar but also creates and displays the representation for the events in their corresponding day aswell, being responsible for the colourcode for the events, their displayed time and title and even the on click for opening their EventInfoModal.

EventInfoModal

The EventInfoModal is responsible for popping up when an event is clicked and displaying additional information, currently displaying a date and time aswell as the organizer for the event.

GoogleCalendarAuth

Handles authentication, fetching and storing data into context. This is the cornerstone of all the information gathered in the current state of the project and is responsible for gathering all the information that the users is displaying in the web application.

3.4.1 Styling

Uses CSS modules for component styling.

- Calendar.module.css
- Sidebar.module.css

3.4.2 State Management

- React Context is used for global state, with the context value from GlobalContext.js.
- Fetched Data is stored in context (calendarEvents and savedTasks) and used by components displaying information.

3.5 Key Features

This section outlines the core functionalities implemented in the Unified Calendar and Task Management web application, highlighting how each feature contributes to the overarching goal of a centralized productivity interface.

Calendar Month View with Events: This feature provides users with a comprehensive, interactive monthly calendar display. It visually aggregates all scheduled events from integrated services, allowing for a quick overview of commitments across different days. Events are clearly indicated inside their corresponding day, providing a visual representation of the user's schedule at a glance.

Task List With Due Dates: Integrated alongside the calendar, this component presents a consolidated list of tasks, prioritizing them by their due dates. This feature aims to combat information fragmentation by gathering all tasks from various sources into a single, manageable list, thereby enhancing task oversight and planning.

Integration With Google Calendar and Google Tasks: A cornerstone of this application, this feature enables seamless synchronization with Google Calendar and Google Tasks via secure Google OAuth 2.0. This integration allows the application to fetch, display and in the future manage events and tasks from these popular Google services, ensuring data consistency and providing a unified view without manual data transfer.

Modal Popups for event and task details: To facilitate efficient interaction and detailed information access, the application incorporates interactive modal pop-ups. When a user clicks on an event or task, a modal window appears, displaying comprehensive details, options for editing (where applicable), and context-specific actions, streamlining the user experience and preventing navigation away from the main view.

Use Case: When using the web application the user can simply log in on the website via their google account. After giving the web application access to their Google Calendar and Google Tasks their data is displayed on the calendar and sidebar. The user can at a glance view the events for the month within the month view of the calendar, get a simpler overview of their upcoming important all day events in the sidebar and also look over their tasks all in one integrated user interface.

4

Results

4.1 Design

As per the goal, the design is based on a normal calendar, with the addition of more sections displaying the calendar information and gathering services in one space. As shown in Figure 4.1. It Contains 3 major sections The Sidebar Figure 4.4, the Header Figure 4.2 and the MonthView Figure 4.3.

4.2 Header

The header as shown in Figure 4.2 is simple and indicates the month and year currently being displayed in the MonthView. The header also containst navigation buttons for the month view alowing the user to to navigate back and forward a month at a time allowing the user to view the schedule in more months than the current month. To simplify the navigation it also contains a Today button used to navigate back to the current month.

4.3 Calendar View

Month View

The month view as shown in Figure 4.3 contains a grid of Days. Each row of days represents a week and the first row also contains the day of the week. All days in the grid contains a number at the top indicating the day of the month as shown in figure 4.7. To assist the user in navigation the current day is indicated by highlighting the day of the month with a blue circle shown in figure 4.8.

Events from the users synced google account is displayed inside the corresponding day. To help with organizing the events adopt the color coding from the google service. Each event is ordered by their time slot in the calendar inside their corresponding day to help the user experience. The start time of the event is displayed before the name of the event, if the event doesn't have a start time and is instead

an all day event it is indicated by All Day text in front of the title of the event as shown in figure 4.7.

4.4 The Sidebar

The sidebar is used both to fight fragmentation and improve accessibility, the section is separated in to two sections as shown in figure 4.4.

To-Do List

The to-do list is located at the top section of the sidebar as shown in figure 4.4 and displays the tasks collected from the users Google tasks in the order of their deadlines. By separating the To-Do list out of the calendar view it allows for displaying the Google Tasks that are not yet scheduled for a time or date which Google Calendars own integration of Google Tasks lacks. This is done to make sure that the user doesn't have to move between applications to make sure they don't miss anything when using their scheduling tool.

Full Day Events List

The all day events list is the bottom section of the sidebar as shown in 4.4, This section aims to improve the accessibility and ease of use of the calendar by displaying the most important upcoming events in this prominent section in addition to the calendar grid. The events are ordered by the closest all day event to the current day and excludes past all day events from the list. This feature was implemented to help the user get a better overview of big events coming up in their schedule.

4.5 Modals

To improve the readability while still keeping all the information accessible for the user the web application implemented modals in the form of drop downs or popups allowing for a more condensed view of the information in a normal view while keeping the information accessible to the user. While the popups and drop down does not currently expand on much information the implementation of them will allow for that addition in the future of the project.

Event Modals

By clicking on one of the events in the month view a modal popup will appear displaying more information about the event that was expanded as shown in the

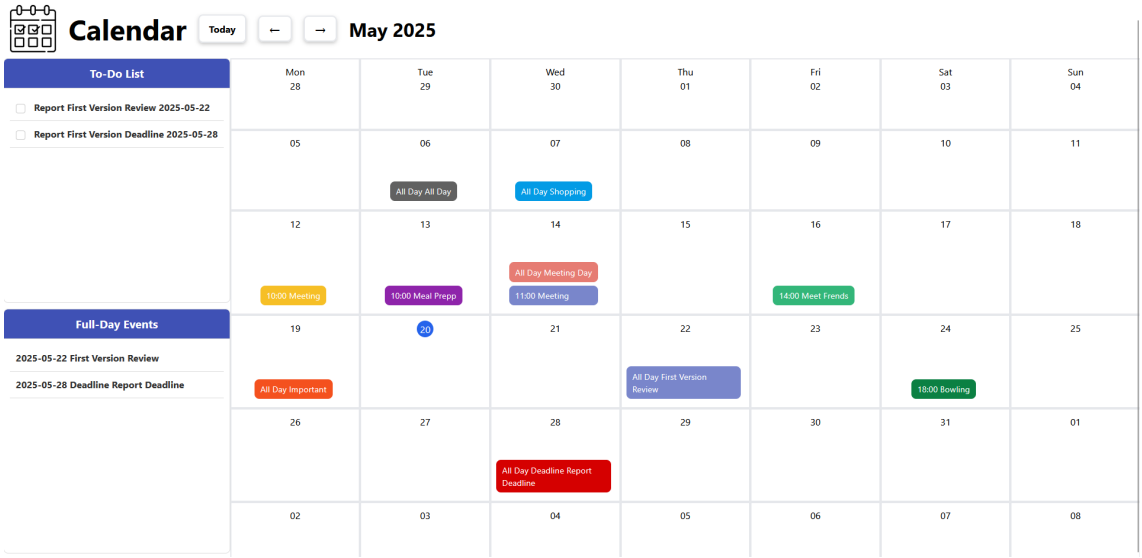


Figure 4.1: Full page

Figure 4.2: Header of the page containing navigation buttons and displaying current month and year

figure 4.9 which is the expanded view of the event called meeting shown in figure 4.7.

To-Do Modals

By clicking on one of the To-Dos in the To-Do list located at the top section of the sidebar as shown in figure 4.5, a drop down will appear displaying more information about the task that was clicked as shown in figure 4.10.

4.6 Quantifiable Measurements

While a comprehensive quantitative evaluation was beyond the scope of this thesis due to time constraints, the developed solution inherently addresses specific inefficiencies present in existing tools, highlighting areas where quantifiable improvements would be observed. One notable example of this project's superiority over Google's native Calendar and Tasks integration concerns the management of unscheduled tasks. When a user adds tasks inside of their Google tasks they will show up in their calendar for the day and time they are scheduled. But non scheduled tasks are not accessible inside the Google Calendar interface which means that these tasks can easily be forgotten when planning. This thesis projects solution puts the unscheduled tasks in the To-Do List ensuring they are not missed during planning.

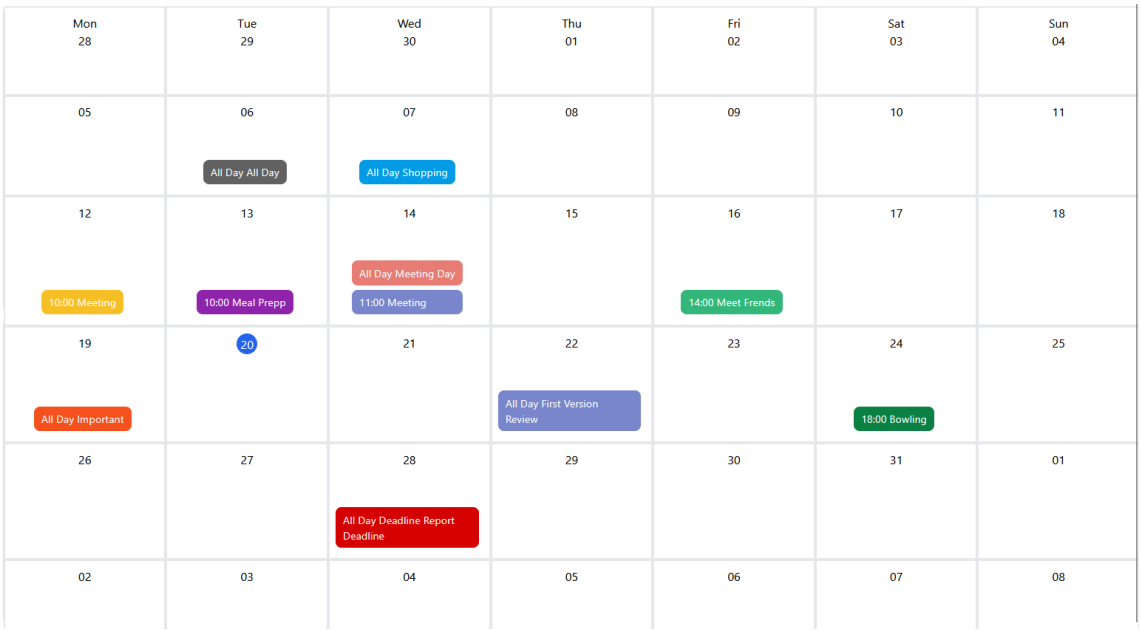


Figure 4.3: Calendar Month View containing a grid of day views

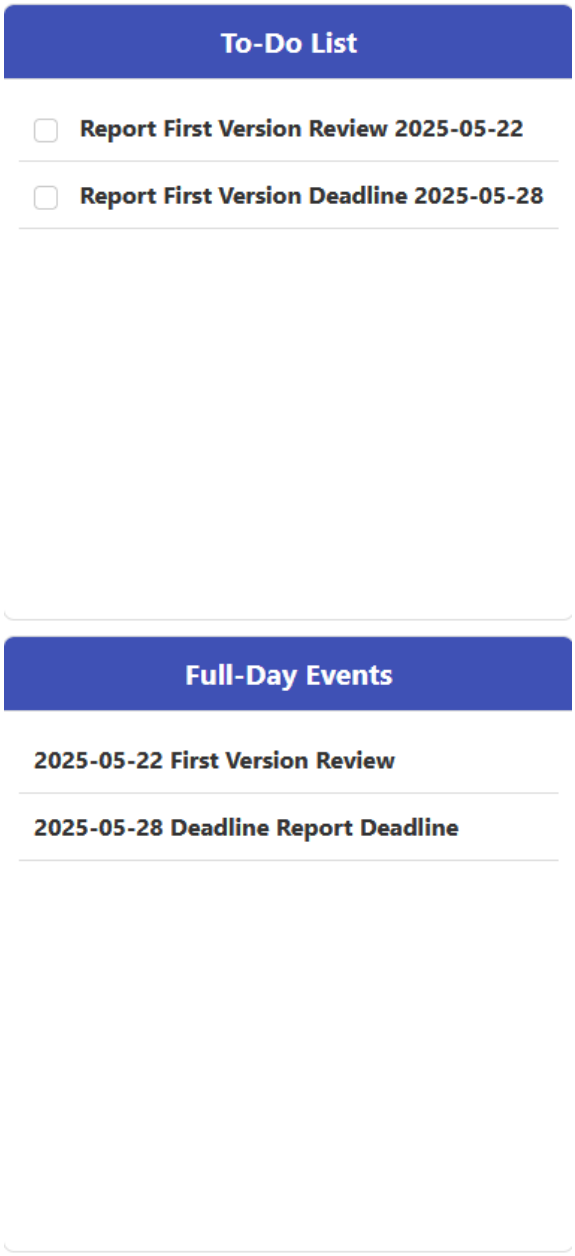


Figure 4.4: Sidebar Containing sections for To-Do List and Full-Day Event List

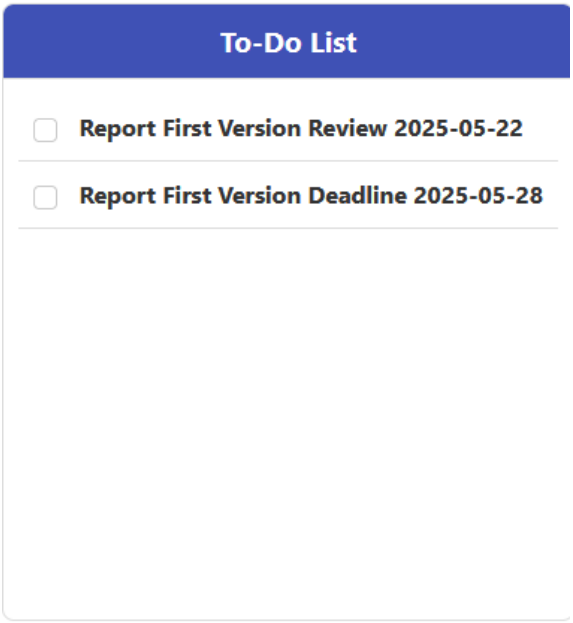


Figure 4.5: To-Do List Containing To-Dos that display Description and Deadline

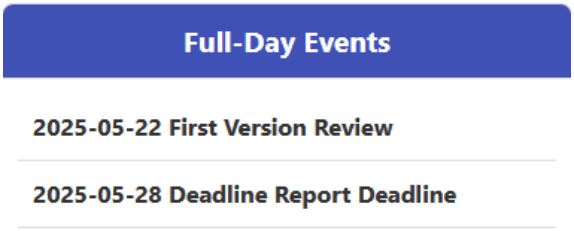


Figure 4.6: Full-Day Event list containing upcoming Full-Day Events displaying Date and Title

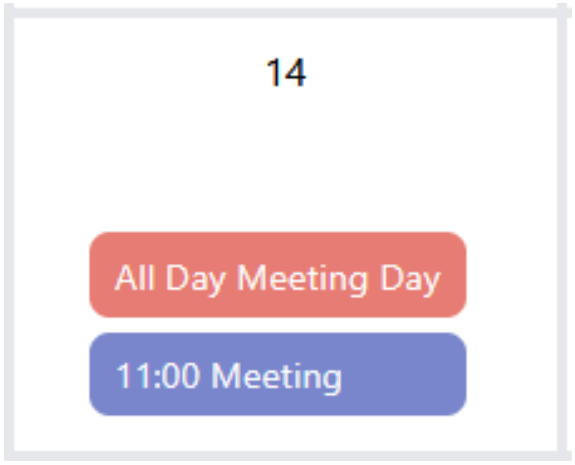


Figure 4.7: Day component containing day of the month and two events that display Title and start time or All-Day Event indication

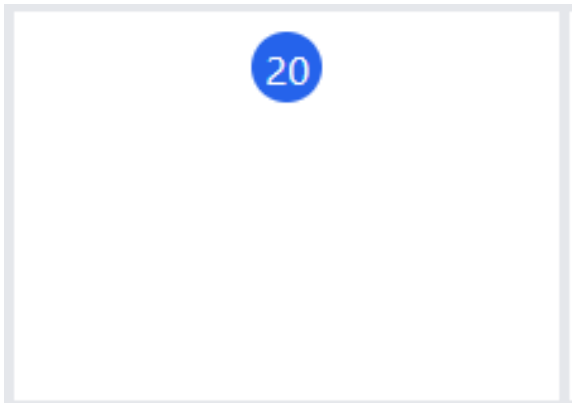


Figure 4.8: Day component of the Current Day indicated by blue circle around day of months

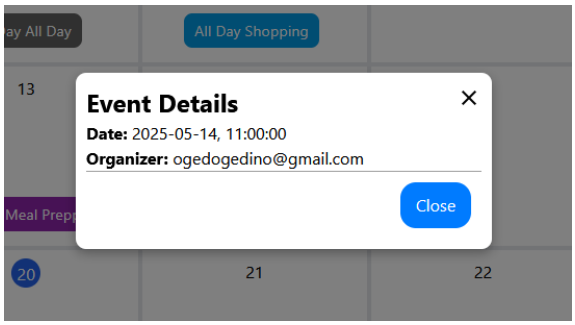


Figure 4.9: Day component of the Current Day indicated by blue circle around day of months

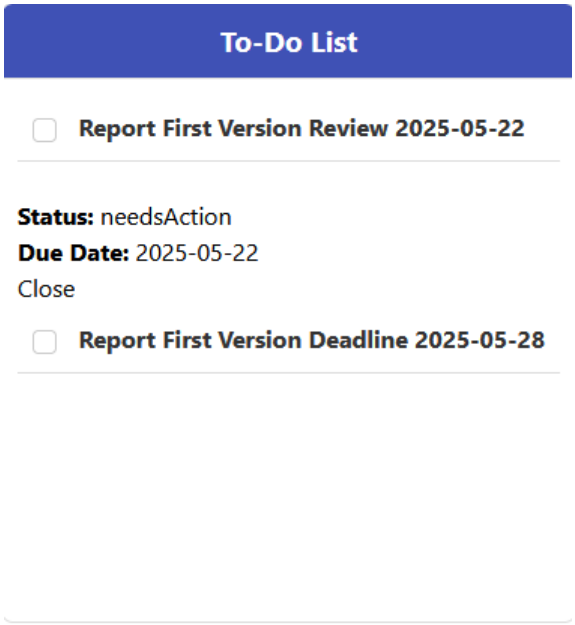


Figure 4.10: Day component of the Current Day indicated by blue circle around day of months

5

Conclusion

5.1 Would have done differently

Throughout the project many lessons have been learned. Even up until the writing of this thesis report.

When collecting figures for the result section of the report it was discovered that one of the features from the prototyping of events in the calendar grid found in figure 4.7 was not working as intended. It was discovered that the color coding of the events was not working as intended, leading to all of the events having a lavender color even through the data collected gave a different color code. Thankfully this problem was a relatively easy fix and just needed to change some values in an array to match the color codes given from Google Calendar API. The lesson was that newly implemented features does not always work with the features already in the project. And this issue highlights the advantages of making automated tests and making sure each previous feature when implementing new one so nothing goes undetected.

Another lesson learned was the importance of spending time on planning before development. Many of the features that were most troublesome were the features that were assumed to be understood before working. Every time that the features where thoroughly thought out, planned and researched before development was started were substantially less troublesome to develop and implement.

5.2 Future of the project

The current iteration of the unified calendar and task management application serves as a robust proof-of-concept. However, there are several future improvements required to expand the functionality, improve user experience and enhance its utility as a comprehensive productivity tool.

Adding More Service Integrations. The application demonstrates effective integration with Google Calendar and Google Tasks. While this provides value, further expanding the comparability to other popular productivity platforms such as Microsoft Outlook, Calendar, Todoist to name a few would significantly broaden

the application's utility and possible user base. This future step would involve research into popularity of the services, their respective APIs and authentication protocols and these individual services, followed by the design and implementation of modular integration layer that ensure data synchronization and display without compromising the performance.

Centralized User Account Management for Services. Presently, the application manages connection to external services directly. As more services integrations are added, a more streamlined systems to managing the users service accounts becomes more important. Implementing a dedicated user account for the web application itself would allow users to link, unlink and manage their service accounts such as Google and Outlook. This enhancement would ensure a more robust and scalable user experience as the number of integrated services grows.

Unified Data Model for Events and To-Dos The current architecture directly accesses the data models given by Google Tasks and Google Calendar. To simplify the implementation of additional services and ensure data consistency across different sources, creating a universal data model for events and tasks is crucial. This would involve designing and implementing a standardized internal data structure for all calendar events and tasks. This will require data translators or adapters each responsible for translating their own services into the standardized internal model, and vice-versa if add, delete and edit functionalities are added, promoting a more modular and maintainable code base.

Enhanced Information within Modals The current modals doesn't provide much farther information than the elements they are representing. This means that the current modals are basically just placeholders for further development. Creating clear concise user friendly modals for the application would improve the user experience and display a more comprehensive list of details. This enhancement would involve identifying key information fields that are relevant and missing from the current modals.

Calendar Selection Functionality The application currently defaults to displayiing the events from the primary Google Calendar associated with the user's account. To provide users with greater control and customization, a feature to select and deselect specific calendars from their integrated services is essential. This would involve implementing improvements to the implementation of the Google Calendar service itself accessing the list of calendars from the account instead of just the primary account then moving on to access all the calendars on the accounts the same way we access the primary calendar in the current solution. Then implementing a selection method for which of the calendars from the users integration that the user wants to display inside of the application.

Implementing Edit and Add Functionality Currently the applications functionality is limited to reading, making this a display tool. To evolve this application into a comprehensive productivity solution, enabling users to edit, remove and add events and tasks directly inside the unified interface is a crucial next step. This would involve leveraging the integrated services APIs for write operations. To accomplish this implementation there would need to be an implementation of a intuitive user

interface element, and implementing necessary API calls to manage their schedule and tasks without leaving the application, thereby maximizing the efficiency.

5.3 Summary

In examination project, a web application has been designed, and developed using the React framework. The objective was a unified interface for compiling and visualizing information from various services. The project involved integration with Google APIs via OAuth 2.0 and followed an adapted, sprint-based methodology for iterative development. The application achieved the goal of offering a centralized platform with key features such as dynamic calendar views, consolidated tasks lists, and detail views, demonstrating the potential for more integrated productivity manager.

Bibliography

[1] *Authenticate to Google Workspace APIs by using OAuth 2.0 client credentials | SAP*. en. URL: https://cloud.google.com/solutions/sap/docs/abap-sdk/on-premises-or-any-cloud/latest/authentication-oauth-client-credentials (visited on 05/22/2025).

[2] *Documentation for Visual Studio Code*. en. URL: https://code.visualstudio.com/docs (visited on 05/22/2025).

[3] *Figma: The Collaborative Interface Design Tool*. en. URL: https://www.figma.com/ (visited on 05/22/2025).

[4] *GitHub · Build and ship software on a single, collaborative platform*. en. 2025. URL: https://github.com/ (visited on 05/22/2025).

[5] Google LLC. *Google Calendar API overview*. en. URL: https://developers.google.com/workspace/calendar/api/guides/overview (visited on 04/28/2025).

[6] *Google Tasks API*. en. URL: https://developers.google.com/workspace/tasks/reference/rest (visited on 05/22/2025).

[7] Meta Platforms, Inc. *React Reference Overview – React*. en. URL: https://react.dev/reference/react (visited on 04/28/2025).

[8] *Obsidian - Sharpen your thinking*. en. URL: https://obsidian.md/ (visited on 05/22/2025).

[9] *Scrum Sprints: Everything You Need to Know*. URL: https://www.atlassian.com/agile/scrum/sprints (visited on 05/22/2025).

[10] *What Is an API (Application Programming Interface)? | IBM*. en. Apr. 2024. URL: https://www.ibm.com/think/topics/api (visited on 05/22/2025).

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY